Nonlinear Hyperelasticity-based Mesh Optimisation

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Jordi Paul

im Oktober 2016

Dissertation

Nonlinear Hyperelasticity-based Mesh Optimisation

Fakultät für Mathematik
Technische Universität Dortmund

Dissertation eingereicht am: 14. 10. 2016

Tag der mündlichen Prüfung: 29. 03. 2017

Erstgutachter: Prof. Dr. Stefan Turek

Zweitgutachter: Prof. Dr. Christian Meyer

Acknowledgements

There are many people I wish to thank. First of all my colleagues M. Geveler, D. Ribbrock and P. Zajac, who have been working on the FEAT3 project since 2010. Without all the work they did and continue to do, I would neither have been able to start where I did nor to end up where I wanted. I want to thank S. Basting for his collaboration on mesh optimisation and other topics, and P. Zajac again for typo hunting and suggestions on the presentation of solver configurations. Thanks goes to Prof. C. Meyer of TU Dortmund for his suggestions with regard to nonconvex optimisation and approximations of the Hessian, and to my supervisor, Prof. S. Turek, for his support and guidance.

Furthermore, I want to thank all other colleagues, friends and family, who had to put up with me during the last years. Your patience, friendship and moral support were essential for my mental health.

Dortmund, Friday 14th October, 2016

Jordi Paul

Contents

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 ([CR72], [SF88], [Bra07]). For many partial differential equations, there are error estimates that (under certain assumptions) guarantee convergence of the discrete to the analytical solution of a certain order, if the space discretisation (which indirectly means the mesh) is sufficiently refined. These error estimates directly or indirectly depend on the used mesh, most notably on the cell shapes, sufficient resolution in regions of interest and that it captures the geometric features of the domain of interest. In this work, I will call this partial differential equation the *original* partial differential equation to set it apart from the mesh optimisation problem, as mesh optimisation is just a tool to create or improve a suitable space discretisation of the original PDE. Because it is already an involved topic, I will focus on the mesh optimisation problem itself, and not treat the original PDE at all, meaning that I only treat the meshes themselves and do not solve any PDE on them.

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 [GP92], [BCR$^+$00], [Bän01] for examples) for flow problems, fluid-structure interactions (e.g. [Wic11], [Ric15] [Bas16]) or even phase transformations (e.g. [BPS13], [BBK15]). A moving mesh introduces the problem of geometric stability, which means that intersections of cells (which are fatal for the finite element solution of the original partial differential equation at hand) 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.

Another common scenario is that some cell size distribution is to be enforced or approximated on a given mesh, e.g. when regions of interest are known a priori (for example near the solid walls of a channel flow at medium to high Reynolds numbers, to resolve the resulting boundary layers) or a posteriori (e.g. in an adaptive method using a posteriori error estimates). Then the space discretisation should be "finer" in these regions, which can be achieved by a variety of techniques, including h, p and r-adaptivity, where only the latter (*rezoning* adaptivity) will be treated in this work. Each technique has its own set of advantages and disadvantages and the choice of the most suitable one is highly dependent on the situation, like regularity of the solution,

geometric complexity, shape type of the mesh (like simplices or hypercubes) and its quality, the finite element spaces used etc. For *r*-adaptivity, where only mesh vertices are moved in space to increase the mesh resolution in regions of interest, the distinctive advantage is that it does not modify the structure of the finite element spaces defined on the mesh, meaning it does not introduce new degrees of freedom or modify the adjacency structure of the existing ones. When multilevel solvers are used, this is a very desirable feature.

If the domain of interest contains interior surfaces that need to be resolved, this can be done in several different ways, including fictitious boundary, a phase field or levelset representation, or a sharp interface representation, each again with different advantages and disadvantages. Implicit representations like the first three share the advantage that the mesh does not need to capture the surface exactly, although it still needs to be sufficiently resolved. They also allow topology changes (e.g. two droplets merging), which is desirable from a modelling point of view, although in many applications (like the aforementioned merging droplets) the physics of the modelled process are unclear. However, the finite element spaces defined on a mesh not capturing e.g. regions with different material properties exactly will have considerably worse approximation properties (see [LMWZ10]). The order of approximation is the strongest advantage of the sharp interface representation, but it does in general not allow topology changes, and extreme deformations (or relative movement) of the corresponding surfaces can be a problem.

The mentioned aspects can be addressed by a variety of methods, see [HR11] for an overview. One possibility is to use geometric information (like vertex coordinates, distances etc.) in algorithms that try to improve the mesh quality, or move vertices to some region of interest. This may require a lot of combinatorial effort and many ideas do not easily generalise to higher space dimensions. A different approach is to define a functional measuring the energy of the deformation of a reference mesh to the current mesh and then to minimise this functional to obtain a new mesh. One common approach is to use the mesh's vertex adjacency graph and solve a system of equations based on the graph-Laplacian to obtain new vertex coordinates. Usually, some weighting is done on the graph (see [Mün16] for a recent work). This technique is very efficient computationally, but cannot guarantee that the resulting mesh is free of intersecting cells. To have results with known properties, it is very common to use a mesh quality functional that is based on some partial differential equation. An important example is solving Laplace's equation for the vertex coordinates, which is equivalent of finding the stationary point of the harmonic energy of the deformation. Other well-known partial differential equations can be used to derive mesh quality functionals, and the available analysis for these equations gives an idea of how the solution (meaning the resulting mesh) will behave with regard to boundary conditions or changes in geometry.

In the following, I will only treat mesh quality functionals that are derived from some PDE because of the analysis available and because the tools for the solution of the mesh optimisation problem are essentially already present, as they are the same that will be employed for solving the original PDE. One more aspect to mention is that, in the current formulation, the optimised mesh is directly obtained by minimising the mesh quality functional. Combining the mesh problem and the original, transient PDE into one system of equations yields a differential algebraic system, which is difficult to solve. One possible solution to this problem is to formulate the mesh quality functional in terms of the *speeds* of the mesh's vertices (as opposed to their coordinates), so the resulting system looses its differential algebraic structure. Another approach is to decouple the original PDE from the geometry problem even it they are coupled. Even when this decoupling of mesh and original PDE problems is performed, it is still possible to formulate the mesh quality functional in terms of vertex speeds and then solve an ordinary differential equation in time (or pseudo-time in the case of stationary problems) like in [GKT08] [GKT10]. Since I do not even solve the original PDE in this work (as mentioned above), I chose to formulate the mesh optimisation problem directly in the vertex coordinates.

The purpose of this work is to expand and improve an idea from [Rum96], which is to derive a mesh quality functional using the stored-energy functional of a hyperelastic material. This allows the use of all the theory available from mathematical elasticity ([Bal76], [Cia88]) to establish the existence and nonuniqueness of minimisers. The idea of surface alignment from [BW13] is to be expanded and combined with *r*-adaptivity. It will be shown how the class of functionals at hand allows a very direct control of the cell sizes by specifying an optimal size for every cell, which is much more direct than the monitor functions used in other methods. In [Rum96] and [BW13], only simplex meshes were considered, but the concepts will be carried over to hypercube meshes in this work. However, the numerical effort to solve the resulting mesh optimisation problem is very high, as it is a nonconvex nonlinear problem with many solutions, which are local minimisers of the mesh quality functional. For this, a preconditioner is introduced, which makes the solution process much more efficient in the cases where it is applicable.

This work is organised as follows

In Chapter 2, some notation will be given, followed by definitions of the terms that will be used to describe meshes and the entities they are comprised of. Theory about the approximation properties of finite element spaces is given for motivational purposes and to show where some of the terms later considered in the mesh optimisation problem enter the interpolation error estimates.

The main chapter is Chapter 3. Here, the notion of mesh quality and how it can be defined to fit a variety of purposes is discussed, with some rather simple but illustrative examples in Section 3.3 and Section 3.4. A mesh quality functional is derived in Section 3.5 from some (reasonable) assumptions and desirable properties for a notion of mesh quality. Only afterwards, the connection to hyperelastic materials is made. For hyperelastic materials, there are well-known existence results, which will be given in Section 3.5.6.

In chapter Chapter 4, numerical methods for the solution of the minimisation problem are introduced and discussed. The finite element spaces for the space discretisation are straightforward, as they are defined by the transformation used for mapping reference cells to mesh cells. Solvers are introduced in Section 4.2, where the focus is on line search based methods for large scale unconstrained optimisation. This class of methods offers the necessary flexibility for the class of preconditioners that will be introduced later, and based on the properties of the functionals at hand we cannot expect good results from a full Newton method. Since the aforementioned surface alignment can be realised through constraints, aspects of constraint optimisation are given in Section 4.3. Some examples are given in Section 4.4 to illustrate the difficulties to be expected when minimising nonconvex functionals without constraints.

The first set of numerical results is presented in Chapter 5. Here, only aspects of the method, its properties and the behaviour are discussed, while everything solver related (except for an examination of the characteristics of the quadratic penalty iteration from Section 4.3 in Section 5.6.1) can be found in Chapter 7. First, heuristics for the cell shape quality and the cell size distribution defect that are independent of any mesh quality functional used are introduced in Section 5.2. These will be used to interpret the quality of the meshes generated in this section. For the stationary model problem in Section 5.3 and the moving boundary problems presented Section 5.4, comparisons between the results obtained by a quadratic, PDE-based mesh quality functional and hyperelasticity-based functionals are done. The aspects of *r*-adaptivity and alignment to (implicit) surfaces are treated in Section 5.5 and Section 5.6 respectively. Here, no comparisons with mesh quality functionals of other type are possible.

In Chapter 6, a family of preconditioners for the minimisation of hyperelasticity-based mesh quality functionals motivated by the functional's properties on the continuous level is introduced. There are known limitations with regard to the applicability of this preconditioner (see Section 6.2), and it will be shown that because of the structure of the mesh quality functionals, the precon-

ditioners cannot be expected to give good convergence results in corner cases.

However, for a good range of moving boundary problems, the family of preconditioners substantially accelerates the iterative solver for the minimisation problem, as is shown in Chapter 7. Since the solution is nonunique, the use of a preconditioner often makes it possible to find better local minima of the mesh quality functional, resulting in meshes of better quality.

2

Preliminaries

In this chapter, I will introduce some basic notation in Section 2.1 and define various geometric terms like simplices, hypercubes, edges, faces, vertices and meshes comprised of these entities in Section 2.2. Finite elements will be introduced in Section 2.3, as will be spaces of parametric finite elements. In this context, it is easier to deal with affine-equivalent finite elements, so interpolation error estimates for this special case are presented first in Section 2.4 as they convey the basic concepts in an easier fashion. The more general case of nonlinear reference cell transformations is treated in Section 2.5. Note that these estimates will not be used directly in proofs, but for motivational purposes as to why mesh optimisation is approached in a PDE-based manner in Chapter 3.

2.1. Notation

The term *iff* means *if and only if*.

Sets

i) $\mathbb{N}$ is the set of positive integers and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$.

ii) $\operatorname{card}(M)$ is the *cardinality* of the set M.

iii) $\operatorname{vol}_n(M)$ is the *n-dimensional volume* of a measurable set $M \subset \mathbb{R}^d, 1 \leq n \leq d$. In the case $n = d$, the symbol $\operatorname{vol}(M)$ is used. For $\mathbb{R}^d$, the *volume measure* will be denoted dx and the *area* or *surface measure* will be denoted $d\sigma$.

iv) For an open subset $\Omega \subset \mathbb{R}^n$, $\lambda \in [0,1]$ and $k \in \mathbb{N}$, the *boundary* $\partial\Omega$ is said to be of class $\mathcal{C}^{0,\lambda}$ or $\mathcal{C}^k$ iff it can be parametrised by a finite number of maps of the corresponding class, see below.

Matrices

If $A \in \mathbb{R}^{m\times n}$ (a member of the vector space of real $m \times n$ matrices), its elements will be denoted by A_{ij}. The *inner product* of this space is denoted by

$$A : B := \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}B_{ij}.$$

If $A \in \mathbb{R}^{n\times n}$:

i) $\det(A)$ is the *determinant* of A.

ii) $\mathrm{GL}_d := \{A \in \mathbb{R}^{d\times d} : \det(A) \neq 0\}$ is the *general linear group* of invertible matrices.

iii) $\mathrm{SL}_d := \{A \in \mathbb{R}^{d\times d} : \det(A) > 0\}$ is the *special linear group* of invertible matrices with positive determinant.

iv) $\mathrm{Cof}(A)$ is the *cofactor matrix* of A. If $A \in \mathbb{R}^{n\times n}$, define $A^{(i,j)} \in \mathbb{R}^{n-1\times n-1}$ as the matrix obtained by deleting the ith column and jth row of A. Then $\mathrm{Cof}(A)_{i,j} = (-1)^{i+j}\det(A^{(i,j)})$ and if $A \in \mathrm{GL}_n$, $\mathrm{Cof}(A) = \det(A)A^{-T}$.

v) $\mathrm{tr}(A)$ is the *trace* of $A \in \mathbb{R}^{n\times n}$, $\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii}$.

Functions and derivatives

Let X, Y be real normed vector spaces and $\Omega \subset X$ an open subset and $f : \Omega \to Y$.

i) Then f is *differentiable* iff

$$\exists f'(x) \in \mathbb{L}(X,Y) : f(x+h) = f(x) + f'(x)h + o(h),$$

where the space $\mathcal{L}(X,Y)$ is defined below. $f'(x)$ is the *(Fréchet) derivative* of the mapping f at $x \in \Omega$.

ii) The *Gâteaux* or *directional derivative* in direction $h \in X$ is defined as

$$f'(x)h := \lim_{t\to 0}\frac{f(x+th)-f(x)}{t} = \frac{d}{dt}f(x+th)_{|t=0} \in Y.$$

Let $\Omega \subset \mathbb{R}^n$ be open, $f : \Omega \to \mathbb{R}$ and denote by $e_i, i = 1,\ldots,n$ the canonical basis of $\mathbb{R}^n$.

i) For $x \in \Omega$ define the *partial derivative* $\frac{\partial f}{\partial x_i}(x) := \lim_{h\to 0}\frac{f(x+he_i)-f(x)}{h}$ if the limit exists.

ii) $\frac{\partial f}{\partial x_i}$ may be abbreviated to $\partial_i f$.

iii) For a *multiindex* $\alpha \in \mathbb{N}_0^n$, define $|a| := \sum_{i=1}^{n}\alpha_i$ and for $x \in \Omega$

$$D^\alpha f(x) := \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1}\ldots\partial x_n^{\alpha_n}}$$

and the *set of all partial derivatives of order* $k \in \mathbb{N}$ as

$$D^k f(x) := \{D^\alpha f(x) : |\alpha = k|\}.$$

iv) In general, the notations

$$f'(x) := D^1 f(x) := (\partial_1 f(x),\ldots,\partial_n f(x))^T \in \mathbb{R}^n$$

for the gradient and

$$f''(x) := D^2 f(x) := (\partial_j\partial_i f(x))_{ij} \in \mathbb{R}^{n\times n}$$

for the Hessian will be used when f is continuously (or twice continuously, respectively) differentiable.

Function spaces

If E, F are topological spaces:

i) Denote by $\mathcal{C}^0(E,F)$ the space of all *continuous* mappings from E to F and $\mathcal{C}^0(X) := \mathcal{C}^0(X,\mathbb{R})$.

ii) $\mathcal{L}(E,F)$ is the space of all continuous *continuous linear* mappings from E to F.

iii) If E and F are normed vector spaces, the space $\mathcal{L}(E,F)$ is a normed vector space equipped with the norm

$$\forall A \in \mathcal{L}(E,F) : \|A\| := \sup_{x \in E\setminus\{0\}} \frac{\|Ax\|_F}{\|x\|_E}.$$

If $\Omega \subset \mathbb{R}^n$ is open and Y is a normed vector space:

i) For $k \in \mathbb{N}, 1 \leq k \leq \infty$, denote by $\mathcal{C}^k(\Omega,Y)$ the space of all k times continuously differentiable mappings from Ω to Y and $\mathcal{C}^k(\Omega) := \mathcal{C}^k(\Omega,\mathbb{R})$.

ii) $\forall \lambda \in [0,1], \mathcal{C}^{0,\lambda}(\Omega)$ is the space of Hölder continuous mappings.

iii) Denote by $L^p(\Omega)$ the space of equivalence classes of dx-almost everywhere (a.e.) equal functions $f : \Omega \to \mathbb{R}$ such that $|f|_{0,p,\Omega} < \infty$, where

$$|f|_{0,p,\Omega} := \begin{cases} \left(\int_\Omega |f(x)|^p dx\right)^{\frac{1}{p}}, & 1 \leq p < \infty \\ \inf_{r\geq 0}\{\mathrm{vol}(\{x \in \Omega : f(x) \geq r\}) = 0\}, & p = \infty \end{cases}.$$

iv) $\forall 1 \leq p \leq \infty$ denote the *Sobolev spaces*

$$W^{m,p}(\Omega) := \{f \in L^p(\Omega) : \forall |\alpha| \leq m : D^\alpha f \in L^p(\Omega)\}.$$

and let $W_0^{m,p}(\Omega)$ be the closure of $\{f \in \mathcal{C}^\infty(\Omega) : \mathrm{supp}\, f \text{ is compact }\}$ in $W^{m,p}(\Omega)$.

v) Denote the *Sobolev seminorms* and *Sobolev norms* by

$$|f|_{m,p,\Omega} := \begin{cases} \left(\int_\Omega \sum_{|\alpha|=m} |D^\alpha f(x)|^p dx\right)^{\frac{1}{p}}, & 1 \leq p < \infty \\ \max_{|\alpha=m|} |D^\alpha f|_{0,\infty,\Omega}, & p = \infty \end{cases},$$

$$\|f\|_{m,p,\Omega} := \begin{cases} \left(\int_\Omega \sum_{|\alpha|\leq m} |D^\alpha f(x)|^p dx\right)^{\frac{1}{p}}, & 1 \leq p < \infty \\ \max_{|\alpha\leq m|} |D^\alpha f|_{0,\infty,\Omega}, & p = \infty \end{cases}.$$

vi) Denote

$$H^m(\Omega) := W^{m,2}(\Omega), \quad H_0^m(\Omega) := W_0^{m,2}(\Omega),$$
$$\|f\|_{m,\Omega} := \|f\|_{2,m,\Omega}, \quad |f|_{m,\Omega} := |f|_{2,m,\Omega}.$$

Differential operators

If for $d = 1,2,3$ $\Omega \subset \mathbb{R}^d$ is open and $\phi : \Omega \to \mathbb{R}^d$.

i) $\nabla\phi : \Omega \to \mathbb{R}^{d\times d}$ is the *deformation gradient* of the mapping ϕ and

$$\forall x \in \Omega : \forall 1 \leq i,j \leq d : (\nabla\phi(x))_{ij} = \frac{\partial \phi_j}{\partial x_i}(x).$$

ii) $\mathbf{D}(\phi) := \frac{1}{2}(\nabla\phi + (\nabla\phi)^T)$ is the *symmetric part* of the deformation gradient of the mapping ϕ.

2.2. Geometric entities and meshes

Definition 2.1 (s-dimensional simplices)**.** *Let $s \in \{1,\dots,d\}$ and $a_0,\dots,a_s \in \mathbb{R}^d$ such that $\forall i \in \{1,\dots,s\} : (a_i - a_0)$ are pairwise linearly independent.*

i) Then

$$S := \left\{ x \in \mathbb{R}^d : x = \sum_{i=0}^{s} \lambda_i a_i, \text{ where } \forall i \in \{1,\dots,s\} : \lambda_i \in \mathbb{R}_{\geq 0}, \sum_{i=0}^{s} \lambda_i = 1 \right\} \tag{2.1}$$

is called (non-degenerate) s-dimensional simplex in $\mathbb{R}^d$*, $a_0,\dots,a_s$ are called its* vertices *and* $\mathcal{V}(S) := \{a_0,\dots,a_s\}$.

ii) $\forall x \in S$, $\lambda_0,\dots,\lambda_s \in [0,1]$ such that $x = \sum_{i=0}^{s} \lambda_i a_i, \sum_{i=0}^{s} \lambda_i = 1$ are called the barycentric coordinates of x with regard to S.

iii) For $r \in \{1,\dots,s\}$ and $a'_0,\dots,a'_r \in \{a_0,\dots,a_s\}$ pairwise unequal,

$$S' := \left\{ x \in \mathbb{R}^d : x = \sum_{i=0}^{r} \lambda_i a'_i, \text{ where } \forall i \in \{1,\dots,r\} : \lambda_i \in \mathbb{R}_{\geq 0}, \sum_{i=0}^{r} \lambda_i = 1 \right\}$$

is called r-dimensional sub-simplex of S.

iv) The s-simplex defined by the vertices $a_0 = 0, \forall i = 1,\dots,s : a_i = e_i$ is called the s-dimensional unit simplex $\hat{S}$.

v) The mapping $R : \hat{S} \to S$ defined by

$$R(\hat{x}) = a_0 + \sum_{i=1}^{s} \hat{x}^i a_i$$

is called the parametrisation of S over $\hat{S}$*. Because it is linear, we also call it* $\mathbb{P}_1$-parametrisation or -transformation.

Note that in this context, simplices are always straight simplices because the parametrisation F is linear by definition. Hypercubes are more difficult to describe because there is no equivalent to the barycentric coordinates λ_i from 2.1, so we have to define them through the parametrisation F.

Definition 2.2 (s-dimensional hypercubes)**.** *Let $s \in \{1,\dots,d\}$.*

i) $\hat{Q}_s := [-1,1]^s$ is called the s-dimensional reference hypercube*. For $s > 1$, its* faces *or $(s-1)$-dimensional sub-hypercubes are the sets*

$s = 2$:

$$\begin{aligned} \hat{Q}'_0 &:= [-1,1] \times \{-1\}, \quad \hat{Q}'_1 := [-1,1] \times \{1\}, \\ \hat{Q}'_2 &:= \{-1\} \times [-1,1], \quad \hat{Q}'_3 := \{1\} \times [-1,1], \end{aligned}$$

$s = 3$:

$$\begin{aligned} \hat{Q}'_0 &:= [-1,1] \times [-1,1] \times \{-1\}, \quad \hat{Q}'_1 := [-1,1] \times [-1,1] \times \{1\}, \\ \hat{Q}'_2 &:= [-1,1] \times \{-1\} \times [-1,1], \quad \hat{Q}'_3 := \times [-1,1] \times \{1\} \times [-1,1], \\ \hat{Q}'_4 &:= \{-1\} \times [-1,1] \times [-1,1], \quad \hat{Q}'_5 := \{1\} \times [-1,1] \times [-1,1]. \end{aligned}$$

In the case $s = 3$, the 2-dimensional reference sub-hypercubes $\hat{Q}'_k$ are 2-dimensional reference hypercubes embedded in 3d, so their faces are well defined and form the 1-dimensional reference sub-hypercubes of $\hat{Q}_3$.

ii) Define the functions $\phi_i : \hat{Q}_s \to \mathbb{R}^d, i \in \{0, \dots 2^d - 1\}$ by

$s = 1$:

$$\phi_0(\hat{x}) := \frac{1}{2}(1-\hat{x}^0), \quad \phi_1(\hat{x}) := \frac{1}{2}(1+\hat{x}^0),$$

$s = 2$:

$$\phi_0(\hat{x}) := \frac{1}{4}(1-\hat{x}^0)(1-\hat{x}^1), \quad \phi_1(\hat{x}) := \frac{1}{4}(1-\hat{x}^0)(1+\hat{x}^1),$$
$$\phi_2(\hat{x}) := \frac{1}{4}(1-\hat{x}^0)(1+\hat{x}^1), \quad \phi_3(\hat{x}) := \frac{1}{4}(1+\hat{x}^0)(1+\hat{x}^1),$$

$s = 3$:

$$\phi_0(\hat{x}) := \frac{1}{8}(1-\hat{x}^0)(1-\hat{x}^1)(1-\hat{x}^2), \quad \phi_1(\hat{x}) := \frac{1}{8}(1+\hat{x}^0)(1-\hat{x}^1)(1-\hat{x}^2),$$
$$\phi_2(\hat{x}) := \frac{1}{8}(1-\hat{x}^0)(1+\hat{x}^1)(1-\hat{x}^2), \quad \phi_3(\hat{x}) := \frac{1}{8}(1+\hat{x}^0)(1+\hat{x}^1)(1-\hat{x}^2),$$
$$\phi_4(\hat{x}) := \frac{1}{8}(1-\hat{x}^0)(1-\hat{x}^1)(1+\hat{x}^2), \quad \phi_5(\hat{x}) := \frac{1}{8}(1+\hat{x}^0)(1-\hat{x}^1)(1+\hat{x}^2),$$
$$\phi_6(\hat{x}) := \frac{1}{8}(1-\hat{x}^0)(1+\hat{x}^1)(1+\hat{x}^2), \quad \phi_7(\hat{x}) := \frac{1}{8}(1+\hat{x}^0)(1+\hat{x}^1)(1+\hat{x}^2).$$

Let $a_0, \dots, a_{2^s-1} \in \mathbb{R}^d$ and define the map

$$R : \hat{Q}_s \to \mathbb{R}^d, \quad R(\hat{x}) = \sum_{i=0}^{2^d-1} \phi_i(\hat{x}) a_i$$

and $Q := R(\hat{Q})$. Q is called non-degenerate s-dimensional hypercube *iff*

$$\forall \hat{x} \in \hat{Q} : \begin{cases} \nabla R(\hat{x}) \in \mathrm{GL}_d, & s = d \\ \nabla R(\hat{x})^T \nabla R(\hat{x}) \in \mathrm{GL}_d, & s < d \end{cases},$$

$a_0, \dots, a_{2^s-1}$ are called its vertices *and R is called the* parametrisation of Q over $\hat{Q}$. *Because it is bilinear by definition in general, we also call it the* $\mathbb{Q}_1$-parametrisation or $\mathbb{Q}_1$-transformation.

For $l \in \{0, \dots, 2s-1\}$, the sets $R(\hat{Q}'_l)$ are called $(s-1)$-dimensional sub-hypercubes *and are non-degenerate if Q is non-degenerate. For $s = 3$, each 2-dimensional sub-hypercube again has 1-dimensional sub-hypercubes. Note that this more general than what is usually defined as a hypercube, but for the sake of brevity I will not use the term "generalised hypercube".*

The converse of the non-degeneracy implication is not true, however. Note that for 3-hypercubes, its 2-subhypercubes are not planar in general, due to the $\mathbb{Q}_1$-transformation being nonlinear. It is possible to show that the $\mathbb{Q}_1$-transformation $R : \hat{Q} \to Q$ is linear if and only if Q is a parallelpiped (sometimes called a rhomboid).

Definition 2.3. *Let $K \subset \mathbb{R}^d$.*

i)
$$h(K) := \sup\{\|x_1 - x_0\|_2 : x_0, x_1 \in K\}$$
is called the diameter *of K.*

ii)
$$\rho(K) := 2\sup\{r : \exists x_0 \in K : B_r(x_0) \subset K\}$$
is called the in-circle diameter *of S.*

iii)
$$\sigma(K) := \frac{h(K)}{\rho(K)}$$
is called the aspect ratio *of K.*

Definition 2.4. *Two subsets $\emptyset \neq \hat{K}, K \subset \mathbb{R}^d$ are called* affine-equivalent *iff $\exists B \in \mathbb{R}^{d\times d}, \hat{x}_0 \in \mathbb{R}^d$ such that for $R : \mathbb{R}^d \to \mathbb{R}^d$ defined by $R(\hat{x}) := B\hat{x} + \hat{x}_0$ it holds that*
$$R(\hat{K}) = K.$$

Definition 2.5. *Let $\Omega \subset \mathbb{R}^d$ be a polygonally bounded domain. A finite set $\mathcal{T}$ of d-simplices or d-hypercubes is called* partitioning of Ω *or* mesh on Ω *iff $\bar{\Omega} = \bigcup_{K\in\mathcal{T}} K$.*
This partitioning is called conforming *or* proper *iff*

i) $\forall K_0, K_1 \in \mathcal{T}, K_0 \neq K_1 : \mathring{K}_0 \cap \mathring{K}_1 = \emptyset$,

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

Definition 2.6 (Geometric entities)**.** *Let $\Omega \subset \mathbb{R}^d$ be a polygonally bounded domain and $\mathcal{T}$ a partitioning of Ω, meaning that $\bar{\Omega} = \bigcup_{K\in\mathcal{T}} K$.*

i) $\forall K \in \mathcal{T} : \mathring{K}$ *is called a d-*cell *of $\mathcal{T}$ and $\mathcal{E}^d(\mathcal{T}) := \{\mathring{K} : K \in \mathcal{T}\}$ is the set of all d-cells.*

ii) For $r \in \{1, \dots, d-1\} : \forall K \in \mathcal{T}$ and its r-dimensional sub-hypercubes or sub-simplices K' are called r-cells and $\mathcal{E}^r(\mathcal{T})$ is the set of all r-cells of $\mathcal{T}$.

iii) $\mathcal{V}(\mathcal{T}) := \mathcal{E}^0(\mathcal{T}) = \bigcup_{K\in\mathcal{T}} \mathcal{V}(K)$ *is the set of all vertices of $\mathcal{T}$.*

In practice, 1*-cells are more often referred to as* edges *and, for $d = 3$,* 2*-cells as* faces*. For all $d = 1, 2, 3$ we will call d-cells simply* cells *and $(d-1)$-cells* facets*.*
Note that all r-cells are open, meaning that we have a disjoint partitioning $\Omega = \biguplus_{r=0}^{d} \mathcal{E}^r(\mathcal{T})$.

Definition 2.7. *Let $(\mathcal{T}_i)_{i\in\mathbb{N}}$ be a family of meshes. It is called a* regular family of meshes *iff*

i) $\sup\{\sigma(K) : K \in \bigcup_{i\in\mathbb{N}} \mathcal{T}_i\} =: \sigma < \infty$,

ii) For $h(\mathcal{T}_i) := \max_{K\in\mathcal{T}_i}$ 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 $(\mathcal{T}_h)$.

2.3. Finite Elements and parametric families of finite element spaces

This part follows [Cia78, Chapter 2].

Definition 2.8 (Finite Element)**.** *Let $K \subset \mathbb{R}^d$ be open, $\mathring{K} \neq \emptyset$, $\partial K \in \mathcal{C}^{0,1}$. Let $P := \{p : K \to \mathbb{R}\}$ and $\Sigma = \{\phi_1, \ldots, \phi_N\} \subset P'$ such that*

$$\forall i = 1, \ldots, N : \exists k_i \in \mathbb{N} : \phi_i(f) = \phi_i(D^\beta(f)), |\beta| = k_i$$
$$\forall \alpha_1, \ldots, \alpha_N \in \mathbb{R} : \exists! p \in P : \forall i = 1, \ldots, N : \phi_i(p) = \alpha_i,$$

*meaning that Σ is P-*unisolvent*. Furthermore, $k_\Sigma := \max\{k_i : i = 1, \ldots, N\}$ and $\Sigma \subset \mathcal{C}^k(K)$.*

i) Then the triple (K, P, Σ) is called a Finite Element*.*

ii) The linear forms ϕ_i are called the degrees of freedom*.*

iii) The functions $p_i \in Pi, i \in \{0, \ldots, N\}$ satisfying $\phi_j(p_i) = \delta_{ij}$ which exist and are unique due to the P-unisolvence, are called the finite element basis functions*.*

By definition, we have $\dim P = \dim \Sigma = N$ and

$$\forall p \in P : p = \sum_{i=1}^{N} \phi_i(p) p_i.$$

Other common names for the degrees of freedom are node functionals *(mostly in connection with conforming Lagrange elements) or* dual basis*.*

This definition can be extended to m-tuples of finite elements in a natural manner.

Definition 2.9 (P-Interpolation Operator)**.** *For a finite element (K, P, Σ) and a function $v \in \mathcal{C}^{k_\Sigma}(K)$, define the P-*interpolant *$I_P v$ by*

$$I_P v = \sum_{i=1}^{N} \phi_i(v) p_i.$$

$I_P v \in P$ is unambiguously defined because of the P-unisolvence of Σ.

Definition 2.10. *Let $(K_i, P_{k_i}, \Sigma_{K_i})_{i \in \mathbb{N}}$ be a family of finite elements. It is called* regular *iff*

i) $\sigma := \sup_{i \in \mathbb{N}} \sigma(K_i) < \infty$*,*

ii) $\lim_{i \to \infty} h(K_i) = 0$*.*

Definition 2.11. *Equivalent finite elements*

Let the reference finite element *$(\hat{K}, \hat{P}, \hat{\Sigma})$ be given. Let $K \subset \mathbb{R}^d$ be open and $R : \hat{K} \to K$ is a diffeomorphism such that $\forall i = 1, \ldots, d : R_i \in G$ with $\exists m \in \mathbb{N} : G \subseteq \mathbb{P}_m(\hat{S})$. Define*

$$P_K := \{p : K \to \mathbb{R} : \exists \hat{p} \in \hat{P} : p = \hat{p} \circ R_K^{-1}\}$$
$$\Sigma_K := \{\hat{\phi}(p \circ R_K) : \hat{\phi} \in \hat{\Sigma}\}$$

Then (K, P_K, Σ_K) is called

i) affine equivalent *to $(\hat{K}, \hat{P}, \hat{\Sigma})$ iff $G = \mathbb{P}_1(\hat{K})$,*

ii) isoparametrically equivalent *to $(\hat{K}, \hat{P}, \hat{\Sigma})$ iff $G = \hat{P}$ or*

iii) subparametrically equivalent *to $(\hat{K}, \hat{P}, \hat{\Sigma})$ iff $G \subsetneq \hat{P}$.*

iv) superparametrically equivalent *to* $(\hat{K},\hat{P},\hat{\Sigma})$ *iff* $G \supsetneq \hat{P}$.

Definition 2.12. *Parametric family of finite elements*

Let a reference element $(\hat{K},\hat{P},\hat{\Sigma})$ *be given.*

A family $(K_i,P_{K_i},\Sigma_{K_i})_{i=1,\dots,N}$ *of finite elements is called either affine, isoparametric, subparametric or superparametric (with regard to the reference element) iff* $\forall i = 1,\dots,N : (K_i,P_{K_i},\Sigma_{K_i})$ *is affine (or isoparametrically or subparametrically or superparametrically, respectively) equivalent* $(\hat{K},\hat{P},\hat{\Sigma})$.

If the family is in either of the three categories, we call it parametric family.

Definition 2.13. *Let* $\Omega \subset \mathbb{R}^d, d = 1,2,3$ *be polygonally bounded and* $\mathcal{T}$ *be a regular and conforming mesh on* Ω. *Define the degrees of freedom*

$$\hat{\Sigma} = \{\phi_i(p) = p(a_i) : a_i \in \mathcal{V}(\hat{K})\} \tag{2.2}$$

and the conforming $\mathbb{P}_k, \mathbb{Q}_k$ *reference elements*

$$(\hat{S},\mathbb{P}_k(\hat{S}),\hat{\Sigma}) \tag{2.3}$$

$$(\hat{Q},\mathbb{P}_k(\hat{Q}),\hat{\Sigma}) \tag{2.4}$$

and the conforming Lagrange spaces

$$\begin{aligned}\mathbb{P}_k(\mathcal{T}) &:= (K,P_K,\Sigma_K)_{K\in\mathcal{T}}, \forall K \in \mathcal{T} : (K,P_K,\Sigma_K) \text{ is affine equivalent to } (\hat{S},\mathbb{P}_k(\hat{S}),\hat{\Sigma})\\ &= \{v \in \mathcal{C}^0(\Omega) : \forall K \in \mathcal{T} : \exists \mathbb{P}_1(\hat{K}) \ni R_K : \hat{K} \to K \text{ a diffeomorphism}\\ &\qquad \text{and } \exists \hat{v} \in \mathbb{P}_k(\hat{K}) : v = \hat{v} \circ R_K^{-1}\}\end{aligned}$$

and

$$\begin{aligned}\mathbb{Q}_1(\mathcal{T}) &:= (K,P_K,\Sigma_K)_{K\in\mathcal{T}}, \forall K \in \mathcal{T} : (K,P_K,\Sigma_K) \text{ is isoparametrically equivalent to } (\hat{Q},\mathbb{Q}_1(\hat{Q}),\hat{\Sigma})\\ &= \{v \in \mathcal{C}^0(\Omega) : \forall K \in \mathcal{T} : \exists \mathbb{Q}_1(\hat{K}) \ni R_K : \hat{K} \to K \text{ a diffeomorphism}\\ &\qquad \text{and } \exists \hat{v} \in \mathbb{P}(\hat{K}) : v = \hat{v} \circ R_K^{-1}\}\end{aligned}$$

2.4. Interpolation estimates for affine-equivalent finite elements

Lemma 2.1. *Let* $\hat{K},K \subset \mathbb{R}^d$ *be affine-equivalent under the mapping* $R : \mathbb{R}^d \to \mathbb{R}^d, R(x) = Bx + x_0, R(\hat{K}) = K$ *and open. Then the estimates*

$$\|B\|_2 \leq \frac{h(K)}{\rho(\hat{K})} \tag{2.5}$$

$$\|B^{-1}\|_2 \leq \frac{h(\hat{K})}{\rho(K)} \tag{2.6}$$

hold.

Proof. See [Cia78, Theorem 3.1.3]. □

Lemma 2.2. *Let* $\hat{K},K \subset \mathbb{R}^d$ *be affine-equivalent under the mapping* $R : \mathbb{R}^d \to \mathbb{R}^d, R(x) = Bx + x_0, R(\hat{K}) = K$ *and open. Let* $m \in \mathbb{N}, p \in [1,\infty]$ *and* $v \in W^{m,p}(K)$. *Define* $\hat{v}(\hat{x}) := v \circ R_K(\hat{x})$. *Then* $\hat{v} \in W(\hat{K})^{m,p}$ *and*

$$\exists C = C(m,d) : \forall v \in W(K)^{m,p} : \quad |\hat{v}|_{m,p,\hat{K}} \leq C(\det(B))^{-\frac{1}{p}} \|B\|^m |v|_{m,p,K}.$$

Analogously, the estimate

$$\exists C = C(m,d) : \forall \hat{v} \in W(\hat{K})^{m,p} : \quad |v|_{m,p,K} \leq C(\det(B))^{\frac{1}{p}} \|B^{-1}\|^m |\hat{v}|_{m,p,\hat{K}}.$$

holds.

Proof. See [Cia78, Theorem 3.1.2]. □

Lemma 2.3. *Let $(\hat{K},\hat{P},\hat{\Sigma})$ be a finite element. Let $k,m \in \mathbb{N}$ and $p,q \in [1,\infty]$ such that*

$$W^{k+1,p} \hookrightarrow \mathcal{C}^{k_\Sigma}(\hat{K}), \tag{2.7}$$
$$W^{k+1,p} \hookrightarrow W^{m,q}(\hat{K}), \tag{2.8}$$
$$P_k(\hat{K}) \subset \hat{P} \subset W^{m,q}(\hat{K}). \tag{2.9}$$

Then there exists a constant $C = C(\hat{K},\hat{P},\hat{\Sigma})$ such that $\forall$ affine-equivalent finite elements (K,P,Σ) and $\forall v \in W^{k+1,p}(K)$ the estimate

$$|v - I_P v|_{m,q,K} \leq C(\hat{K},\hat{P},\hat{\Sigma}) \mathrm{vol}(K)^{\left(\frac{1}{q}-\frac{1}{p}\right)} \frac{h(K)^{k+1}}{\rho(K)^m} |v|_{k+1,p,K}$$

Proof. See [Cia78, Theorem 3.1.5]. □

Lemma 2.4. *Let $(K_i,P,\Sigma)_{i\in\mathbb{N}}$ be a regular affine family of finite elements, whose reference finite element $(\hat{K},\hat{P},\hat{\Sigma})$ satisfies the assumptions* (2.7) *to* (2.9)*. Then there exists a constant $C = C(\hat{K},\hat{P},\hat{\Sigma})$ such that*

$$\forall K_i : \forall v \in W^{k+1,p}(K_i) : \quad |v - I_P v|_{m,q,K_i} \leq C(\hat{K},\hat{P},\hat{\Sigma}) \mathrm{vol}(K_i)^{\left(\frac{1}{q}-\frac{1}{p}\right)} \frac{h(K_i)^{k+1}}{\rho(K_i)^m} |v|_{k+1,p,K_i}.$$

Proof. See [Cia78, Theorem 3.1.6]. □

Theorem 2.5. *Let $\Omega \subset \mathbb{R}^d$ be polygonally bounded, $(\mathcal{T}_h)$ be a regular family of meshes such that $\forall K \in \mathcal{T}_h : (K,P,\Sigma)$ are affine-equivalent to a single reference finite element $(\hat{K},\hat{P},\hat{\Sigma})$. Let $X_h \subset \mathcal{C}^0(\Omega)$ be the finite element spaces formed by the families of finite elements such that $X_h \subset V \subset H^1(\Omega)$. Let $k,l \in \mathbb{N}, l \leq k$ such that*

$$P_k(\hat{K}) \subset \hat{P} \subset H^l(\hat{K}),$$
$$H^{k+1}(\hat{K}) \hookrightarrow \mathcal{C}^{k_\Sigma}(\hat{K}).$$

Then $\exists C > 0$ independent of h such that

$$\forall v \in H^{k+1}(\Omega) \cap V :$$
$$\forall 0 \leq m \leq \min\{1,l\} : \qquad \|v - I_P v\|_{m,\Omega} \leq C h^{k+1-m} |v|_{k+1,\Omega}$$
$$\forall 2 \leq m \leq \min\{k+1,l\} : \qquad \sqrt{\sum_{K\in\mathcal{T}_h} \|v - I_P v\|_{m,K}} \leq C h^{k+1} |v|_{k+1,\Omega}$$

Proof. See [Cia78, Theorem 3.2.1]. □

2.5. Interpolation estimates for finite elements with nonlinear transformations

In Section 2.4 only affine-equivalent families of finite elements were regarded. This means the transformation $R : \hat{K} \to K$ was always linear, greatly simplifying several estimates. It also allowed the expression of $\|\nabla R\|$ in simple geometric terms (Lemma 2.1). In many finite element methods, it is very important to use transformations that are not linear. Important examples are:

i) The $\mathbb{Q}_1$ transformation (see Definition 2.2), because a linear transformation of a hypercube can map only to parallelpipeds, greatly limiting the geometric flexibility.

ii) Transformations of types $\mathbb{P}_k, \mathbb{Q}_k$ for $k > 1$. These transformations allow for higher order discretisations of the the boundary $\partial\Omega$ and isoparametric finite elements. [Cia78, Chapter 4.4].

In this work, I will almost exclusively use the $\mathbb{Q}_1, \mathbb{P}_1$ transformations.

Lemma 2.6. *Let $\hat{K}, K \subset \mathbb{R}^d$ be open and bounded such that $\exists R : \hat{K} \to K$ sufficiently smooth, one-to-one and with sufficiently smooth inverse $R^{-1} : K \to \hat{K}$. Let $V : \hat{K} \to \mathbb{R}$. If for some $l \geq 0, p \in [1,\infty]$ we have $v \in W^{l,p}\hat{K}$ and there exist constants C such that*

$$\forall \hat{v} \in L^p(\hat{K}) : |v|_{0,p,K} \leq \left(|\nabla R|_{0,\infty,\hat{K}}\right)^{\frac{1}{p}} |\hat{v}|_{0,p,\hat{K}}$$

$$\forall \hat{v} \in W^{1,p}(\hat{K}) : |v|_{1,p,K} \leq C\left(|\nabla R|_{0,\infty,\hat{K}}\right)^{\frac{1}{p}} |R^{-1}|_{1,\infty,K}|\hat{v}|_{1,p,\hat{K}}$$

$$\forall \hat{v} \in W^{2,p}(\hat{K}) : |v|_{2,p,K} \leq C\left(|\nabla R|_{0,\infty,\hat{K}}\right)^{\frac{1}{p}} \left(|R^{-1}|^2_{1,\infty,K}|\hat{v}|_{2,p,\hat{K}} \quad + |R^{-1}|_{2,\infty,K}|\hat{v}|_{1,p,\hat{K}}\right)$$

$$\forall \hat{v} \in W^{3,p}(\hat{K}) : |v|_{3,p,K} \leq C\left(|\nabla R|_{0,\infty,\hat{K}}\right)^{\frac{1}{p}} \left(|R^{-1}|^3_{1,\infty,K}|\hat{v}|_{3,p,\hat{K}} \quad + |R^{-1}|_{1,\infty,K}|R^{-1}|_{2,\infty,K}|\hat{v}|_{2,p,\hat{K}} + |R^{-1}|_{3,\infty,K}|\hat{v}|_{1,p,\hat{K}}\right)$$

Proof. See [Cia78, Theorem 4.3.2] (this is the same lemma as Lemma 2.2 conceptually). □

Theorem 2.7. *Let $\hat{\Omega} \subset \mathbb{R}^d$ be open with $\mathcal{C}^0$-boundary, $R : \hat{\Omega} \to \mathbb{R}^d$, $R(\hat{\Omega}) = \Omega$ be a $\mathcal{C}^k$-diffeomorphism for some $k \in \mathbb{N}$. Let $\hat{\Pi} \in \mathbb{L}(W^{k+1,p}(\hat{\Omega}), W^{m,p}(\hat{\Omega}))$ with $0 \leq m \leq k+1$ and with the property*

$$\forall \hat{v} \in \mathbb{P}_k : \hat{\Pi}\hat{v} = \hat{v}.$$

Define $\Pi \in L(W^{k+1,p}(\Omega), W^{m,p}(\Omega))$ by

$$\forall v \in W^{k+1,p} : \widehat{\Pi v} = \hat{\Pi}\hat{v}.$$

Then there exists a constant $C = C(d,k,m,p,\hat{\Omega},\hat{\Pi})$ such that

$$\forall v \in W^{k+1,p}(\Omega) : |v - \Pi v|_{m,p,\Omega} \leq C\left(\frac{\sup_{\hat{x}\in\hat{\Omega}} |\det(\nabla R(\hat{x}))|}{\inf_{\hat{x}\in\hat{\Omega}} |\det(\nabla R(\hat{x}))|}\right)^{\frac{1}{p}} \left(\sum_{l=1}^{m} \sum_{i\in I(l,m)} \sup_{x\in\Omega} \prod_{\lambda=1}^{m} \|DR^{-1}(x)\|^{i_\lambda}\right) \cdot \left(\sum_{l=1}^{k+1} |v|_{l,p,\Omega} \sum_{j\in I(l,k+1)} \sup_{\hat{x}\in\hat{\Omega}} \prod_{\lambda=1}^{k+1} \|DR(\hat{x})\|^{i_\lambda}\right),$$

where

$$I(l,m) := \{a \in \mathbb{N}^m : \|a\|_1 = l, \sum_{k=1}^{m} k a_k = m\}.$$

Proof. See [CR72, Theorem 1]. □

3

Mesh optimisation

In this chapter, I want to elaborate the term "mesh quality" and show how mesh quality can be defined, measured and improved in a PDE-based manner.

The interpolation estimates in Section 2.4 and Section 2.5 form the basis for estimates of the discretisation error $\|u - u_h\|$ for various problems, different finite element spaces and corresponding norms $\|\cdot\|$. I will not give any examples, as the details depend on the specific situation and instead refer the reader e.g. to [Cia78, Chapters 3 to 7] for various classes of problems.

However, recall the reference mappings $R_K : \hat{K} \to K$ from Definition 2.11 and the global version $R : \hat{\Omega} \to \mathbb{R}^d$ e.g. from Theorem 2.7, where we can also see the dependence of the interpolation error estimate on $\det(\nabla R)$. Also note how Theorem 2.5 and Theorem 2.7 depend on a regular family of meshes $(\mathcal{T}_h)$. So if a finite element method is to provide convergence of the discrete to the continuous solution, e.g.

$$\lim_{h \to 0} \|u - u_h\| = 0,$$

we have to guarantee that $(\mathcal{T}_h)_h$ is a regular and conforming family of meshes. It is possible to prove this property for a variety of mesh refinement strategies, including regular (global) refinement as well as local refinement e.g. by bisecting simplices. However, as soon as the geometry discretised by $\mathcal{T}_h$ is no longer fixed (consider fluid-structure interactions as an example), this becomes increasingly difficult. Many strategies for reducing the error $\|u - u_h\|$ at moderate computational cost (especially without globally refining $\mathcal{T}_h$) by some sort of adaptivity revolve around modifying $\mathcal{T}_h$ locally (see Section 3.2.2), so that the regularity of the resulting family $(\mathcal{T}_h)_h$ cannot be guaranteed *a priori*.

This serves as motivation to examine mesh quality with special attention paid to the term ∇R. Note that we encountered the terms $\|\nabla R\|_2$ and $\det(\nabla R)$ in Lemma 2.2 (where $\nabla R = B \in \mathbb{R}^{d \times d}$ due to R being affine). As we will see later (Section 3.5.1 and Remark 3.3), these terms have interpretations with regard to the scaling of different geometric quantities. I chose the PDE-based approach, where mesh quality is defined by a functional with an underlying partial differential equation. A minimiser of such a mesh quality functional is a solution of this PDE in an appropriate sense, meaning that results about existence and (non-)uniqueness of solutions can be reused. Also, since all of this is done to obtain a mesh to numerically solve another PDE on, the structures and tools are already available.

The chapter is organised as follows:

The terms $\|\nabla R\|_2$ and $\det(\nabla R)$ already suggest to look into the field of elasticity, so various related terms and definitions that will be useful are introduced in Section 3.1. In Section 3.2, I will give examples of use cases for mesh optimisation, as well as of various ideas of "mesh quality" in Section 3.3 for the cases $d = 1$ and $d = 2$. In a somewhat inverse approach, I will discuss the use of different PDEs to define mesh quality functionals in Section 3.4 for the computation of an extension operator for a problem with moving outer boundary. All these sections are supposed to give some insight on the criteria for designing (or rather choosing) a suitable class of PDEs for use as the basis of mesh quality functionals.

The main part of this chapter is Section 3.5. In this section, a family of nonlinear mesh quality functionals is derived based on mechanical principles and some assumptions on the properties we would expect a mesh quality functional to have (Section 3.5.1 and Section 3.5.2). It turns out that this class of functionals can be interpreted as stored energy functionals for hyperelastic materials (Section 3.5.5) and that the choice of reference cells and scales (Section 3.5.3 and Section 3.5.4) was nothing more than constructing an inhomogeneous material response function. In the field of elasticity, hyperelastic materials are well-studied, so analysis is available and presented in Section 3.5.6. Section 3.5.7 deals with *r*-adaptivity and Section 3.5.8 with the alignment of the mesh to surfaces, which can be used to further adapt the mesh to the requirements of the original PDE.

3.1. Deformations, variations and mesh quality functionals

The term "variation" is taken from [Rum96].

Definition 3.1. *The spaces of admissible deformations*
Let $\Omega \subset \mathbb{R}^d, d = 2,3$ be a bounded domain with

$$\partial\Omega \in \mathcal{C}^{0,1}, \Gamma_0, \Gamma_1, \Gamma_2 \subset \partial\Omega, \mathrm{vol}_{d-1}\left(\partial\Omega \setminus (\Gamma_0 \cup \Gamma_1 \cup \Gamma_2)\right) = 0.$$

i) Then

$$\tilde{\mathcal{D}}_{op} := \left\{\Phi : \Omega \to \mathbb{R}^d : \forall x \in \Omega : \nabla\Phi(x) \in \mathrm{SL}_d, \right\} \tag{3.1}$$

is called the space of orientation preserving deformations.

ii) If Ω_h is a polygonal approximation of Ω and $\mathcal{T}_h$ a regular, conforming mesh on Ω_h, define the discrete space of orientation preserving deformations *as*

$$\tilde{\mathcal{D}}_{op,h} := \tilde{\mathcal{D}}_{op,h}(\mathcal{T}_h) := \tilde{\mathcal{D}} \cap V_h, V_h = \begin{cases} \mathbb{P}_1(\mathcal{T}_h), & \hat{K} = \hat{S} \\ \mathbb{Q}_1(\mathcal{T}_h), & \hat{K} = \hat{Q} \end{cases}. \tag{3.2}$$

iii) Since we are mainly interested in deformations which preserve the boundary $\partial\Omega$, we also define the subspaces of boundary preserving deformations

$$\tilde{\mathcal{D}}_{bp} := \{\Phi \in \mathcal{D}_{op} : \forall x \in \partial\Omega : \Phi(x) \in \partial\Phi(\Omega)\}$$
$$\tilde{\mathcal{D}}_{bp,h} := \tilde{\mathcal{D}}_{bp,h}(\mathcal{T}_h) := \{\Phi \in \mathcal{D}_{op,h} : \forall x \in \partial\Omega_h : \Phi_h(x) \in \partial\Phi_h(\Omega_h)\}$$

To these spaces, a variety of boundary conditions can be applied. Likewise, constraints can be imposed. Examples are:

iv) Displacement boundary condition: If $\phi_0 : \Gamma_0 \to \mathbb{R}^d$ is sufficiently smooth, require that

$$\forall x \in \Gamma_0\ d\sigma\ a.e. : \Phi(x) = \phi_0(x).$$

v) Traction boundary condition: If $\hat{T} : \bar{\Omega} \times \mathrm{SL}_d \to \mathbb{R}^{d\times d}$ *is a response function for the first Piola-Kirchhoff stress and* ν *is the outer unit normal field, then a traction boundary condition is*

$$\forall x \in \Gamma_1 \; d\sigma \; a.e. : \hat{T}(x, \nabla\Phi(x))n(x) = \hat{g}(x, \nabla\Phi(x))n(x).$$

vi) Unilateral boundary condition of place: For a given $\Gamma \subset \mathbb{R}^d$ *require that*

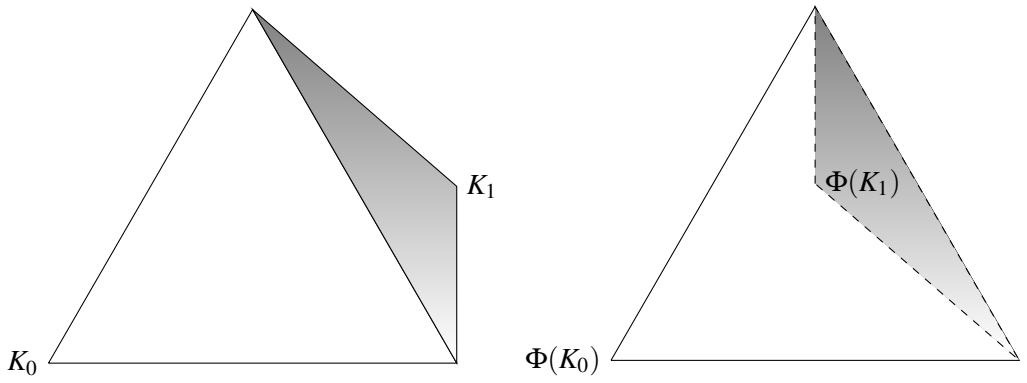
$$\forall x \in \Gamma_2 \; d\sigma \; a.e. : \Phi(x) \in \Gamma$$

vii) Injectivity constraint:

$$\int_\Omega \det(\nabla(\Phi)(x)) \; dx \leq \mathrm{vol}(\Omega).$$

viii) Locking constraint: See Remark 3.14.

Note that the condition $\forall x \in \Omega : \nabla\Phi(x) \in \mathrm{SL}_d$ implies $\det(\nabla\Phi(x)) > 0$ and this ensures that Φ is one to one *locally*, but not *globally*. For a discrete deformation Φ_h which is e.g. piecewise linear for simplical meshes and $\det(\nabla\Phi_{h|K})$ is constant $\forall K \in \mathcal{E}^d(\mathcal{T})$, one could wonder what happened if $\det(\nabla\Phi_{h|K_1})$ changed sign (see Figure 3.1). Because of the continuity, this would lead to cell K_1 having an overlap with an adjacent cell K_0 where $\det(\nabla\Phi_{h|K_0}) > 0$ (this is sometimes called *mesh tangling*). Because of the boundary conditions, it is in general not possible for a deformation to change the orientation of *all* cells, so we do not treat this case specifically.



(i) Initial configuration. (ii) After applying the mapping Φ.

Figure 3.1: Overlap of cells resulting from one cell changing orientation.

Remark 3.1. *Note that the regularity was left unspecified in some of the definitions. Just formally, one can assume* $\Phi \in \left(\mathcal{C}^1(\Omega)\right)^d$ *at first. Later, we will see that the analysis requires*

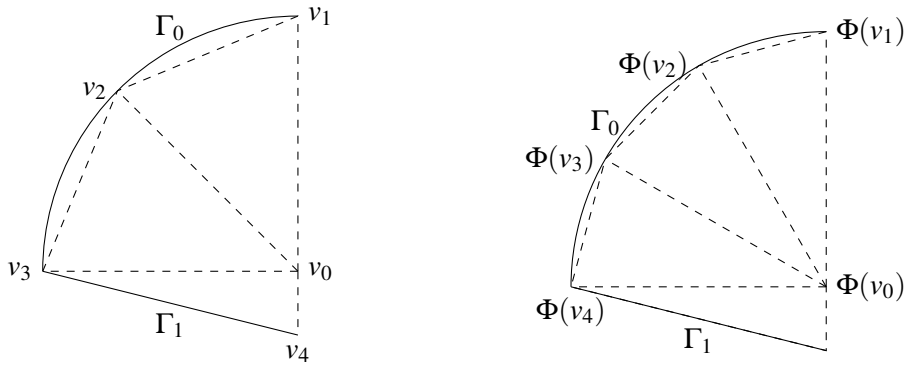
$$\Phi \in W^{1,p}(\Omega), \mathrm{Cof}(\nabla\Phi) \in L^q(\Omega), \det(\nabla\Phi) \in L^r(\Omega)$$

for some constants p, q, r *(Theorem 3.11).*

If $\partial\Omega$ is a union of smooth surfaces segments, smooth curves and singular points, the definition of deformations does not prevent vertices of a mesh $\mathcal{T}_h$ on a polygonal approximation Ω_h to Ω from switching from one part to another as depicted in Figure 3.2. As it is mentioned in [Rum96], treating this is a combinatorial task and does not fit well into a variational setting. So in the following, we will work with *variations* instead of deformations, which is a significant restriction.

Definition 3.2. *The space of admissible variations*

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$ *and* $\partial\Omega = \biguplus_{m=0}^{d-1} \partial\Omega^m$, *where* $\partial\Omega^0$ *is a set of singular points and* $\partial\Omega^m$ *are sets of relatively open, smooth m-dimensional manifolds.*



(i) Initial state.

(ii) After applying the deformation Φ.

Figure 3.2: The deformation Φ lets the singular vertex v_3 switch to Γ_0.

i) Then

$$\mathcal{D} := \{\Phi \in \tilde{\mathcal{D}}_{bp} : \forall m = 0,\dots,d-1 : \forall \Gamma \in \partial\Omega^m : \forall x \in \Gamma : \Phi(x) \in \Gamma\} \tag{3.3}$$

is called the space of variations.

ii) If Ω_h is a polygonal approximation of Ω, $\mathcal{T}_h$ a regular, conforming mesh on Ω_h and $\forall m = 0,\dots,d : \partial\Omega_h^m$ are the sets of polygonal approximations to $\partial\Omega^m$ respectively, define the space of discrete variations *as*

$$\mathcal{D}_h := \mathcal{D}_h(\mathcal{T}_h) := \{\Phi \in \tilde{\mathcal{D}}_{bp,h} : \forall m = 0,\dots,d-1 : \forall \Gamma \in \partial\Omega_h^m : \forall x \in \Gamma : \Phi(x) \in \Gamma\}, \tag{3.4}$$

Now we briefly state the definitions of some important quantities from mathematical elasticity in a less than formal manner to establish the vocabulary. The details are beyond the scope of this work and the reader is referred to [Cia88, Chapter 2] for a complete presentation.

Definition 3.3.

i) The right Cauchy-Green strain tensor *is denoted by*

$$C : \bar{\Omega} \to \mathbb{R}^{d\times d}_{sym}, C(x) := (\nabla\Phi(x))^T \nabla\Phi(x) \tag{3.5}$$

ii) The Cauchy stress tensor *of a deformation Φ is denoted by*

$$T^\Phi : \Phi(\Omega) \to \mathbb{R}^{d\times d}_{sym}. \tag{3.6}$$

iii) The first Piola-Kirchhoff stress tensor *is defined by*

$$T : \bar{\Omega} \to \mathbb{R}^{d\times d}, \quad T(x) := \det(\nabla\Phi(x)) T^\Phi(\Phi(x))(\nabla\Phi(x))^{-T}. \tag{3.7}$$

iv) A material is called elastic *iff*

$$\exists \hat{T} : \bar{\Omega} \times \mathrm{SL}_d \to \mathbb{R}^{d\times d} : \quad \forall x \in \bar{\Omega} : T(x) = \hat{T}(x, \nabla\Phi(x)) \tag{3.8}$$

and $\hat{T}$ is called the material's response function.

v) A material is called hyperelastic *iff*

$$\exists \mathcal{L} : \bar{\Omega} \times \mathrm{SL}_d \to \mathbb{R} : \quad \forall x \in \bar{\Omega} : \hat{T}(x, \nabla\Phi(x)) = \frac{\partial \mathcal{L}}{\partial A}(x, \nabla\Phi(x)) \tag{3.9}$$

and $\mathcal{L}$ is called the material's stored energy function.

3.2. Three important aspects for mesh optimisation

Let $\Omega \subset \mathbb{R}^d, d = 1,2,3$ be the region of interest, Ω_h its polygonally bounded approximation and $\mathcal{T}_h$ be a regular, conforming discretisation of Ω_h into d-simplices or d-hypercubes. We are looking for deformations $\Phi : \Omega_h \to \mathbb{R}^d$ and the optimal deformation Φ^* that minimises a functional $\mathcal{F}$ over a set V of admissible deformations.

I want to give three important use-cases for this setting.

3.2.1. Computation of inverse trace operators

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

$$\text{For } t \in [0,\bar{t}] \text{ find } \Phi^* = \operatorname{argmin}_{\Phi \in V} \mathcal{F}(\Phi), \quad V = \{\Phi \in \mathcal{D}_h : \forall x \in \partial\Omega_0 : \Phi(x) = \operatorname{tr}\varphi(t,x)\}.$$

In general, $\operatorname{tr}\varphi$ is unknown and part of the solution to the original PDE (like the position of the free capillary boundary (see e.g. [Bän98]), the evolution of the phase boundary (e.g. [BPS13]) etc.). The difficult part is to find such a functional $\mathcal{F}$ such that its minimisers define meshes that are well suited for the discretisation of the original PDE.

3.2.2. *r*-adaptivity

Many variational problems give rise to error estimates of the form

$$\|u - u_h\|_{1,\Omega} \leq Ch^k |u|_{k+1,\Omega}$$

which is an error estimate [Cia78, Theorem 3.2.2] for the solution u and its finite element approximation u_h of local polynomial degree k to a second order elliptic problem.

To reduce the error without globally refining the mesh $\mathcal{T}_h$, there are three common strategies:

i) h-adaptivity: Locally refine cells to reduce h in regions with high contributions to the global error, creating more DoF. Especially in $3d$, devising a local refinement algorithm that guarantees lower bounds on geometric quantities like the aspect ratio of the cells and can be proven to terminate for arbitrary meshes is a challenge [Bän91].

ii) p-adaptivity: Locally use a different FE basis (e.g. of higher polynomial degree k). It is possible to combine this with h adaptivity. $h-p$-adaptivity is a very powerful concept and quite complex, both from a mathematical and an implementation point of view. [SSD03, Chapter 6]

iii) r-adaptivity: Move mesh vertices according to a mesh density function to reduce h locally in regions with high contributions to the global error, increasing h in regions with low contributions to the global error. [HR11, Chapter 6]

r-adaptivity has the advantage that it does not increase the number of DoF or modify the adjacency structure of the underlying FE spaces, which is advantageous from a solver perspective. On the other hand, the local cell size *h* cannot be varied as quickly as in *h*-adaptivity, which is due to the needed regularity of the mesh density (or monitor) function. Also, nonuniform meshes tend to reduce the effectiveness of some linear solvers. *r*-adaptivity will be further examined in section Section 3.5.7.

3.2.3. Mesh alignment to (implicit) surfaces

Interior boundaries like a phase boundary in two phase flow or the liquid-solid boundary in fluid-structure interactions can be represented in different ways, including:

- Implicit (e.g. levelset or phase field based representation), which allows changes to the topology and the use of a fixed reference mesh.

- Sharp interface, which makes devising an arbitrary Lagrangian-Eulerian (ALE) formulation for a moving domain very easy. The drawbacks are that changes to the surface topology are ruled out by necessary assumptions, and the need for very robust mesh deformation methods.

One other aspect for choosing one method over the other are the error estimates, which are in favour of the sharp interface approach in general. Without going into the details, I will just state results proven in [LMWZ10].

If $\Omega = \Omega_1 \cup \Omega_2 \subset \mathbb{R}^d, d = 2,3$ is bounded with $\partial\Omega, \partial\Omega_1, \partial\Omega_2 \in \mathcal{C}^{0,1}$, $\Omega_1 \subset\subset \Omega, \Omega_2 = \Omega \setminus \Omega_1$ and $\Gamma = \partial\Omega_1 \in \mathcal{C}^2$, consider the elliptic interface problem

$$\begin{aligned} \nabla \cdot (\beta \nabla u) &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega, \\ [u]_\Gamma := (u_{1|\Gamma} - u_{2|\Gamma}) &= 0, \\ [\beta \partial_\nu u]_\Gamma := (\beta_1 \partial_{\nu_1} u_1 + \beta_2 \partial_{\nu_2} u_2) &= 0, \end{aligned}$$

where $u_i := u_{|\Omega_i}$ and $\beta_i = \beta_{|\Omega_i}$. This leads to the following continuous variational problem:

$$\begin{aligned} &\text{Find } u \in H^1(\Omega) \text{ such that} \\ &\qquad a(u,v) = L(v) \; \forall v \in H_0^1(\Omega), \text{ where} \\ &a(u,v) := \sum_{i=1}^{2} \int_{\Omega_i} \beta_i \nabla u \cdot \nabla v dx, \; L(v) := \int_\Omega f v dx \end{aligned} \tag{3.10}$$

For a given mesh $\mathcal{T}_h$ on Ω such that every cell $K \in \mathcal{T}_h$ is parametrically (affine or of higher order) equivalent to $\hat{S}$, define the δ-tube

$$S_\delta(\Gamma) := \{x \in \Omega : \text{dist}(x,\Gamma) < \delta\}$$

and with that the decompositions of $\mathcal{T}_h$

$$\begin{aligned} \mathcal{T}_h^i &:= \{K \in \mathcal{T}_h : K \cap S_\delta(\Gamma) = \emptyset\}, \\ \mathcal{T}_* &:= \mathcal{T}_h \setminus (\mathcal{T}_h^1 \cup \mathcal{T}_h^2), \\ \mathcal{T}_*^i &:= \{K \in \mathcal{T}_h : K \subset (\Omega_i \cup \mathcal{T}_*)\}. \end{aligned}$$

Assume now that $\delta \leq \frac{h}{2}$ and Γ is δ-resolved, meaning that $\mathcal{T}_*^1 \cap \mathcal{T}_*^2 = \emptyset$ and $\mathcal{T}_* = \mathcal{T}_*^1 \cup \mathcal{T}_*^2$ and define

$$\bar{\Omega}_{h,i} := \{ \bigcup_{K \in \mathcal{T}_h^i \cup \mathcal{T}_*^i} \bar{K} \},$$
$$\Gamma_h := \partial \Omega_{h,1}.$$

The parameter δ describes how well the interface Γ is approximated. $\delta \leq \frac{h}{2}$ already means that $\nexists \mathcal{T}_h \ni K \subset \mathcal{T}_*^1 \cap \mathcal{T}_*^2$.

With this we get the following discrete variational problem:

$$\begin{aligned} &\text{Find } u_h \in S^p(\mathcal{T}_h) := \{v \in H^1(\Omega) : v \circ R_K \in \mathbb{P}_p(\hat{S})\} \text{ such that} \\ &\quad a_h(u,v) = L(v) \; \forall v \in S_0^p(\mathcal{T}_h), \text{ where} \\ &\quad a_h(u,v) := \sum_{i=1}^{2} \int_{\Omega_{h,i}} \beta_i \nabla u \cdot \nabla v dx \end{aligned} \tag{3.11}$$

Then [LMWZ10, Theorem 4.1] states that, under some assumption on the interpolation operator I_P, the solutions u, u_h to the variational problems (3.10), (3.11) with $u \in H^1(\Omega) \cap H^s(\Omega_1 \cup \Omega_2)$ for some $s \in [1, p+1]$ satisfy the estimates

$$\forall s \in \left[1, \frac{3}{2}\right) : \|u - u_h\|_{1,\Omega} \leq C h^{s-1} \|u\|_{s,\Omega_1 \cup \Omega_2}, \tag{3.12}$$

$$\begin{aligned} \forall s \in \left[\frac{3}{2}, p+1\right] \wedge \nabla u \in B^{\frac{1}{2}}_{2,1}(\Omega_1 \cup \Omega_2) \\ \Rightarrow \|u - u_h\|_{1,\Omega} \leq C h^{s-1} \|u\|_{s,\Omega_1 \cup \Omega_2} + \sqrt{\delta} \|\nabla u\|_{0, B^{\frac{1}{2}}_{2,1}(\Omega_1 \cup \Omega_2)}, \end{aligned} \tag{3.13}$$

with the Besov space $B^{\frac{1}{2}}_{2,1}(\Omega_1 \cup \Omega_2) := \left(L^2(\Omega_1 \cup \Omega_2), H^1(\Omega_1 \cup \Omega_2)\right)_{\frac{1}{2},1}$.

Typical values for δ are

$$\delta = O(h), \text{ (levelset representation of } \Gamma),$$
$$\delta = O(h^{m+1}), \text{ (sharp interface representation with } R_K \in \mathbb{P}_m(\hat{S})).$$

This leads to [LMWZ10, Theorem 4.12]: If Γ is sufficiently smooth and $R_K \in \mathbb{P}_m(\hat{S})$ with u, u_h as above and $u \in H^{p+1}(\Omega)$, then

$$m = p : \|u - u_h\|_{1,\Omega} \leq C h^{\frac{p+1}{2}} \|u\|_{p+1,\Omega_1 \cup \Omega_2}, \tag{3.14}$$
$$m \geq 2p - 1 : \|u - u_h\|_{1,\Omega} \leq C h^p \|u\|_{p+1,\Omega_1 \cup \Omega_2}. \tag{3.15}$$

If additionally, for $f \in L^2(\Omega) : \exists \tau > \frac{3}{2}$ and $u \in H_0^1(\Omega) \cap H^\tau(\Omega_1 \cup \Omega_2)$ that satisfy the a priori estimate

$$\|u\|_{\tau,\Omega_1 \cup \Omega_2} \leq C \|f\|_{0,\Omega},$$

then in both cases the additional L^2 estimate

$$\|u - u_h\|_{0,\Omega} \leq C h^{p+\tau-1} \|u\|_{p+1,\Omega_1 \cup \Omega_2}$$

holds. This means that for a sharp interface representation, $\mathbb{P}_1(\mathcal{T}_h)(m = p = 1)$ offers the optimal order of convergence in both L^2- and H^1-norms, where isoparametric elements suffer an order reduction in the H^1-norm, while still having the optimal order of convergence in the L^2-norm. To recover the optimal order in the H^1-norm, one has to resort to superparametric elements (see Definition 2.11).

Using an implicit representation of Γ still means $\delta = O(h)$ so the order of convergence in the H^1-norm stays limited to $\sqrt{h}$ independent of m for $s \geq \frac{3}{2}$ and is optimal only for $s = \frac{3}{2}$.

This means the price paid for the ability to handle topology changes is a harsh limit on the order of convergence. However, in [BW13] an implicit representation of the surface is used and the mesh is then aligned with the surface, recovering a sharp interface representation. This will be elaborated in Section 3.5.8.

3.3. Examples for mesh quality in $1d$ and $2d$

3.3.1. An example for mesh quality in $1d$

Mesh optimisation is mostly trivial in $1d$, but some important aspects can be discussed in such a simple setting without technical and geometric difficulties distracting from the main ideas.

Let $\Omega = [0,1] \subset \mathbb{R}$ be given and define the vertices $a_0 = 0, a_1 = \frac{1}{3}, a_2 = 1$. We can define a mesh $\mathcal{T}$ on Ω consisting of the cells $K_0 = (a_0, a_1), K_1 = (a_1, a_2)$, so we have the geometric entities $\mathcal{E}^0(\mathcal{T}) = \{a_0, a_1, a_2\}$ and $\mathcal{E}^1(\mathcal{T}) = \{K_0, K_1\}$, with $\Omega = \biguplus_{r=0}^{1} \mathcal{E}^r(\mathcal{T})$.

(i) Ω

(ii) Ω (in gray) and $\tilde{\Omega}$ (in black).

Figure 3.3: The domain of interest.

Assume now that the boundary vertices a_0, a_2 are moved to the positions $b_0 = a_0 + \frac{3}{10}$ and $b_2 = a_2 + \frac{1}{3}$ and we want to find the new position b_1 of the vertex a_1. The cell $\tilde{K}_0 = (b_0, a_1)$ is very small as compared to the cell $\tilde{K}_1 = (a_1, b_2)$ at the moment, with $\mathrm{vol}(\tilde{K}_0) = \frac{3}{100}, \mathrm{vol}(\tilde{K}_1) = \frac{97}{100}$.

Let us have a look at different strategies for finding b_1, or rather, finding a function $\Phi : \Omega \to \tilde{\Omega}$, $\Phi \in \mathcal{C}^0(\Omega)$, with the properties $\Phi(a_0) = b_0, \phi(a_2) = b_2$. Ideas are:

i) b_1 should be in the centre of gravity of all adjacent vertices, so set $b_1 = \frac{1}{2}(b_2 - b_0) = \frac{8}{10}$. Now we have $\mathrm{vol}(\tilde{K}_0) = \mathrm{vol}(\tilde{K}_1) = \frac{1}{2}$, meaning we changed the relative sizes of the cells. This just adjacenty graph based Laplacian smoothing. It can be generalised to other situations, including weighting, and stays as simple as in the 1d case as (apart e.g. from the weighting function), it naturally decouples the components.

ii) Conserve the relative cell sizes by setting $\frac{\tilde{K}_0}{K_0} = \frac{\tilde{K}_1}{K_1}$. This is very useful for maintaining a given cell size distribution e.g. coming from local refinement. This condition gives us

$$\begin{aligned} \frac{b_1 - b_0}{a_1 - a_0} &= \frac{b_2 - b_1}{a_2 - a_1} \\ \Leftrightarrow\ b_1 &= \frac{(a_1 - a_0)b_2 + (a_2 - a_1)b_0}{(a_1 - a_0) + (a_2 - a_1)} \\ &= \frac{19}{30}. \end{aligned}$$

This condition is harder to generalise to higher space dimensions because of different adjacency structures.

iii) Let $V := H^1(\Omega)$, $V_0 := H_0^1(\Omega)$, $V_D := \{v \in V : v(a_0) = b_0, v(a_2) = b_2\}$ and consider the bilinear form

$$a(u,v) = \int_\Omega u'v'dx. \tag{3.16}$$

We are looking for a deformation $\Phi \in V_D$ such that $\Phi = \operatorname{argmin}_{v \in V_D} a(v,v)$. Exploiting finite element theory and variational calculus, one can show that this is equivalent to $\forall v \in V_0 : a(\Phi, v) = 0$.

Now we use $([0,1], \mathbb{P}_1([0,1]), \hat{\Sigma})$ with $\hat{\Sigma} = \{\phi_0, \phi_1\}$, $\phi_0(v) = v(a_0), \phi_1(v) = v(a_2)$ as our reference finite element and use that to discretise the minimisation problem (3.16). Recall the standard $\mathbb{P}_1$ transformation F_0 for the cell K_0:

$$F_0(\hat{x}) = a_0(1-\hat{x}) + a_1\hat{x}, \quad F_0'(\hat{x}) = (a_1 - a_0).$$

Recalling that

$$v(x) = \hat{v}(F_0)^{-1}(x), \quad \frac{dv}{dx}(x) = \frac{d\hat{v}}{d\hat{x}}(F_0^{-1}(x))\frac{dF_0^{-1}}{dx}(x)$$

and using the transformation theorem, we can rewrite the integration as

$$\begin{aligned}\int_{F_0(\hat{K})} u'v'dx &= \int_{\hat{K}} \left(\hat{u}'(\hat{x})(F_0^{-1})'(F_0(\hat{x}))\right) \left(\hat{v}'(\hat{x})(F_0^{-1})'(F(\hat{x}))\right) \det(F_0'(\hat{x}))d\hat{x} \\ &= (a_1 - a_0)^{-1} \int_{\hat{K}} \hat{u}'\hat{v}'d\hat{x}\end{aligned}$$

Now, evaluating this integral for the finite element basis functions ϕ_0, ϕ_1, ϕ_2 for the cells K_0, K_1 and writing our solution as $\Phi = \sum_{i=0}^2 b_i\phi_i$ gives rise to the linear system of equations

$$\begin{pmatrix} (a_1-a_0)^{-1} & -(a_2-a_1) & 0 \\ -(a_1-a_0)^{-1} & (a_1-a_0)^{-1} + (a_2-a_1)^{-1} & -(a_2-a_1) \\ 0 & -(a_2-a_1)^{-1} & (a_2-a_1) \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Because of the boundary values, the first and the last equations are eliminated, leaving only the second equation, which gives

$$b_1 = \frac{(a_1-a_0)b_2 + (a_2-a_1)b_0}{(a_1-a_0)+(a_2-a_1)} = \frac{19}{30}$$

as in option ii. This does not generalise for $d > 1$, however.

Because we are in $1d$, a cell only has one quantity describing its quality: Its size, because that uniquely defines $\det(\nabla R_K)$. To define the quality of a mesh, one just has to define an optimal size for each cell.

In higher space dimensions, more quantities are involved, which gives more freedom to define quality measures for meshes, as will be shown in the following example.

3.3.2. Examples of mesh quality in $2d$

Let $\Omega = [0,1]^2$ and $\mathcal{T}$ made up of four cells $K_0, \ldots, K_3$ as depicted in Figure 3.4. We want to fix all vertices on the boundary $\partial\Omega$, so the only vertex we are allowed to move is vertex $v = \left(\frac{1}{2}, \frac{1}{3}\right)$.

We now want to optimise the quality of the mesh $\mathcal{T}$ by moving vertex v. Without formally defining the mesh quality functionals, we want to give some examples of criteria. Some of them are defined directly for the mesh $\mathcal{T}$, some of them are given in more general form.

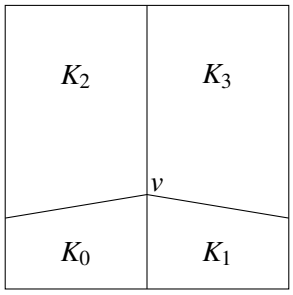


Figure 3.4: Ω with mesh $\mathcal{T}$.

i) Minimise the distance of each vertex to the centre of gravity of all vertices it shares an edge with. More formally:

$$\begin{aligned}
&\forall v \in \mathcal{E}^0(\mathcal{T}) : \mathcal{V}^k(v) := \{w \in \mathcal{E}^0 : \exists E \in \mathcal{E}^k(\mathcal{T}) : v, w \in \bar{E}\},\\
&f(v) := v - \frac{1}{|\mathcal{V}^1(v)|} \sum_{w \in \mathcal{V}^1(v)} w,\\
&\mathcal{F}(\mathcal{T}) := \sum_{v \in \mathcal{E}^0(\mathcal{T})} f(v).
\end{aligned}$$

ii) Uniformly distribute the volume of $\mathcal{T}$ over all cells, meaning

$$\forall K_i, K_j \in \mathcal{E}^2(\mathcal{T}) : \mathrm{vol}(K_i) = \mathrm{vol}(K_j).$$

iii) Maximise the minimum angle between two edges.

iv) Distribute the cell volume in a specific way, e.g. such that

$$\mathrm{vol}(K_{0,1}) = \frac{\mathrm{vol}(K_2)}{7} = \frac{\mathrm{vol}(K_3)}{7}.$$

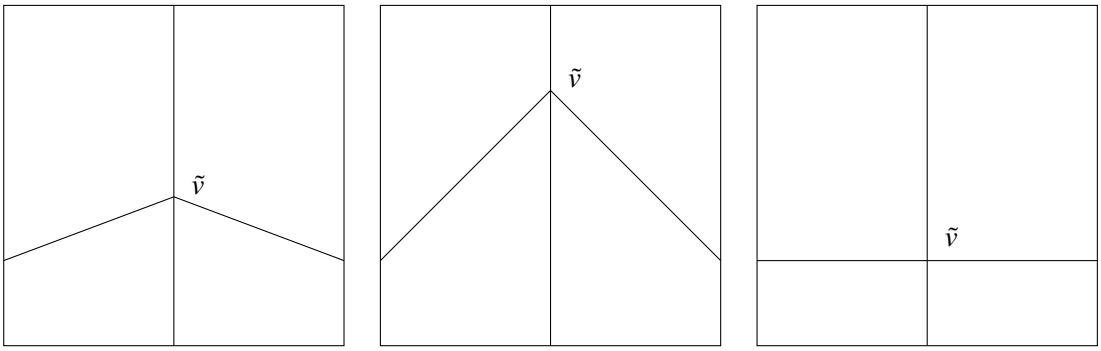
v) Distribute the cell volume such that

$$\mathrm{vol}(K_0) = \frac{\mathrm{vol}(K_1)}{2}.$$

Note that this prescribes no conditions for $K_{2,3}$.

The notions of mesh quality (i), (ii) and (iii) are very general and (in this case) have a unique minimiser leading to regular meshes. The question is if this is still the case for $3d$ meshes, if they can be used to construct well-defined mesh quality functionals and how to minimise those functionals.

Even for $3d$ and arbitrary complex meshes, the functional from (i) can easily be assembled into a linear system of equations which is quite easy to solve because it is weakly diagonally dominant as per construction. In $1d$, (ii) can be treated with de Boor's algorithm (see [HR11, Chapter 2]), which does not generalise to $2d$ and $3d$. For the higher space dimensions, the formulation of the equidistribution condition is more involved, as is assembling and solving the resulting system of

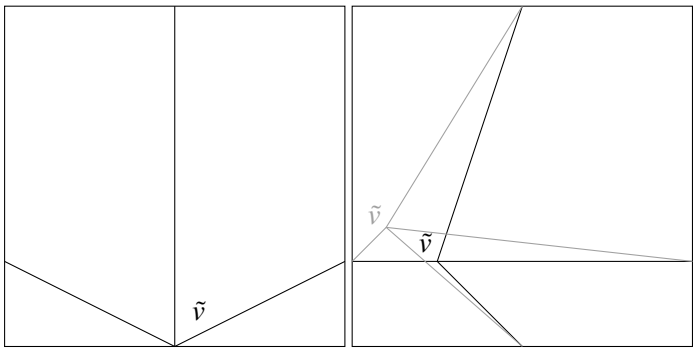


(i) Centre of gravity of the surrounding vertices.

(ii) Uniform distribution of the volume.

(iii) Maximising the minimum angle.

Figure 3.5: Well-defined notions mesh quality.



(i) $\mathrm{vol}(K_{0,1}) = \frac{\mathrm{vol}(K_2)}{7} = \frac{\mathrm{vol}(K_3)}{7}$ (non-regular solution).

(ii) $\mathrm{vol}(K_0) = \frac{\mathrm{vol}(K_1)}{2}$ (no unique solution).

Figure 3.6: Notions mesh quality leading to non-regular meshes or nonunique solutions.

equations [HR11, Chapter 4]. (iii) can be used to create a highly nonlinear mesh quality functional, which is not very useful for practical computations.

To show that not every notion of mesh quality (or rather set of conditions for various quantities) has good minimisers, examples (iv) and (v) are given. In (iv), the target cell sizes can be chosen to fulfil the necessary condition $\sum_{K\in\mathcal{T}} \mathrm{vol}(K) = \mathrm{vol}(\Omega)$, but since the vertex coordinates on $\partial\Omega$ are fixed, the only possible solution is to degenerate the quadrilaterals to triangles, which means that the transformation is not injective on those cells. As can be expected, if too few conditions are specified, the solution might not be unique as it is in example (v). The additional difficulty in this case is that the solutions are not isolated. Instead, there is a whole connected set of which each member is a vertex such that the volume distribution condition is fulfilled.

These examples illustrate that it it easy formulate what an "optimal" mesh should be, but not all of these not all of these notions give rise to unique minimisers, or even minimisers in the space of orientation preserving deformations. Moreover, finding a numerical method to actually computing these minimisers might prove difficult.

3.4. Computation of extension operators in $2d$

In the examples before, we started with an idea what we would like the mesh quality measure to be and then sought to define functionals representing this quality measure. In this section, we take the different approach of using (more or less) well-known functionals and then examining the quality

of the meshes given by their minimisers.

Set $\Omega_0 := [0,1]^2$, $0 = t_0 < \cdots < t_{50} = \bar{t} = 0.5$ and let

$$\forall x \in \partial\Omega_0 : \varphi_\Gamma(t,x) = \begin{cases} (x_1, x_2 + \frac{1}{2} t x_2 \sin(2\pi x_1))^T, & x_2 = 1 \\ (x_1, x_2)^T, & \text{else} \end{cases}$$

For a given discretisation $\mathcal{T}_h$ of a reference domain $\hat{\Omega}_t$ (assume for now $\hat{\Omega}_t = \Omega_0$) define

$$V(t) = \left\{ v \in \mathbb{P}_1(\hat{\Omega}_t) : v_{|\partial\hat{\Omega}_t} = \varphi_\Gamma(t) \right\}, W(t) = \left\{ w \in \mathbb{Q}_1(\hat{\Omega}_t) : w_{|\partial\hat{\Omega}_t} = 0 \right\}.$$

$\forall k = 1, \ldots, N$: Compute $\Omega(t_k) = \varphi(t_k, \hat{\Omega}_t)$ by finding $\Phi_h \in V(t)$:

$$\forall \Psi_h \in W(t_k) : \mathcal{F}'(\Phi_h)\Psi_h = 0 \text{ and setting } \Omega(t_k) = \Phi(\hat{\Omega}_t).$$

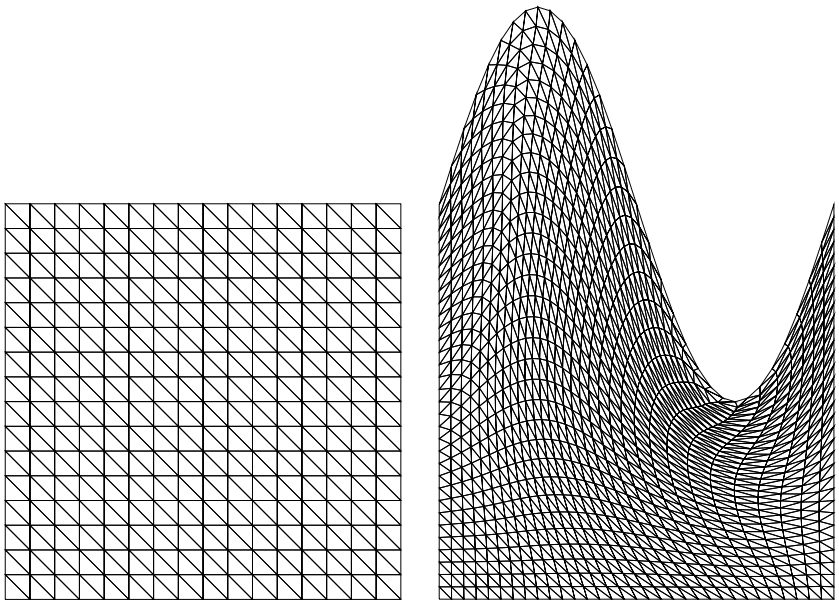


Figure 3.7: Left: Ω_0, right: $\Omega(t_{50})$ with interior mesh computed by a hyperelasticity based mesh optimiser.

I will now present three different mesh quality functionals. All mesh quality functionals were implemented using FEniCS 1.6.0 [ABH+15].

3.4.1. Minimisation of harmonic energy

Define the well-known bilinear form

$$\forall (u,v) \in V(t) \times W(t) : \quad a_{h,t}(u,v) = \int_{\hat{\Omega}_t} \nabla u : \nabla v dx \tag{3.17}$$

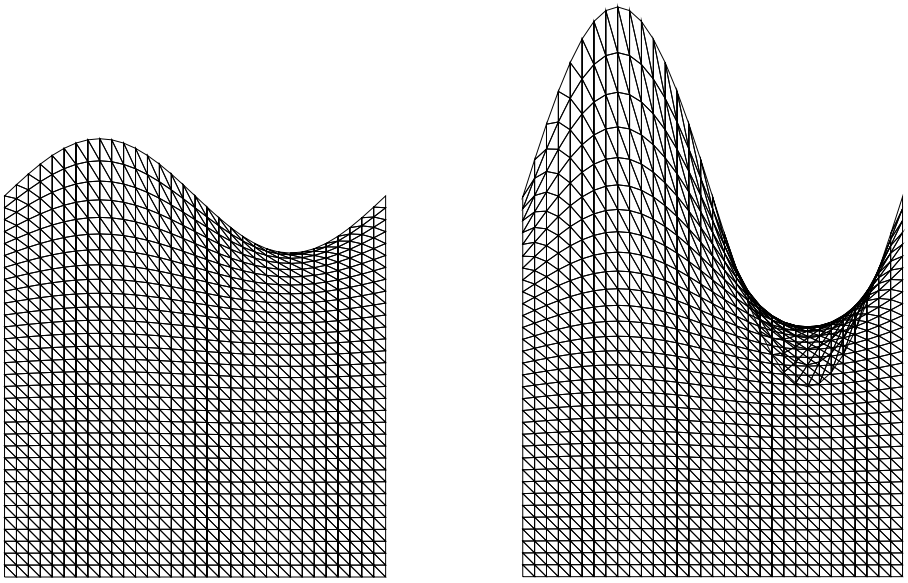
and with this the mesh quality functional

$$\mathcal{F}(u) := \frac{1}{2} a_{h,t}(u,u). \tag{3.18}$$

Because of the boundedness, coerciveness and symmetry of a_h we get the existence and uniqueness of a solution u_h of (3.18) and that

$$u_h = \operatorname{argmin}_{u \in V(t)} \mathcal{F}(u) \Leftrightarrow \forall w \in W(t) : \ \frac{\partial \mathcal{F}}{\partial w} = 0. \tag{3.19}$$

The question is now if the discrete solution u_h is a variation by Definition 3.2. The simplest method possible sets $\hat{\Omega}_t = \hat{\Omega}_0$, meaning that we have a fixed reference domain on which we solve (3.19). The underlying linear system of equations does not change over time, making the method rather inexpensive computationally.



(i) $t_{15} = 0.15$ (ii) $t_{50} = 0.50$

Figure 3.8: The mesh as given by the solution of (3.19) with $\hat{\Omega}_{t_k} = \hat{\Omega}_0$.

As it can be seen in Figure 3.8, so called *mesh tangling* (see [HR11, Chapter 1.6]) occurs: The condition $\forall K_0, K_1 \in \mathcal{E}^2(\mathcal{T}_h), K_0 \neq K_1 : K_0 \cap K_1 = \emptyset$ is violated. It can be seen that some cells changed orientation (meaning $\det(\nabla R_K) < 0$) so this overlap occurs. From this we can already deduce that the discrete solution is not a discrete variation in the sense of Definition 3.2. The root cause of this is the well-known maximum principle for harmonic functions [Eva98, Chapter 2.2], which carries over to the discrete form in (3.17). In the following, I omit the index h on the discrete solution.

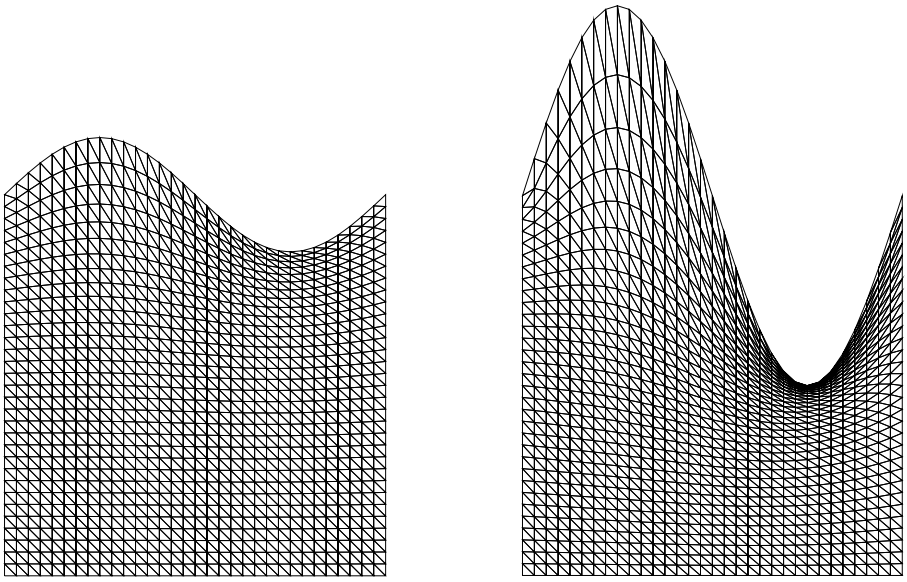
The deformations $u_{t_k}, u_0 = id$ are discrete-harmonic and so is $u_{t_k} - u_0$, which has the boundary values $\phi_\Gamma(t_k, x) - \Phi_\Gamma(0, x)$. As the components are decoupled, it suffices to regard the x_2-component, or rather an edge e aligned with the x_2-axis. Assume that $\bar{e}$ is defined by the vertices $\hat{z}_0 \in \Gamma(0), \hat{z}_1 \in \mathring{\hat{\Omega}}_0$. Because of the discrete maximum principle, and the fact that we can arbitrarily increase $|u_{t_l}(z_0) - z_0|$, there is a time instant t_l for which

$$|u_{t_l}(z_0) - z_0| > |u_{t_l}(z_1) - z_1|,$$

which leads to the aforementioned mesh tangling and is independent of the chosen time step size.

In order to avoid this, we may chose a different reference domain $\hat{\Omega}_{t_k}$, for example $\hat{\Omega}_{t_k} = \hat{\Omega}_{t_{k-1}}$. Because the boundary update $|(u_{t_k} - u_{t_{k-1}})_{|\Gamma}|$ is now smaller, we do not get the same problems from the discrete maximum principle or can avoid them by choosing the time step size small enough, as it can be seen in Figure 3.9.

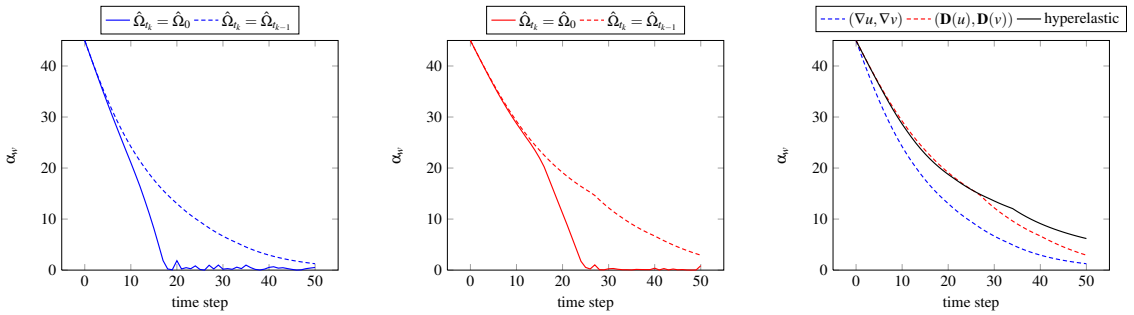
Note that in both variants of this method, the governing equations for the coordinate's components are decoupled. Because the boundary Γ only moves in x_2-direction, the deformations in x_1-direction are the identity at all times (see Figure 3.9). No direct control of $\det(\nabla R_K)$ is possible, although it enters the equations through the transformation and chain rule when assembling the linear system of equations, meaning that it is entirely coincidental if the solution u_h is a discrete variation.



(i) $t_{15} = 0.15$ (ii) $t_{50} = 0.50$

Figure 3.9: The mesh as given by the solution of (3.19) with $\hat{\Omega}_{t_k} = \hat{\Omega}_{t_{k-1}}$.

One indicator for the quality of a simplical mesh is the smallest spatial angle between any two edges. This quantity is plotted over time in Figure 3.10 (i). It can be seen that the mesh tangling in the method with fixed reference domain occurs in time step 16, meaning from there on the resulting mesh is no longer suitable for constructing finite element spaces on. For the method using the moving reference mesh, it can be observed that the angles remain above 1°, but this is still not a satisfactory lower bound. Computationally, this method is more expensive: In every time step, the linear system of equations has to be re-assembled and solved anew, whereas for a fixed reference domain, one could (for a sufficiently small problem) compute the *LU* factorisation once and just backwards-substitute in every time step.



(i) Worst angles using $(\nabla u, \nabla v)$ (see (3.22)).

(ii) Worst angles using $(\mathbf{D}(u), \mathbf{D}(v))$ (see (3.19)).

(iii) Worst angles for the meshes resulting from (3.19), (3.22), (3.24) .

Figure 3.10: Worst angles α_w for meshes resulting from different mesh quality functionals.

The next step is now to use a slightly more sophisticated mesh quality functional that couples the coordinate's components.

3.4.2. A functional coupling both components

Instead of (3.17) we now use the bilinear form

$$\forall (u,v) \in V(t) \times W(t): \quad a_{h,t}(u,v) = \int_{\hat{\Omega}_t} \mathbf{D}(u) : \mathbf{D}(v) dx \tag{3.20}$$

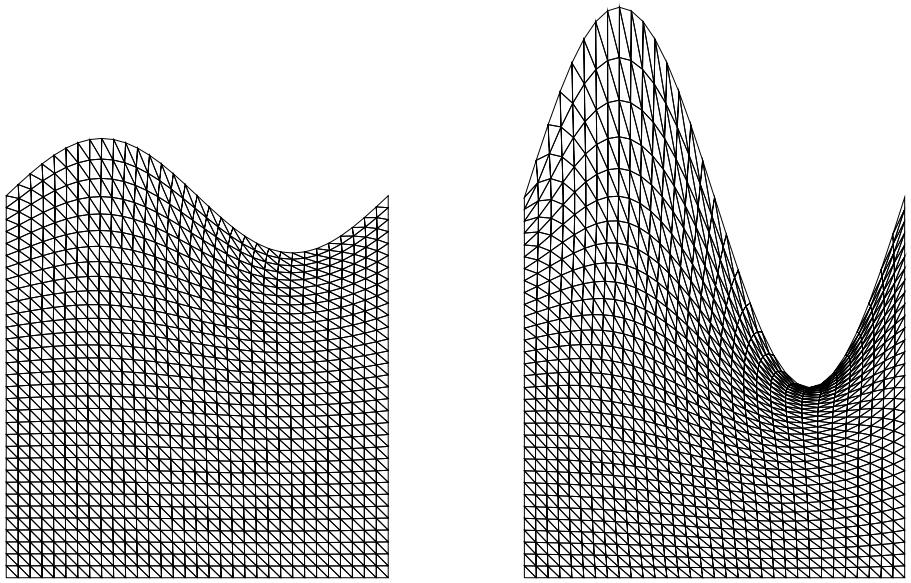
and with this the mesh quality functional

$$\mathcal{F}(u) := \frac{1}{2} a_{h,t}(u,u) \tag{3.21}$$

as before. Again, because of the boundedness, coerciveness and symmetry of a_h we get the existence and uniqueness of a solution u_h of (3.21) and that

$$u_h = \operatorname{argmin}_{u \in V(t)} \mathcal{F}(u) \Leftrightarrow \forall w \in W(t): \ \mathcal{F}'(u_h) w = 0. \tag{3.22}$$

Now the equations for the different components of u_h are coupled, meaning that vertices also move in x_1-direction even though the deformation Φ_Γ only has an x_2 component (see Figure 3.11).



(i) $t_{15} = 0.15$ (ii) $t_{50} = 0.50$

Figure 3.11: The meshes as given by the solution of (3.22) with $\hat{\Omega}_{t_k} = \hat{\Omega}_{t_{k-1}}$.

Using a fixed reference mesh results in mesh tangling again, due to a (different) maximum principle, as can be expected, so only the meshes for using a moving reference domain are shown in Figure 3.11. As can be seen from the worst angle plot in Figure 3.10 (ii), the mesh tangling occurs later than for the decoupled functional (3.19) and the angles are slightly better in the moving reference domain case, but still far from satisfactory.

This functional is computationally more expensive because the resulting linear system of equations is more strongly coupled (each component of each DoF is coupled to both components of all adjacent DoF as opposed to only the corresponding components as in the decoupled functional) and suffers from the same drawbacks as described in Section 3.4.1.

One common characteristic of both functionals presented so far is that the orientation preserving condition does not appear and is not enforced in any direct manner. This is addressed in the next example of a (nonlinear) stored-energy functional for hyperelastic materials.

3.4.3. Minimisation of hyperelastic strain energy

Define the nonlinear functional

$$\mathcal{F}(\Phi) = \int_{\hat{\Omega}_{t_n}} c_f(\|\nabla\Phi\|_F^2 - d)^2 dx + (\det(\nabla\Phi))^{p_d} dx + \frac{c_d}{\left(\det(\nabla\Phi) + \sqrt{\delta_r^2 + (\det(\nabla\Phi))^2}\right)^{p_d}} dx. \tag{3.23}$$

A functional of this class is derived in [Rum96] and other functionals of this class can be found in [HR11, Example 6.2.3 and Chapter 6.5.5]. More mathematical theory can be found in [Cia88]. This functional is not convex, but polyconvex (see Section 3.5.5), so minimisers are nonunique in general. However, note that for $\Phi = id$, the Frobenius norm term $(\|\nabla\Phi\|_F^2 - d)^2$ vanishes and that c_d can be chosen so that $\Phi = id$ minimises the other part of the functional.

Let $M : \mathbb{R} \to \mathbb{R}^{n\times n}$ be a matrix valued mapping. Recall that

$$\frac{d}{dt}\det(M(t)) = \det(M(t))\,\mathrm{tr}\left(M(t)^{-1}\frac{d}{dt}M(t)\right).$$

For two variations Φ, η using $M(t) := \nabla\Phi + t\nabla\eta$ locally on each $K \in \mathcal{T}_h$, we can compute the following derivatives:

$$\begin{aligned}
G_1(\Phi) &:= (\|\nabla\Phi\|_F^2 - d)^2, \quad G_1'(\Phi)\eta = 4\left(\|\nabla\Phi\|_F^2 - d\right)\nabla\Phi : \nabla\eta,\\
G_2(\Phi) &:= \det(\nabla\Phi)^{p_d}, \quad G_2'(\Phi)\eta = p_d \det(\nabla\Phi)^{p_d}(\nabla\Phi)^{-T} : \nabla\eta,\\
G_3(\Phi) &:= \left(\det(\nabla\Phi) + \sqrt{\det(\Phi)^2 + \delta_r^2}\right)^{-pd},\\
G_3'(\Phi)\eta &= -\frac{p_d\ \det(\nabla\Phi)\ \nabla\Phi^{-T} : \nabla\eta}{(\det(\nabla\Phi)^2 + \delta_r^2)^{\frac{p_d}{2}}\left(\det(\nabla\Phi) + \sqrt{\det(\nabla\Phi)^2 + \delta_r^2}\right)^{p_d}}.
\end{aligned}$$

We are now looking for an optimal variation Φ_h^* in the sense that

$$\Phi_h^* = \mathrm{argmin}_{\Phi\in V(t)}\,\mathcal{F}(\Phi), \tag{3.24}$$

which is not unique (see Section 3.5.6), but still needs to satisfy the necessary condition

$$\forall\eta \in W(t): \quad \mathcal{F}'(\Phi_h^*)\eta = 0.$$

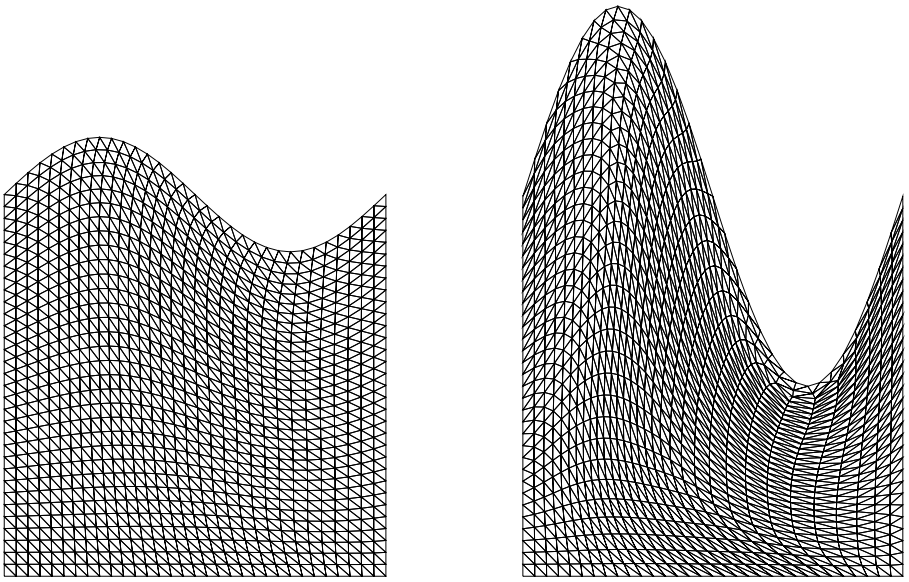
In this example, the parameters are

$$c_f = \frac{1}{100}, \delta_r = 10^{-8}, p_d = 2, c_d = 2\sqrt{\delta_r^2 + 1} + 2(\delta_r^2 + 1) + \delta_r^2\sqrt{\delta_r^2 + 1}.$$

c_d is chosen so that $\Phi = id$ minimises the det-based part of the functional. More information about these parameters and their choice will be given in Section 3.5. The fixed reference domain $\hat{\Omega}_{t_k} = \hat{\Omega}_0$ was used and the resulting minimisation problem was solved by using FEniCS' Newton-Krylov solver.

The resulting meshes can be seen in Figure 3.12. A comparison of the worst angles of all the methods mentioned is in Figure 3.10 (iii). The main improvement is that here, we can directly control the term $1/\det(\nabla\Phi)$ and guarantee that a discrete solution of (3.24) is indeed in the space of discrete variations. However, the functional measures the energy of a deformation with regard to the reference domain (in this case $\hat{\Omega}_0$), which means that for every cell it was constructed in such a way that the identity would be a minimiser *if it were in the space of variations with the correct boundary values*. This means that every cell $\Phi(K)$ should be "similar" to K, which might not be ideal if the reference cell K was of poor shape.

This leads to the idea of measuring the energy of a transformation of a different reference cell, which will be elaborated in Section 3.5.



(i) $t_{15} = 0.15$ (ii) $t_{50} = 0.50$

Figure 3.12: The meshes as given by the solution of (3.24) with $\hat{\Omega}_{t_k} = \hat{\Omega}_0$.

3.5. A class of nonlinear mesh quality functionals

Recall (3.23):

$$\mathcal{F}(\Phi) = \int_{\hat{\Omega}_{t_n}} c_f(\|\nabla\Phi\|_F^2 - d)^2 dx + (\det(\nabla\Phi))^{p_d} dx + \frac{c_d}{\left(\det(\nabla\Phi) + \sqrt{\delta_r^2 + (\det(\nabla\Phi))^2}\right)^{p_d}} dx.$$

Since we are working with the space $\mathcal{D}$ of admissible variations and need to somehow control $\forall K \in \mathcal{T} : \det(\nabla R_K)$, this functional is very useful. In the following section, we want to derive the more general class of functionals it comes from. It can be shown that a mesh quality functional has to be of this more general class if it is to have some very important properties.

Note that here, we take a different approach than in the examples in Section 3.4: There, PDE-based mesh quality functionals were regarded with little to no motivation as to why they might be suitable. Here, necessary or useful properties of a mesh quality functional are given, which lead to a certain class of PDEs.

The derivation of this class of nonlinear mesh quality functions follows [Rum96] but uses more general formulations from [HR11, Chapter 6].

3.5.1. Basic properties

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

$$\mathcal{F}(\Phi) = \int_\Omega \mathcal{L}(x, \Phi) dx, \tag{3.25}$$

which is already a (sensible) assumption on the regularity. In practice, we need $\mathcal{F}_h : \mathcal{D}_h \to \mathbb{R}$, so the basic axiom is formulated (as in [Rum96]) stronger still:

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

$$\mathcal{F}_h(\Phi_h) = \int_\Omega \mathcal{L}(x, \Phi_h) dx = \sum_{K \in \mathcal{T}_h} \mu_K \int_K \mathcal{L}_K(x, \Phi_h) dx, \tag{3.26}$$

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

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

We now state some desirable properties for $\mathcal{F}_h$ and thus $\mathcal{L}$.

$$\text{Translation invariance :} \qquad \forall c \in \mathbb{R}^d : \mathcal{F}(K, \Phi + c) = \mathcal{F}(K, \Phi) \tag{3.27}$$

$$\Rightarrow\ \forall K \in \mathcal{T}_h : \qquad \exists \mathcal{L}_K^G : K \times \mathrm{SL}_d \to \mathbb{R} : \mathcal{L}_K(\cdot, \Phi) = \mathcal{L}_K^G(\cdot, \nabla\Phi) \tag{3.28}$$

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

Assume that K is given by the reference cell $\hat{K}_R$ and the reference mapping $R_K : \hat{K}_R \to K$. Note that $\hat{K}_R \neq \hat{K}$ in general and may be different for every K. The choice of this reference cell is very important, see Section 3.5.3. We have the relations

$$\begin{aligned} \Phi &: \mathcal{T}_h \to \Phi(\mathcal{T}_h), \quad \Phi_{|K}(x) = \Phi(R_K(\hat{x})) \text{ where } R_K(\hat{x}) = x \\ \Phi \circ R_K &: \hat{K} \to \Phi(R_K(\hat{K})), \quad R_K(\Phi) = \Phi \circ R_K, \end{aligned}$$

see Figure 3.13.

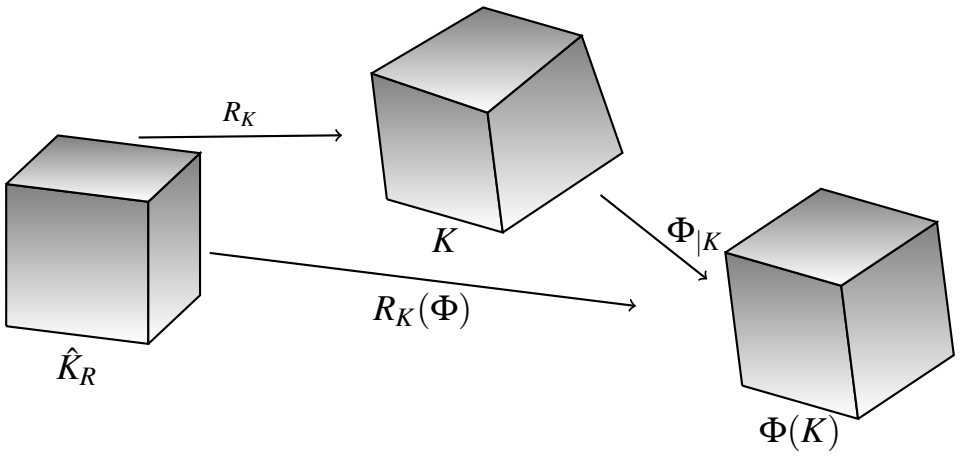


Figure 3.13: The relations of the mapping R_K.

Ruling out any other dependencies, we deduce that

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

Furthermore, there are the principles of

$$\text{Frame indifference :} \qquad \forall Q \in SO_d : \mathcal{L}\left(\nabla R_K(\Phi)(\cdot)\right) = \mathcal{L}\left(Q \nabla R_K(\Phi)(\cdot)\right). \tag{3.29}$$

$$\text{Isotropy :} \qquad \forall Q \in SO_d : \mathcal{L}(\nabla R_K(\Phi), \cdot) = \mathcal{L}(\nabla R_K(\Phi)(\cdot) Q). \tag{3.30}$$

Translation invariance means that the quality of a cell only depends on its shape, not its position. *Frame indifference* means that a cell's quality does not depend on the observer's position,

while *isotropy* means it does not depend on the coordinate system of the reference cell $\hat{K}_R$. It will be shown in Section 3.5.3 that this already places a restriction on the choice of the reference element.

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-Erikson Representation Theorem [Cia88, Theorem 3.6-1], we deduce that

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

and define

$$F(\nabla R_K(\Phi)) := \int_K L(\|\nabla R_K(\Phi)(x)\|_F^2, \|\operatorname{Cof}(\nabla R_K(\Phi)(x))\|_F^2, \det(\nabla R_K(\Phi)(x)))dx, \tag{3.31}$$

so $\mathcal{L}$ is a *stored energy function*.

Remark 3.2. *In this section, the general structure of a mesh quality functional was deduced from some (reasonable) assumptions on its behaviour. It is no surprise that the resulting stored energy function fits into a more general framework based on the same set of assumptions, namely the modelling of compressible, hyperelastic materials (see Section 3.5.5).*

Because of this, the complete set of analysis from [Bal76, Cia88] can be applied, see Section 3.5.6, as long as the stored energy function fulfils the basic assumptions of the theorems in that section, namely polyconvexity, a stability property and coerciveness.

Remark 3.3. *Geometric interpretations of the terms*

The terms $\|\nabla\Phi\|_F, \operatorname{Cof}(\nabla\Phi)$ *and* $\det(\nabla\Phi)$ *have obvious geometric meanings that are well-known in mechanics. For* $d = 3$

i) $\|\nabla\Phi\|_F$ *expresses the length change of curves under the deformation* Φ *([Cia88, Section 1.8]),*

ii) $\operatorname{Cof}(\nabla\Phi)$ *expresses the area change of surfaces under the deformation* Φ *([Cia88, Theorem 1.7-1]) and*

iii) $\det(\nabla\Phi)$ *expresses the volume change of volumes under the deformation* Φ *([Cia88, 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.

3.5.2. Properties of the local integrand

Before going deeper into the analysis, I want to discuss some more properties the stored energy function should have for the specific purpose of mesh optimisation

Note that for simplices or parallelpipeds, $\nabla R_K(\Phi) = \text{const}$. In the following, some properties of the local stored energy function L_K will be derived, from which properties for the local reference cell $\hat{K}$ can be deduced.

We for now assume that the map R_K is affine, which is the case for simplices or if K is parallelpiped like $\hat{K}$.

$$\begin{aligned}
&\Rightarrow \nabla R_K(\Phi) = \text{const} \\
&\Rightarrow F(\nabla R_K(\Phi)) = \text{vol}(K) L(\|\nabla R_K(\Phi)\|_F^2, \|\operatorname{Cof}(\nabla R_K(\Phi))\|_F^2, \det(\nabla R_K(\Phi)))
\end{aligned}$$

Since $\hat{K}$ is the optimal cell for the quality measure defined by L and thus F,

$$F(\nabla R_{\hat{K}}) = F(id) = \text{vol}(K) \min_{A \in SL_d} L(\|A\|_F^2, \|\operatorname{Cof}(A)\|_F^2, \det(A)).$$

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

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

Because we want to stay in the space of variations, we want L to fulfil the additional stability property

$$\lim_{\det(A) \to 0} L(\cdot,\cdot,\det(A)) = \infty, \tag{3.32}$$

both because a variation with vanishing determinant should blow up the value of the local functional, and because this property is needed for [Cia88, Theorem 7.7-1] (also see [HR11, Theorem 6.2.4]) that ensures (together with other conditions) the existence of a minimiser. This property is based on the idea that "infinite stress must accompany extreme strains" ([Ant83]) and is revisited in Section 3.5.5.

If R_K is not linear, the same argumentation applies since the integral is monotone and $L(\cdot,\cdot,\cdot) > 0$.

3.5.3. Choosing reference cell shapes

For a given cell K, choosing the reference cell $\hat{K}_R$ is equivalent to defining an optimal cell, and for every cell the quality functional measures the energy of the transformation to its current shape from this optimal cell. For every $K \in \mathcal{T}_h$, a different optimal reference cell $\hat{K}_R$ can be chosen.

In Section 3.4.3, for every $K \in \mathcal{T}_h(t_n)$ its representation in the computational domain $\hat{\Omega}_{t_k} = \hat{\Omega}_0$ was chosen. This means ill-shaped reference cells in the computational domain $\hat{\Omega}$ (which is even more likely if a truly moving reference domain like $\hat{\Omega}_{t_k}$ is used) will lead to ill-shaped cells in Ω.

So it is useful to define reference cells that are independent of the computational domain used for other PDEs. One very important choice 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

3.14).

Hypercubes :

$$\hat{Q}_n = [-1,1]^d = \hat{Q} \tag{3.33}$$

Simplices :

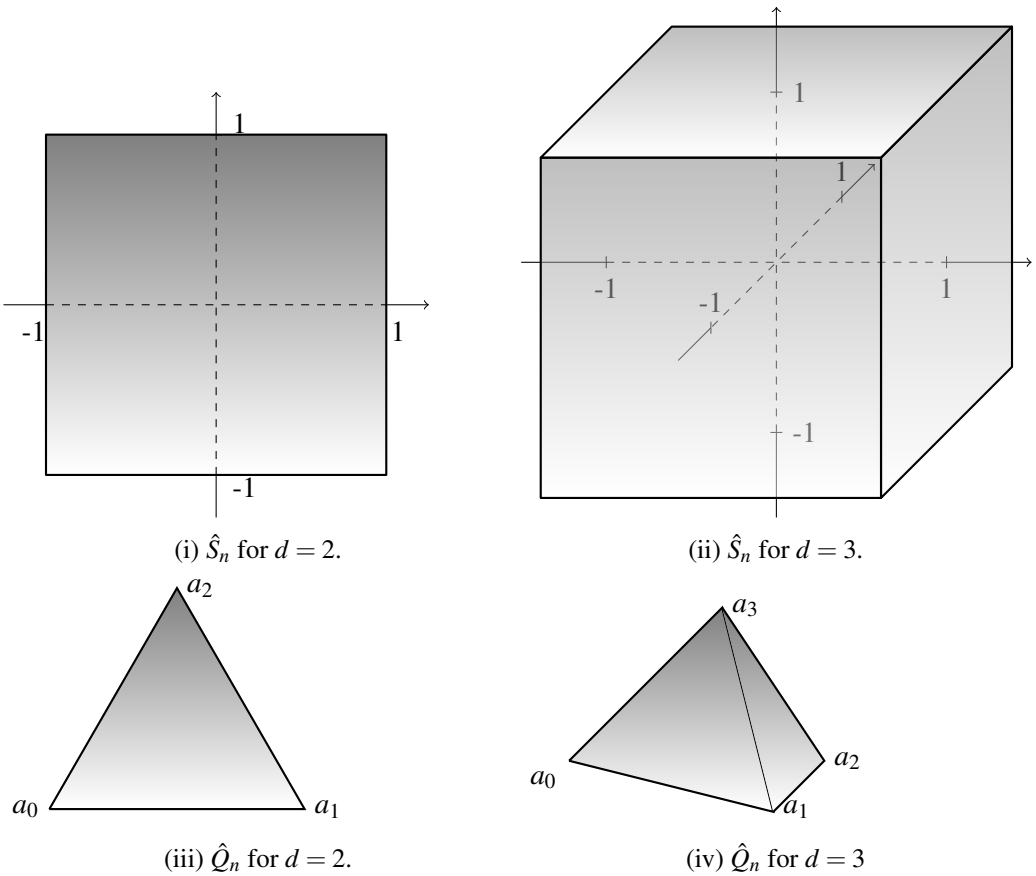
$$\hat{S}_n = \left\{ x \in \mathbb{R}^d : x = \sum_{i=0}^{s} \lambda_i a_i, \text{ where } \forall i \in \{1,\dots,s\} : \lambda_i \in \mathbb{R}_{\geq 0}, \sum_{i=0}^{s} \lambda_i = 1 \right\} \tag{3.34}$$

with

$$\begin{cases} a_0 = (0,0)^T, a_1 = (0,1)^T, a_2 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right), & d = 2 \\ a_0 = (0,0,0)^T, a_1 = (0,1,0)^T, a_2 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right), a_3 = \left(\frac{1}{2}, \frac{\sqrt{3}}{6}, \frac{\sqrt{6}}{3}\right) & d = 3 \end{cases} \tag{3.35}$$

The hypercubes $\hat{Q}_n$ are just the regular reference d-hypercubes, while the *normalised* simplices $\hat{S}_n$ are not. These reference cells have the additional benefit of requiring no combinatorial testing in the evaluation of the local integrands L, as they are invariant with regard to their local numbering. For a given numbering of the reference cell's vertices and its faces' vertices, we consider permutations of these numberings that do not change the orientation of the respective entities.

For simplices, one can get all local numberings by taking the even permutations of the cell's vertices, as every vertex is connected to every other vertex through an edge.



(i) $\hat{S}_n$ for $d = 2$.

(ii) $\hat{S}_n$ for $d = 3$.

(iii) $\hat{Q}_n$ for $d = 2$.

(iv) $\hat{Q}_n$ for $d = 3$

Figure 3.14: Reference cells for $d = 2, 3$.

For hypercubes, there are four local numberings of positive orientation in $2d$ (see Figure 3.15) and 24 in $3d$.

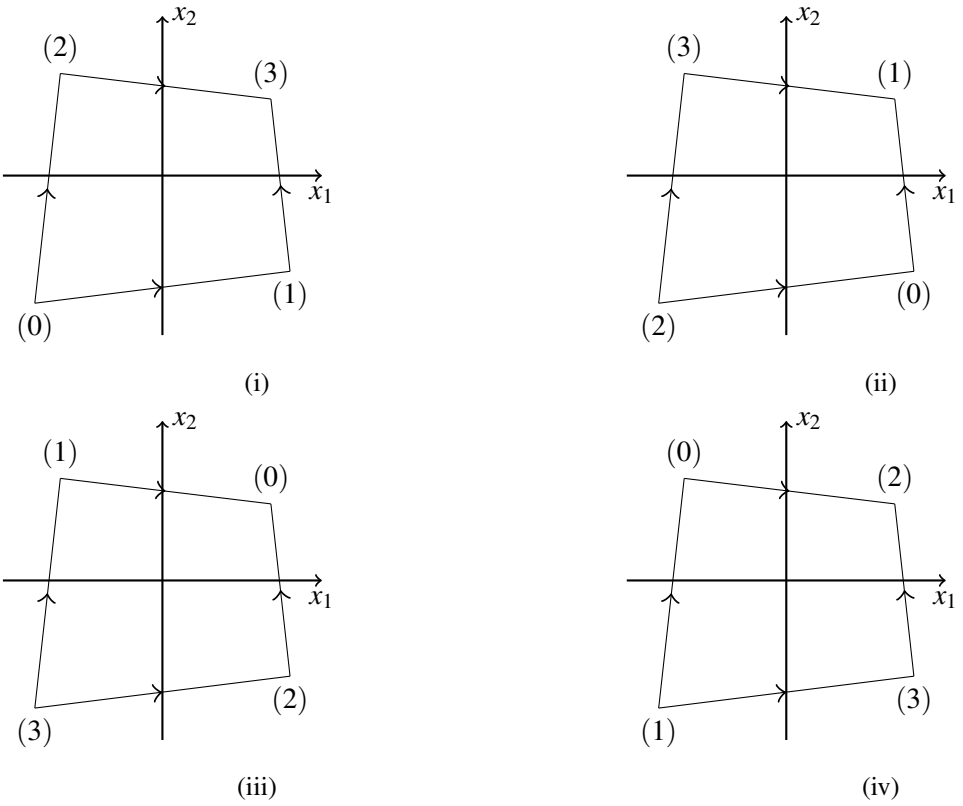


Figure 3.15: Four possible local numberings of a reference hypercube in $2d$.

Remark 3.4. *If other reference cells are chosen which are not invariant with regard to admissible vertex permutations, one has to check all numberings of the reference cell (for simplices: 2 ($d=1$), 3 ($d=2$), 12 ($d=3$); for hypercubes: 2 ($d=1$), 4 ($d=2$), 24 ($d=3$)) and select the reference cell which gives the lowest functional value. As this is very costly, it is preferable to use the invariant reference cells* (3.33)*,* (3.34) *or to find the right permutation by construction.*

Conversely, it can be shown that if the functional is invariant with regard to the numbering of the reference cell $\tilde{K}$, then the reference cell is $\hat{S}_n$ (or $\hat{Q}_n$, respectively) up to scaling and rotation.

Lemma 3.5. *Let $\mathcal{P}$ be the set of all mappings defining an admissible vertex permutation on the reference cell $\tilde{K}$. If every local functional F is invariant with regard to $\mathcal{P}$, meaning*

$$\forall \Phi \in \mathcal{D}_h : \forall p \in \mathcal{P} : F(\nabla\Phi \circ R_K \circ \nabla p) = F(\nabla\Phi \circ R_K)$$

then the reference cell $\tilde{K}$ is $\hat{S}_n$ (if $\tilde{K}$ is a simplex) or $\hat{Q}_n$ (if $\tilde{K}$ is a hypercube) up to scaling and rotation.

Proof. The proof for simplices in $3d$ can be found in [Rum96, Section 5].

We have

$$\forall \Phi \in \mathcal{D}_h : \forall p \in \mathcal{P} : \int_K \mathcal{L}_K(\nabla\Phi \circ R_K \circ \nabla p(x))dx = \int_K \mathcal{L}_K(\nabla\Phi \circ R_K(x))dx.$$

If K is such that R_K is affine, and Φ is affine too, then all matrices (including $\nabla p =: P$) are constant in x and we have

$$\mathcal{L}_K(\nabla\Phi \circ R_K \circ P) = \mathcal{L}_K(\nabla\Phi \circ R_K)$$

Since this has to hold for all admissible (affine) Φ, this implies

$$\forall A \in \mathrm{SL}_d : \|AP\|_F = \|A\|_F, \|\operatorname{Cof}(AP)\|_F = \|\operatorname{Cof}(A)\|_F, \det(AP) = \det(A).$$

The last identity directly implies $\det(P) = 1$. Let $\{E^{ij}\}_{1\leq i,j\leq d}$ be the canonical matrix basis in $\mathbb{R}^{d\times d}$ and define the matrices $A^{ij} = E^{ij} + \varepsilon I_d$. We can calculate that

$$\begin{aligned}
\forall 1 \leq i \leq d : &\quad \det(A^{ij}) = \varepsilon^3, \\
\forall 1 \leq i \leq d : &\quad \operatorname{Cof}(A^{ii})_{kl} = \begin{cases} 0, & k \neq l \\ \varepsilon^2, & k = l \neq i \\ \varepsilon(1+\varepsilon), & k = l = i \end{cases}, \\
\forall 1 \leq i,j \leq d, i \neq j : &\quad \operatorname{Cof}(A^{ij})_{kl} = \begin{cases} 0, & k \neq l, (k,l) \neq (i,j) \\ \varepsilon^2, & k = l \neq i \\ -\varepsilon, & k = l = i \end{cases}.
\end{aligned}$$

From this it follows that

$$\lim_{\varepsilon\to 0} \operatorname{Cof}(A^{ij}) = E^{ij},$$

so since $\|\cdot\|_F$ is continuous, passing to the limit implies

$$\begin{aligned}
&\|E^{ij}P\|_F = \|P_{\cdot j}\|_2 = 1, \quad \|\operatorname{Cof}(P)_{\cdot j}\|_2 = 1 \\
\Rightarrow\ &\forall 1 \leq k,l \leq d, k \neq l : P_{\cdot k} \perp P_{\cdot l} \\
\Rightarrow\ &P \in \mathrm{SL}_d.
\end{aligned}$$

So P is a rotation. For 2-simplices, any even vertex permutation rotates the triangle's vertices, so for the functional value to remain unaffected, all edges must have the same length. The same is true in $3d$ if one considers an even vertex permutation of 3 vertices, which rotates the corresponding 2-simplex around its centre. Since this can be done for each boundary 2-simplex of the 3-simplex, all edges of the 3-simplex must have the same length.

For 2-hypercubes, by the same argument, every admissible vertex permutation rotates the quadrilateral around its centre, so all edges have to be of the same length and all angles have to be $\frac{\pi}{2}$. The same is true for the corresponding rotations in $3d$, which concludes the proof. □

Remark 3.6. *The consequence of this is that the functional's assumed invariance to the local numbering of the reference cell already implies the isotropy of the functional F and restricts us to scaled versions of the reference elements $\hat{S}_n$ and $\hat{Q}_n$. Conversely, if we chose different reference cells, we lose these important properties and $\mathcal{L}$ no longer solely depends on $\|\nabla R_K(\Phi)\|_F^2$, $\|\operatorname{Cof}(\nabla R_K(\Phi))\|_F$ and $\det(\nabla R_K(\Phi))$, but also on the variant of the reference cell.*

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

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.

3.5.4. Choosing reference cell scales

Now that the restrictions of the local functional's isotropy on the reference cell's shape are clear, let us examine the scaling of the reference cells. The first question is if a mesh quality functional can be scaling invariant, meaning an optimal reference cell can be characterised just by its shape and not its size. This would of course mean that *r*-adaptivity is not possible, which could partially be mitigated by pre-refining or condensing the mesh.

Assume for simplicity that R_K is affine. Then scaling invariance means that

$$\forall A \in \mathrm{SL}_d : \forall \lambda \in \mathbb{R}_+ : F(\lambda A) = F(A).$$

Recalling that

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

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

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

A functional based on F cannot be polyconvex as F is not convex in its argument $A \in \mathbb{R}^{d\times d}$ (see Section 3.5.5) and does not satisfy the coerciveness condition in Theorem 3.11 as e.g. in general $F(A) < +\infty$ 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. [FK02]). 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 $\hat{K}_R$. Assume for now that we are in the isotropic case. Because of Lemma 3.5, it is clear that

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

and we call h the *optimal scales*. See Figure 3.16 for the connection between $R_K, R_{K,n}$ and $\Phi \circ R_K = R_K(\Phi)$.

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

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

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

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

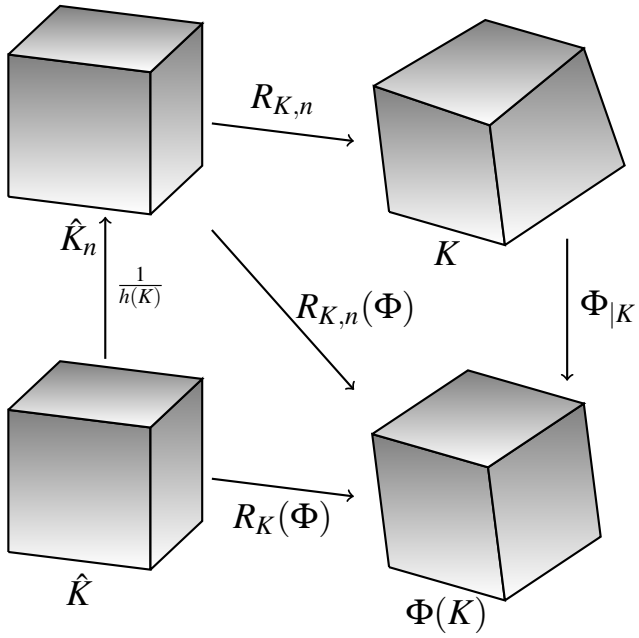


Figure 3.16: The relations of the mapping $R_{K,n}$.

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

$$h(K) = \sqrt[d]{\lambda(K) \sum_{T \in \mathcal{T}_h} \int_{\hat{K}_n} \det(\nabla R_{K,n}(id))}. \tag{3.36}$$

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 3.7 (Some mesh concentration functions)**.**

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

ii) 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 $\tilde{\mathcal{T}}_h$.

iii) 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.

iv) 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.

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

Remark 3.8. *Note that in Example 3.7 iv), there is a great difference between*

$$c(K) = f(\mathrm{dist}(s_{\Phi^*(K)}, \Gamma))$$

and

$$c(K) = f(\mathrm{dist}(s_K, \Gamma))$$

The first concentration function uses the distance of the cell $\Phi^*(K)$, which is a priori unknown. This means that the concentration (and thus the local optimal scale $h(K)$) becomes part of the problem and must be treated accordingly. More on this can be found in Section 3.5.7.

The second concentration function uses the distance of the original cell K for computing the optimal scale $h(K)$. This does not introduce any new difficulties into the functional, but it might result in a not very useful cell size distribution, as $|\mathrm{dist}(s_{\Phi^*(K)}, \Gamma) - \mathrm{dist}(s_K, \Gamma)|$ might be large.

3.5.5. Relation to hyperelastic materials, polyconvexity

Let us briefly revisit the class of mesh quality functionals from (3.31) from Section 3.5 to see that it models an (isotropic), compressible, hyperelastic material.

The first assumption was *translation invariance* ((3.27)), 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*. An equally important assumption prior to this was that the strain energy can be expressed in the form

$$\mathcal{F}(\Phi) = \int_\Omega \mathcal{L}(x, \nabla\Phi) dx, \tag{3.37}$$

because 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 carries the additional benefit that the constitutive equation

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

is formally equivalent to the equations

$$\forall \eta : \bar{\Omega} \to \mathbb{R}, \eta_{|\Gamma_0} = 0 : \mathcal{F}'(\Phi^*)\eta = 0 \tag{3.38}$$

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 [Cia88, Theorem 4.1-2] for the details.

Since we do not pose the incompressibility condition $\det(\nabla\Phi) = 1$, the material is compressible. All we did by choosing reference cells and scales in Section 3.5.3 and Section 3.5.4 was to define a discretised version of a material property function. Choosing reference cells so that the resulting local functional is isotropic just means we chose an isotropic material, which simplifies the representation of $\mathcal{L}$, but is not necessary for the existence results in Section 3.5.6.

We still need to examine the properties of the stored energy function $\mathcal{L}$. The simplest case is that $\mathcal{L}$ is convex in the sense that

$$x \in \bar{\Omega} : \mathcal{L}(x,\cdot) : \mathrm{SL}_d \to \mathbb{R}$$

is convex. If it is furthermore strictly convex, the strain energy (3.37) has at most one stationary point. This contradicts the lack of uniqueness of solutions to elasticity problems observed in physical situations see [Gur78, Nol78] for examples.

Even without this contradiction there is the issue of the behaviour of $\mathcal{L}$ as $\det(\nabla\Phi) \to 0$, which corresponds to the scenario of "infinite stress". Apart from the mathematical reasoning that

$$\lim_{\det(A)\to 0} \mathcal{L}(x,A) = +\infty, A \in \mathrm{SL}_d$$

because we are working in the space of orientation preserving deformations, there is also the physically motivated notion that "infinite stress must accompany extreme strains" [Ant83] and consequently that volumes can only be annihilated by infinite force. But [Cia88, Theorem 4.8-1] shows that convexity of $\mathcal{L}$ contradicts this behaviour.

So indeed it is not feasible to work with a convex stored energy function $\mathcal{L}$. However, John Ball was able to replace this by the weaker requirement that $\mathcal{L}$ is *polyconvex* and prove the existence theorems stated in Section 3.5.6.

Definition 3.4. *A stored energy function $\mathcal{L} : \bar{\Omega} \times \mathrm{SL}_d \to \mathbb{R}$ is called* polyconvex*, iff*

$$\begin{aligned} \forall x \in \bar{\Omega} : \exists \mathcal{L}_c(x,\cdot,\cdot,\cdot) : \mathrm{SL}_d \times \mathrm{SL}_d \times (0,+\infty) \to \mathbb{R} \text{ convex} : \\ \forall A \in \mathrm{SL}_d : \mathcal{L}(x,A) = \mathcal{L}_c(x,A,\mathrm{Cof}(A),\det(A)). \end{aligned}$$

Example 3.9. *Consider the matrices*

$$A = \begin{pmatrix} 2 & & \\ & 1 & \\ & & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & & \\ & 2 & \\ & & 1 \end{pmatrix} \text{ and } F : [0,1] \to \mathrm{SL}_d, F(\lambda) = \lambda A + (1-\lambda)B = \begin{pmatrix} 1+\lambda & & \\ & 2-\lambda & \\ & & 1 \end{pmatrix},$$

which is the convex combination of A and B.

i) Consider the mapping

$$f(M) = \det(M).$$

Since $\det(F_\lambda) = 2+\lambda-\lambda^2$ *and* $\lambda\det(A) + (1-\lambda)\det(B) = 2$*, we have* $f(F_{\frac{1}{2}}) = \frac{9}{4} > 2$*, this mapping is not convex in M.*

However, it is polyconvex since the function $f_c : (0,+\infty) \to \mathbb{R}, f_c(\delta) = \delta$ is convex.

ii) Consider the mapping

$$g(M) = \|\mathrm{Cof}(M)\|_F^2.$$

We can compute that

$$\mathrm{Cof}(A) = \mathrm{diag}(1,2,2), \mathrm{Cof}(B) = \mathrm{diag}(2,1,2), \mathrm{Cof}(F_\lambda) = \mathrm{diag}(2-\lambda, 1+\lambda, (2-\lambda)(1+\lambda)),$$

With this, $\|\mathrm{Cof}(A)\|_F^2 = 9 = \|\mathrm{Cof}(B)\|_F^2$ *and* $\|\mathrm{Cof}(F_\lambda)\|_F^2 = \lambda^4 - 2\lambda^3 - \lambda^2 + 2\lambda + 9$*. Since* $g(F_{\frac{1}{2}}) = \frac{153}{16} > 9$*, it is not a convex function.*

It is however polyconvex, since the function $f_c : \mathrm{SL}_3 \to \mathbb{R}, f_c(H) = \|H\|_F^2$ is convex because

$$\forall F, G \in \mathrm{SL}_3 : f_c''(F)(G,G) = 2\|G\|_F^2 \geq 0.$$

iii) If $a > 0, b > 0$ and $\Gamma : (0,+\infty) \to \mathbb{R}$ is convex, the stored energy function

$$\mathcal{L} : \bar{\Omega} \times \mathrm{SL}_3, \quad \mathcal{L}(x,F(x)) = a\|F(x)\|_F^2 + b\|\operatorname{Cof}(F(x))\|_F^2 + \Gamma(\det(F(x)))$$

is polyconvex since the function

$$\mathcal{L}_c : \mathbb{R}^{3\times3} \times \mathbb{R}^{3\times3} \times (0,+\infty), \quad \mathcal{L}(F,H,\delta) = a\|F\|_F^2 + b\|H\|_F^2 + \Gamma(\delta)$$

is convex.

Remark 3.10. *Going into the details of polyconvex stored energy functions is beyond the scope of this work and the reader is referred to [Bal76, Cia88]. 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 [Cia88, Chapter 4.10].*

3.5.6. Existence of minimisers

The following theorems are taken from [Cia88, Section 7], although the truly pioneering work on the subject was [Bal76]. The results are for $d = 3$ but can be restricted to the case $d = 2$. See [Mie05] for a reference dealing with the case $d = 2$.

Theorem 3.11 (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:*

i) ***Polyconvexity:***

$$\forall x \in \Omega \text{ a.e.} : \exists \tilde{L}(x,\cdot,\cdot,\cdot) : \mathrm{SL}_3 \times \mathrm{SL}_3 \times (0,\infty) \to \mathbb{R} : \forall F \in \mathrm{SL}_3 : \\ \mathcal{L}(x,F) = \tilde{L}(x,F,\operatorname{Cof}(F),\det(F)),$$

where $\forall (F,H,\delta) \in \mathrm{SL}_3 \times \mathrm{SL}_3 \times (0,\infty) : \tilde{L}(\cdot,F,H,\delta) \in L^1(\Omega)$.

ii) ***Stability:***

$$\forall x \in \Omega \text{ a.e} : \lim_{\det(F)\to 0} \mathcal{L}(x,F) = +\infty$$

iii) ***Coerciveness:***

$$\exists \alpha \in \mathbb{R}_+, \beta \in \mathbb{R}, 2 \le p \in \mathbb{N}, \frac{p}{p-1} \le q \in \mathbb{N}, 1 < r \in \mathbb{R} : \\ \forall x \in \Omega \text{ a.e.}, \forall F \in \mathrm{SL}_3 : \\ \mathcal{L}(x,F) \ge \alpha(\|F\|_F^p + \|\operatorname{Cof}(F)\|_F^q + \det(F)^r) + \beta$$

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

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

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 [Cia88, Theorem 7.7-1] and the corresponding proof. □

Remark 3.12.

i) 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 [Cia88, 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.*

ii) The resulting problem is a pure displacement problem.

iii) The proof does not use the isotropy assumption on the function $\tilde{L}$ and remains valid in the absence of this property.

iv) The minimiser is not unique in general, which is observed for hyperelastic materials in practice.

Theorem 3.13 (Existence of minimisers for problems with a unilateral boundary condition of place and a locking constraint)**.** *Assume $\Omega \subset \mathbb{R}^3$ to be a given domain such that $\partial\Omega \supset \Gamma_0, \Gamma_1, \Gamma_2$ and* $\mathrm{vol}_2(\partial\Omega \setminus \{\Gamma_0 \cup \Gamma_1 \cup \Gamma_2\}) = 0$. *Let Γ_i be $d\sigma$-measurable, relatively open, disjoint and* $\mathrm{vol}_2(\Gamma_0) > 0$. *Assume further that we have $\mathcal{L} : \Omega \times \mathrm{SL}_3 \to \mathbb{R}$ with the same properties as in Theorem 3.11 (polyconvexity, regularity, coerciveness). Let $\Lambda : \mathrm{SL}_3 \to \mathbb{R}$ be a polyconvex function such that*

$$\exists \hat{\Lambda} : \mathrm{SL}_3 \times \mathrm{SL}_3 \times (0, +\infty) \to \mathbb{R} : \quad \forall F \in \mathrm{SL}_3 : \Lambda(F) = \hat{\Lambda}(F, \mathrm{Cof}(F), \det(F)).$$

Let further $\Gamma \subset \mathbb{R}^3$ be closed and $\phi_0 \in L^1(\Gamma_0, \mathbb{R}^3)$ such that

$$\begin{aligned} \emptyset \neq \mathcal{D}_{\phi_0,\Gamma} := \{\Phi \in W^{1,p}(\Omega) : & \mathrm{Cof}(\nabla\Phi) \in L^q(\Omega), \det(\nabla\Phi) \in L^r(\Omega), \\ & \forall x \in \Omega \ a.e. : \det(\nabla\Phi)(x) > 0, \\ & \forall x \in \Omega \ a.e. : \Lambda(\nabla\Phi(x)) \leq 0, \\ & \forall x \in \Gamma_0 \ d\sigma \ a.e. : \Phi(x) = \phi_0(x), \\ & \forall x \in \Gamma_2 \ d\sigma \ a.e. : \Phi(x) \in \Gamma_2\} \end{aligned}$$

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 $\exists \Phi \in \mathcal{D}_{\phi_0,\Gamma} : \mathcal{F}(\Phi) < +\infty$, there exists at least one

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

Proof. See [Cia88, Theorem 7.8-1] and the corresponding proof. □

Remark 3.14.

i) A variant of this theorem with proof for the discrete case can be found in [Rum96].

ii) The additional boundary condition in Theorem 3.13 is a unilateral boundary condition of place *as defined in Definition 3.1 and is well-known for elasticity problems. It is particularly useful for the application for mesh deformations and the underlying mathematical property of the definition of the space of variations (Definition 3.2).*

iii) The function Λ in Theorem 3.13 defining an additional constraint is a locking function *as per Definition 3.1. The notion of a* locking *constraint was introduced for linearised elasticity in [Pra57] for materials that become locked if some measure of strain reaches a critical level. For the nonlinear case, [CN85] proposed a locking constraint based on the deviatoric part of the Green-St-Venant stress. Let $\alpha \in \mathbb{R}_+$ and define*

$$E := \left((\nabla\Phi)^T\nabla\Phi - I_3\right), \quad E_d := E - \frac{1}{3}\mathrm{tr}(E)I_3,$$
$$\Lambda : \mathrm{SL}_3 \to \mathbb{R}, \quad \Lambda(F) := \|E_d\|_F^2 - \alpha.$$

The locking constraint is entirely optional. Although it has not been used in the current work, it could provide an additional tool to further improve the methods.

Theorem 3.15 (Existence of minimisers for displacement-traction problems with injectivity constraint)**.** *Assume $\Omega \subset \mathbb{R}^3$ to be a given domain such that $\partial\Omega \supset \Gamma_0, \Gamma_1$, and $\mathrm{vol}_2(\partial\Omega \setminus \{\Gamma_0 \cup \Gamma_1\}) = 0$. Let Γ_i be $d\sigma$-measurable, relatively open, disjoint and $\mathrm{vol}_2(\Gamma_0) > 0$. Assume further that we have $\mathcal{L} : \Omega \times \mathrm{SL}_3$ with the same properties as in Theorem 3.11 (polyconvexity, regularity, coerciveness) with $p > 3$.*

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

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

Let $f \in L^p(\Omega)$ such that the linear form

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

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 minimiser

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

and all minimisers $\Phi^* : \Omega \to \mathbb{R}^3$ are injective in Ω a.e..

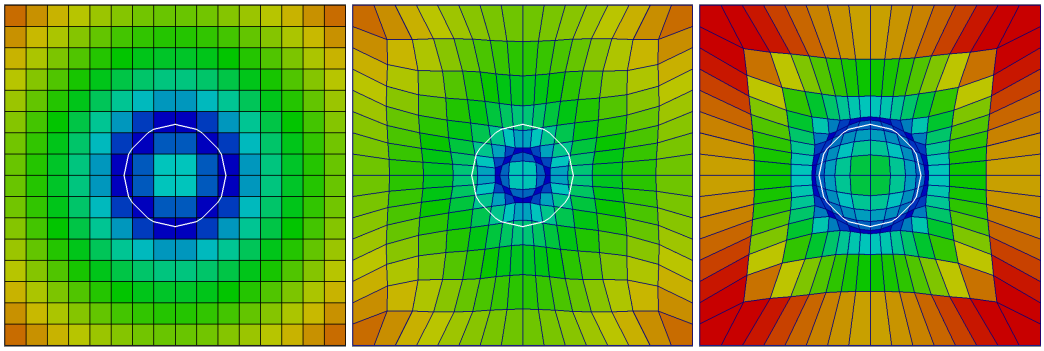
Proof. See [Cia88, Theorem 7.9-1] and the corresponding proof. □

Remark 3.16.

i) The assumption that there is no surface force g was made in [Cia88, Theorem 7.9-1] only for simplicity.

ii) The results can be extended to the case $p > 2$.

iii) The additional injectivity constraint $\int_\Omega \det(\nabla\Phi)dx \leq \mathrm{vol}(\Phi(\Omega))$ is fulfilled automatically in the case of a pure displacement problem ($\mathrm{vol}_2(\partial\Omega \setminus \Gamma_0) = 0$ in Theorem 3.11) or fixed in combination with a unilateral boundary condition of place ($\mathrm{vol}_2(\partial\Omega \setminus \{\Gamma_0 \cup \Gamma_2\}) = 0$ and $\mathrm{vol}_2(\partial\Phi(\Omega) \setminus \{\phi_0(\Gamma_0) \cup \Gamma\}) = 0$ in Theorem 3.13) fulfilling an appropriate condition on the volume.



(i) Initial configuration. (ii) After one iteration. (iii) End of Picard iteration.

Figure 3.17: Iterates of a Picard iteration.

3.5.7. *r*-adaptivity

As we have seen in Section 3.5.4, *r*-adaptivity is nothing but choosing the local optimal scales

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

with a given mesh concentration function c. As was mentioned in Remark 3.8, there are situations in which we would rather use $c = c(\Phi^*(K))$.

Example 3.17. *Recall the two formulations from Example 3.7*

$$c_1(K) = f(\operatorname{dist}(s_{\Phi^*(K)}, \Gamma))$$
$$c_2(K) = f(\operatorname{dist}(s_K, \Gamma))$$

and consider the unit square $\bar{\Omega} = [0,1]^2$ *and the implicitly given surface*

$$\Gamma = \{x \in \mathbb{R}^2 : \|x - (0.5, 0.5)^T\|_2 = 0.15\},$$

which is just the boundary of a circle. Let the concentration function $c(K) = (\alpha + \operatorname{dist}(S_K, \Gamma))^\beta$ *be given, with* $\alpha, \beta > 0$ *and use the mesh quality functional from Section 3.4.3.*

This results in a case where $|\operatorname{dist}(s_{\Phi^*(K)}, \Gamma) - \operatorname{dist}(s_K, \Gamma)|$ *is large, as can be seen in Figure 3.17 (ii). A simple Picard iteration computing a sequence* $(\Phi)_j$ *of deformations setting*

$$\Phi_0 = id_\Omega, \quad \forall j > 0 : c(K) = f(\operatorname{dist}(s_{\Phi_{j-1}(K)}, \Gamma))$$

even fails to converge within 50 iterations.

This is an important example of why the additional nonlinearity introduced by making the mesh concentration function depend on the solution needs to be incorporated into the functional, or rather taken into account when computing the derivative.

Some formal differentiation

Assume that we are already in the discrete case, namely we are in the discrete space of admissible variations (see (3.4)). Then we can formally compute the partial derivative of the optimal scale h with regard to the degrees of freedom of the finite element function Φ_h.

$$h(K) = \sqrt[d]{\frac{c(K)}{\sum_{T\in\mathcal{T}_h} c(T)} \sum_{T\in\mathcal{T}_h} \det(\nabla R_{T,n}(id))}$$

$$\Rightarrow \frac{\partial h(K)}{\partial u_i} = \frac{1}{d}\left(\frac{c(K)}{\sum_{T\in\mathcal{T}_h} c(T)} \sum_{T\in\mathcal{T}_h} \det(\nabla R_{T,n}(id))\right)^{\frac{1}{d}-1} \left[\frac{c(K)}{\sum_{T\in\mathcal{T}_h} c(T)} \frac{\partial}{\partial u_i}\left(\sum_{T\in\mathcal{T}_h} \det(\nabla R_{T,n}(id))\right)\right.$$
$$\left. + \frac{\frac{\partial c(K)}{\partial u_i}\sum_{T\in\mathcal{T}_h} c(T) + c(K)\sum_{T\in\mathcal{T}_h}\frac{\partial c(T)}{\partial u_i}}{\left(\sum_{T\in\mathcal{T}_h} c(T)\right)^2}\left(\sum_{T\in\mathcal{T}_h} \det(\nabla R_{T,n}(id))\right)\right],$$

where u_i is a degree of freedom of Φ_h.

Remark 3.18. *The additional dependency of h on the degrees of freedom of a variation Φ does not affect the polyconvexity of the associated stored energy function by the definition of polyconvexity. However, it is necessary to establish some lower bound on h as to still satisfy the prerequisites of the existence theorems in Section 3.5.6.*

3.5.8. Alignment to (implicit) surfaces

As described in Section 3.2.3, there are situations where it is very advantageous to align the mesh with a surface, that might even be given implicitly. Examples are particulate flows (see [BP13] for a method using interface alignment and [MMT12] for an application of the fictitious boundary method), fluid-structure interactions (where interface alignment might even be necessary for the accurate representation of boundary conditions, see e.g. [RTH+12]).

Assume that our domain is $\Omega = \Omega_1 \cup \Omega_2$ and denote by $\Gamma = \bar{\Omega}_1 \cap \bar{\Omega}_2$ the interface between the two subdomains. The mesh $\mathcal{T}_h$ on Ω is assumed to already capture Γ sufficiently well, meaning that Γ is $\delta = \frac{h}{2}$ resolved (see Figure 3.18 (i)).

Recall from Section 3.2.3 the definition of the δ-tube

$$S_\delta(\Gamma) := \{x \in \Omega : \operatorname{dist}(x,\Gamma) < \delta\} \tag{3.39}$$

and with that the decompositions of $\mathcal{T}_h$

$$\mathcal{T}_h^i := \{K \in \mathcal{T}_h : K \cap S_\delta(\Gamma) = \emptyset\},$$
$$\mathcal{T}_* := \mathcal{T}_h \setminus (\mathcal{T}_h^1 \cup \mathcal{T}_h^2),$$
$$\mathcal{T}_*^i := \{K \in \mathcal{T}_h : K \subset (\Omega_i \cup \mathcal{T}_*)\}.$$

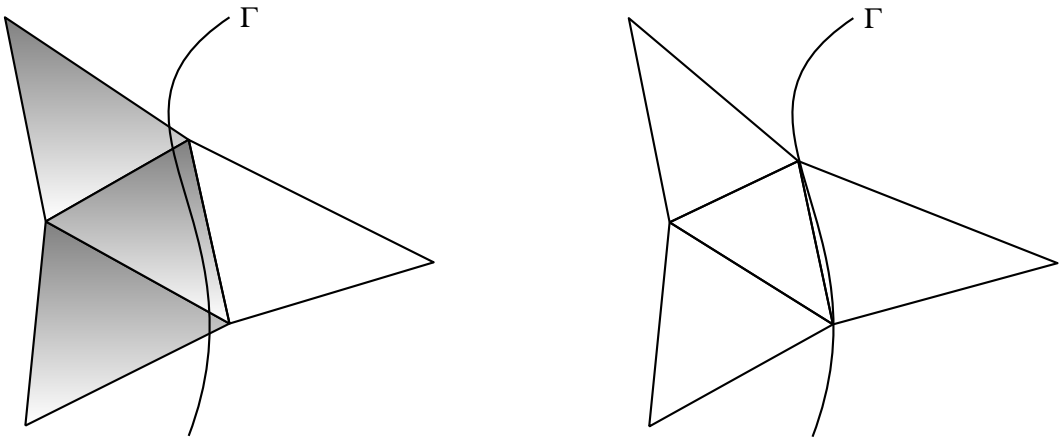
If $\delta \leq \frac{h}{2}$ and Γ is δ-resolved, meaning that $\mathcal{T}_*^1 \cap \mathcal{T}_*^2 = \emptyset$ and $\mathcal{T}_* = \mathcal{T}_*^1 \cup \mathcal{T}_*^2$, define

$$\bar{\Omega}_{h,i} := \{\bigcup_{K\in\mathcal{T}_h^i\cup\mathcal{T}_*^i} \bar{K}\},$$
$$\Gamma_h := \partial\Omega_{h,1}.$$

The parameter δ describes how well the interface Γ is approximated. $\delta \leq \frac{h}{2}$ already means that $\nexists \mathcal{T}_h \ni K \subset \mathcal{T}_*^1 \cap \mathcal{T}_*^2$. Since we use a $\mathbb{P}_1/\mathbb{Q}_1$ transformation, we can expect to $\delta = O(h^2)$-resolve Γ by suitably aligning the mesh's vertices with Γ (see Figure 3.18 (ii)).

We now want to define a constraint $\mathcal{H}$ such that $\mathcal{H}(\Phi) = 0$ implies that $\Phi(\mathcal{T}_h)$ is $O(h^2)$-aligned with Γ. Consider the term

$$\mathcal{H}_1(\Phi) = \sum_{K\in\Phi(\mathcal{T}_h)} \operatorname{vol}(K\cap\Omega_1)\operatorname{vol}(K\cap\Omega_2).$$



(i) Γ is only $\delta = \frac{h}{2}$-resolved by $\mathcal{T}_h$ due to the shaded cells.

(ii) The mesh's vertices have been aligned with Γ, which is now $\delta = O(h^2)$-resolved.

Figure 3.18: Different degrees of alignment of $\mathcal{T}_h$ to Γ.

It is clear that $\mathcal{H}_1 = 0$ only for perfectly aligned meshes, meaning $\delta = 0$. Numerically speaking, terms like $\mathrm{vol}(K \cap \Omega_i)$ are difficult to compute, let alone differentiate with regard to the vertex coordinates. Also note that driving $\mathrm{vol}(K) \to 0$ will also satisfy the constraint, so this would need to be incorporated as well.

Alternatively, we could just take

$$\mathcal{H}_2(\Phi) = \mathrm{vol}(\mathcal{T}^*),$$

with $\mathcal{T}^*$ according to $\Phi(\mathcal{T}_h)$, so we trivially get the notion that for every $\delta \geq 0$ $\mathcal{H}_2(\Phi) = 0$ implies that $\Phi(\mathcal{T}_h)$ is δ-aligned. However, this faces the same difficulties as $\mathcal{H}_1$.

So instead, we will derive a constraint based on a signed distance function s_Γ with $s_\Gamma < 0$ on $\mathring{\Omega}_1$ and $s_\Gamma > 0$ on $\mathring{\Omega}_2$. The idea is then to rule out sign changes of s_Γ along any line connecting two vertices a_i, a_j of a cell K (see [BW13]) by using the constraint

$$\mathcal{H}(\Phi) = \sum_{K \in \Phi(\mathcal{T}_h)} \left(\sum_{a_{i,j} \in \mathcal{E}^0(K), i \neq j} r(s_\Gamma(a_i) s_\Gamma(a_j)) \right), \tag{3.40}$$

with

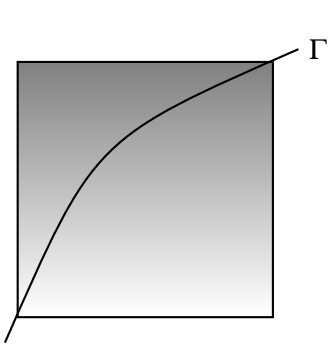
$$\begin{aligned} &r : \mathbb{R} \to \mathbb{R}_{\geq 0}, \quad r \in \mathcal{C}^1(\mathbb{R}) : \\ &\forall x \in \mathbb{R} : r'(x) \geq 0, \quad r(x) \begin{cases} > 0, & x < 0 \\ = 0, & x \geq 0 \end{cases} \end{aligned}$$

Various regularisations of the Heaviside function could be used for the function r,

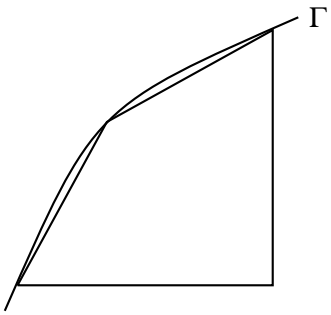
For simplices, every local vertex a_i is connected to all other local vertices a_j by an edge. This is not the case for hypercubes, where the constraint (3.40) introduces "virtual" edges in the interior of K.

However, there are geometric situations where such an alignment condition lets the mesh deteriorate, as even in $2d$, an interior angle might be forced towards π, see Figure 3.19. This is a characteristic of hypercube meshes and cannot be ruled out in advance in a general setting.

There are several ways to overcome this difficulty, which are not discussed further in this work. In $2d$, one approach could be to locally refine a problematic cell to split the angle being forced to



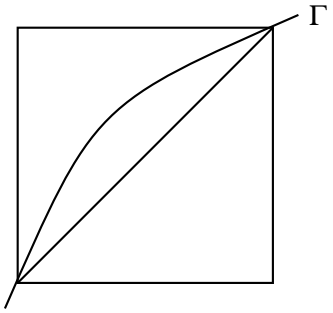
(i) The cell K violates the alignment condition.



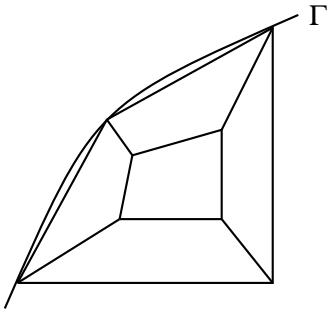
(ii) The cell K does not violate the alignment condition, but is of poor quality.

Figure 3.19: Example of the surface alignment condition (3.40) letting the mesh deteriorate.

π. This involves combinatoric work and the resulting cell still suffer from poor quality. Another approach would be to split the hypercube into simplices in a suitable way. This can be expressed as a local modification of the finite element spaces and could even be done without first aligning vertices to Γ (see [FR14]), although this incurs a serious condition number penalty e.g. for the stiffness matrix.



(i) The cell K violating the alignment condition is split into two triangles.



(ii) The cell K not violating the alignment condition is split into five hypercubes.

Figure 3.20: Example of the surface alignment condition (3.40) letting the mesh deteriorate.

At this point it is not yet clear how to address the constraint (3.40) numerically. One important observation is that the constraint is invariant with regard to vertices moving within Γ, which already means that for a mesh $\mathcal{T}$ fulfilling the alignment condition, the derivatives of the constraint with regard to these movements will be zero. This is a serious drawback which does not seem to be overcome easily.

An additional problem is the regularity assumption $r \in \mathcal{C}^1$ in conjunction with the condition $\forall x \geq 0 : r(x) = 0$, as this necessarily means $r'(0) = 0$ and rules out all the constraint qualifications presented in [Pet73], severely limiting the possibilities so solve the constrained minimisation problem numerically. It might be possible to devise a formulation of the surface alignment constraint that is easier to treat in practice, but this is an open problem to the best of my knowledge.

Remark 3.19. *When using higher order transformations ($\mathbb{P}_k, \mathbb{Q}_k$, $k > 1$), it is still advantageous to first use the condition* (3.40) *and then treat the additional DoF. See [Wei12] for details in the case of simplices, $d = 2$ and $k = 2$.*

4

Numerical methods for hyperelasticity problems

In this chapter I want to describe how to discretise the system of nonlinear equations arising from the minimisation of the hyperelasticity-based mesh quality functional. The discretisation is given in Section 4.1, which is more of a summary of points already discussed. The finite element spaces are given in Section 4.1.1, but since we are not interested in how well the discrete solution approximates the continuous problem, I will not give any error analysis. Much of the theory can be found in [NW06], although I also recommend [BKST15] for some details on nonlinear multilevel-based solvers and [GK09b], [GK09a] for multilevel trust-region solvers.

4.1. Discretisation

In this part, we want to apply the theoretical background from Section 3.5.6, so we pose the following general assumptions.

Assume $\Omega \subset \mathbb{R}^d$ to be a given domain such that

$$\partial\Omega \supset \Gamma_0, \Gamma_1, \Gamma_2, \mathrm{vol}_{d-1}(\partial\Omega \setminus \{\Gamma_0 \cup \Gamma_1 \cup \Gamma_2\}) = 0.$$

Let Γ_i be $d\sigma$-measurable, relatively open and disjoint. Assume further that we have $\mathcal{L} : \Omega \times \mathrm{SL}_d$ with the same properties as in Theorem 3.11:

i) **Polyconvexity:**

$$\forall x \in \Omega \ a.e. : \exists \tilde{L}(x,\cdot,\cdot,\cdot) : \mathrm{SL}_3 \times \mathrm{SL}_3 \times (0,\infty) \to \mathbb{R} : \forall F \in \mathrm{SL}_3 : \\ \mathcal{L}(x,F) = \tilde{L}(x,F,\mathrm{Cof}(F),\det(F)),$$

where $\forall (F,H,\delta) \in \mathrm{SL}_3 \times \mathrm{SL}_3 \times (0,\infty) : L(\cdot,F,H,\delta) \in L^1(\Omega)$.

ii) **Stability:**

$$\forall x \in \Omega \ a.e : \lim_{\det(F)\to 0} \mathcal{L}(x,F) = +\infty$$

iii) **Coerciveness:**

$$\begin{aligned}\exists \alpha \in \mathbb{R}_+, \beta \in \mathbb{R}, 2 \leq p \in \mathbb{N}, \frac{p}{p-1} \leq q \in \mathbb{N}, 1 < r \in \mathbb{R} : \\ \forall x \in \Omega \ a.e., \forall F \in \mathrm{SL}_3 : \\ \mathcal{L}(x,F) \geq \alpha(\|F\|_F^p + \|\operatorname{Cof}(F)\|_F^q + \det(F)^r) + \beta\end{aligned}$$

Let further $\Gamma \subset \mathbb{R}^d$ be closed and $\phi_0 \in L^1(\Gamma_0, \mathbb{R}^d)$ such that

$$\begin{aligned}\emptyset \neq \mathcal{D}_{\phi_0,\Gamma} := \{\Phi \in W^{1,p}(\Omega) : &\operatorname{Cof}(\nabla\Phi) \in L^q(\Omega), \det(\nabla\Phi) \in L^r(\Omega), \\ &\forall x \in \Omega \ a.e. : \det(\nabla\Phi)(x) > 0, \\ &\forall x \in \Omega \ a.e. : \Lambda(\nabla\Phi(x)) \leq 0, \\ &\forall x \in \Gamma_0 \ d\sigma \ a.e. : \Phi(x) = \phi_0(x) \\ &\forall x \in \Gamma_2 \ d\sigma \ a.e. : \Phi(x) \in \Gamma_2\}\end{aligned}$$

Note that any of the sets Γ_i might be empty, if there is no part of the boundary where we want to enforce the corresponding boundary condition. We also assume that $\mathcal{L}$ is *isotropic* and *frame indifference*, so that the Rivlin-Erikson Representation Theorem gives that

$$\begin{aligned}\exists L : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R} \cup \{\infty\} \\ \forall A \in \mathrm{SL}_d : \mathcal{L}(x,A) = L(\|A\|_F^2, \|\operatorname{Cof}(A)\|_F^2, \det(A))\end{aligned}$$

Under the assumption that $\exists \Phi \in \mathcal{D}_{\phi_0,\Gamma} : \mathcal{F}(\Phi) < +\infty$, Theorem 3.13 gives there exists at least one

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

Also see [Rum96, Theorem 1] for a proof for simplex meshes in $3d$.

4.1.1. Finite Element spaces

Instead of solving the problem in the space $\mathcal{D}_{\phi_0,\Gamma}$, we now use the discrete space $\mathcal{D}_{\phi_0,\Gamma} \cap \mathcal{D}_h$.

Recall the definition of the discrete spaces of admissible variations (Definition 3.2) on a domain $\hat{\Omega}$:

$$\mathcal{D}_h := \{\Phi \in \tilde{\mathcal{D}}_{bp,h} : \forall m = 0, \ldots, d-1 : \forall \Gamma \in \partial\Omega_h^m : \forall x \in \Gamma : \Phi(x) \in \Gamma\}$$

which incorporate the property of being orientation preserving formalised by $\det(\nabla\Phi_h) > 0$. Like the divergence-free condition in the Stokes problem, this property is not trivial to incorporate into standard finite element spaces. Since the stored energy function has the stability property $\lim_{\det(F)\to 0} \mathcal{L}(x,F) = +\infty$, a suitable numerical method for minimising $\mathcal{F}$ should keep the orientation intact.

Displacement boundary conditions of the type

$$\forall x \in \Gamma_0 \ d\sigma \ a.e. : \Phi(x) = \phi_0(x)$$

can easily be enforced (in this case even in the strong sense) through a projection operator $\mathcal{P}_{\phi_0}$.

Enforcing the unilateral boundary condition of place

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

is more difficult, as it involves a projection $\mathcal{P}_\Gamma$ of a point $\Phi(x)$ to the boundary Γ which must not violate any of the other constraints for the space $\mathcal{D}_{\phi_0,\Gamma} \cap \mathcal{D}_h$, such as the orientation preservation, and must not lead to self-intersections at the boundary Γ.

4.1.2. Discretised functional and its gradient

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

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

[Cia88, Theorem 4.1-1, Theorem 4.1-2] gives that the minimisation of this functional is formally equivalent to the equations

$$\forall \eta \in \mathcal{D}_{0,\Gamma} : \mathcal{F}_h'(\Phi^*)\eta = 0,$$

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

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

Remark 4.1.

i) 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.

ii) Because

$$\frac{\partial \det(\nabla\Phi(x))}{\partial A} = \det(\nabla\Phi(x))(\nabla\Phi(x))^{-T} : \nabla\eta(x),$$

the usual approach for evaluating integrals of using numerical integration does not seem feasible, as we are dealing with rational functions.

iii) This also means that working with hypercube meshes is significantly more costly that working with simplex meshes. For simplex meshes, $\forall K \in \mathcal{E}^d(\mathcal{T}_h) : (\nabla\Phi_h)_{|K}(x) \equiv const$, so that no real numerical integration has to be performed for evaluating the local integrals $\int_K \mathcal{L}_K(x,\Phi_h)dx$. For hypercube meshes, the general case is the one of nonlinear transformations, so numerical integration with a formula of sufficient order or exact integration (for the rational functions) has to be used. This increases the cost of a single functional or gradient evaluation by a factor of # integration points used.

4.2. Methods for the unconstrained minimisation of nonconvex functionals

In this section I want to introduce algorithms for the minimisation of nonlinear, nonconvex functionals on finite dimensional spaces. This is the class of methods we need for minimising the discrete mesh quality functionals from Section 4.1.2.

If $f : U \to \mathbb{R}^n$ is of class $\mathcal{C}^{k+1}$, define the kth Taylor polynomial by

$$T_k f(x,x_0) := \sum_{|\alpha|\le k} \frac{1}{\alpha!} D^\alpha f(x_0)(x-x_0)^\alpha$$

and the $(k+1)$ rest by

$$R_{k+1,\xi} f(x,x_0) := \sum_{|\alpha|=k+1} \frac{1}{\alpha!} D^\alpha f(\xi)(x-x_0)^\alpha.$$

i) If $x, x_0 \in U$ such that for all $[x_0;x] := \{\lambda x + (1-\lambda)x_0, \lambda \in [0,1]\} \subset U$, there exists $\xi \in [x;x_0]$ such that

$$f(x) = T_k f(x,x_0) + R_{k+1,\xi} f(x,x_0).$$

ii) For any $x_0 \in U$ and any sequence $x \to x_0$ in U

$$f(x) = T_k f(x,x_0) + o(\|x-x_0\|_2^p).$$

The proof for this can be found in most elementary analysis textbooks. For solving the (nonlinear) equation $f(x) = 0$, *Newton's method* based on approximating f locally by its first Taylor polynomial is very popular.

Lemma 4.2. *Newton's method*

Let $U \subset \mathbb{R}^n$ be open, $U_0 \subset U$ convex such that $\bar{U}_0 \subset U$. Let $f \in \mathcal{C}^0(U, \mathbb{R}^n)$, $x_0 \in U_0$ and $\alpha, \beta, \gamma \in \mathbb{R}$ such that

i) $f' \in \mathcal{C}^{0,1}(U_0, \mathbb{R}^n)$ with $\forall x, y \in U_0 : \|f'(x) - f'(y)\| \leq \gamma \|x-y\|$,

ii) $\forall x \in U_0 : \exists (f'(x))^{-1}$ and $\|(f'(x))^{-1}\| \leq \beta$,

iii) $\|(f'(x_0))^{-1}\| \leq \alpha$,

with the constants α, β, γ additionally satisfying

$$\frac{\alpha\beta\gamma}{2} < 1,$$
$$B_r(x_0) \subseteq U_0, \text{ where } r := \frac{2\alpha}{2 - \alpha\beta\gamma}$$

Denote $h := \frac{\alpha\beta\gamma}{2}$. Then

i) For $x^{(0)} = x_0$ the sequence of Newton iterates

$$x^{(k+1)} = x^{(k)} - f'(x^{(k)})^{-1} f(x^{(k)}), k = 0, 1, \ldots \tag{4.1}$$

is well-defined and $\forall k \in \mathbb{N} : x^{(k)} \in B_r(x^{(0)})$,

ii) The limit $\lim_{k\to\infty} x^{(k)} =: x^*$ exists, $x^* \in \overline{B_r(x^{(0)})}$, $f(x^*) = 0$ and

iii)

$$\forall k \in \mathbb{N}_0 : \|x^* - x^{(k+1)}\| \leq \alpha \frac{h^{2^k - 1}}{1 - h^{2^k}}. \tag{4.2}$$

Proof. See [Sto04, Section 5.3]. □

Remark 4.3. *Drawbacks of Newton's method*

If $\mathcal{F}$ is sufficiently smooth, we could solve our nonlinear equation $\mathcal{F}'(\Phi^*) = 0$ on the finite dimensional space $\mathcal{D}_h$ with Newton's method (replace f by $\mathcal{F}'$ and x by Φ in Lemma 4.2 above). However, there are three limiting factors.

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

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

$$\mathcal{F}''(\Phi^{(k)})d^{(k)} = -\mathcal{F}(\Phi^{(k)}) \tag{4.3}$$

is not even a descent direction.

iii) In our case, the Hessian $\mathcal{F}''(\Phi^{(k)})$ 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 (4.3)*. However, for other problems with less pronounced nonlinearity, iterative methods have been applied successfully for solving* (4.3) *(see [BKST15] for some examples), even if the true Hessian is not available and has to be approximated using a divided difference scheme [DHOT13].*

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

i) Trust region methods: Minimising a quadratic model and rejecting steps if the difference between predicted and true functional value is too large ([NW06, Chapter 4]).

ii) Line search methods: Successive minimisation in search directions (Section 4.2.2).

iii) Quasi Newton methods: Recursive updating of an approximation of the inverse Hessian (Section 4.2.1).

iv) Inexact or truncated Newton methods: (4.3) is solved only approximately, e.g. by using an iterative solver and terminating early and/or when the Hessian is indefinite. ([NW06, Chapter 7]).

All these methods are closely related (e.g. we will see in Section 4.2.2 that quasi and inexact Newton methods are special preconditioners for line search methods) and have been combined with each other in about every possible combination. In the following, I will touch quasi Newton methods just briefly and then proceed to line search methods. This has the benefit of being able to derive and use PDE-based preconditioners, instead of using the (more or less) black box preconditioners given by the various quasi Newton methods, which are more algebraic in nature.

4.2.1. Quasi Newton methods

One idea to overcome the main drawbacks of Newton's method (Remark 4.3) is to replace the Hessian $\mathcal{F}''(\Phi^{(k)})$ by matrices $H^{(k)}$. In this section, I will only introduce some general ideas, as the lBFGS-preconditioned nonlinear steepest descent method (NLSD-lBFGS) will be used later for comparisons, but I will spend more time on general line search methods.

Definition 4.1. *Quasi Newton methods*
If the relation

$$H^{(k+1)}(\mathcal{F}'(\Phi^{(k+1)} - \Phi^{(k)})) = \Phi^{(k+1)} - \Phi^{(k)} \tag{4.4}$$

is satisfied, the search direction $d^{(k+1)}$ defined as the solution of

$$H^{(k+1)}d^{(k)} = -\mathcal{F}(\Phi^{(k)})$$

is called quasi Newton direction *and the resulting method is a* quasi Newton method*.*

Remark 4.4.

i) If $H^{(k)}$ is symmetric and positive definite, sometimes the term variable metric method *is used instead of quasi Newton method. This is due to the fact that every $H^{(k)}$ induces a scalar product inducing a metric which we use for minimisation purposes, but this metric changes in every iteration.*

ii) If $H^{(k+1)}$ is symmetric and positive definite, the quasi Newton direction is a descent direction, so these are essential properties.

Definition 4.2. *Broyden class*

Denote $\left(H^{(k)}\right)^{-1} =: B^{(k)}$. Iff the quasi Newton direction is a descent direction, it holds that

$$0 < (\mathcal{F}'(\Phi^{(k)}), d^{(k)}) = (\mathcal{F}'(\Phi^{(k)}), B^{(k)}\mathcal{F}'(\Phi^{(k)})).$$

Set $p^{(k)} := \Phi^{(k+1)} - \Phi^{(k)}$ and $q^{(k)} := \mathcal{F}'(\Phi^{(k+1)}) - \mathcal{F}'(\Phi^{(k)})$ and then for some $\theta_k > 0$

$$\begin{aligned} B^{(k+1)} = B^{(k)} &+ \left(1 + \theta_k \frac{(q^{(k)}, B^{(k)} q^{(k)})}{(p^{(k)}, q^{(k)})}\right) \frac{1}{(p^{(k)}, q^{(k)})} p^{(k)} \left(p^{(k)}\right)^T \\ &- \frac{1-\theta_k}{(q^{(k)}, r^{(k)} q^{(k)})} (B^{(k)} q^{(k)}) \cdot \left(B^{(k)} q^{(k)}\right)^T \\ &- \frac{\theta_k}{(p^{(k)}, q^{(k)})} p^{(k)} \left(\left(q^{(k)}\right)^T B^{(k+1)} + B^{(k)} q^{(k)} \left(p^{(k)}\right)^T\right). \end{aligned}$$

This formula defines the Broyden class *of updates for $B^{(k)}$. There are some important special cases:*

i) $\forall k \in \mathbb{N} : \theta_k = 0$: Davidon, Fletcher, Powell (DFP) update ([Dav59], [FP63]). This was the first known efficient update formula.

ii) $\forall k \in \mathbb{N} : \theta_k = 1$: Broyden, Fletcher, Goldfarb, Shanno (BFGS) update ([Bro70]). Very often, this is considered the most effective Broyden class update formula [NW06, Chapter 6].

iii) $\forall k \in \mathbb{N} : \theta_k = (q^{(k)}, p^{(k)})/((q^{(k)}, p^{(k)}) - (p^{(k)}, H^{(k)} q^{(k)})$: Symmetric rank one update. This does not carry the implication that, if $H^{(k)}$ is positive definite, $H^{(k+1)}$ is positive definite, too.

Remark 4.5.

i) It is easy to see that in a Broyden class update, a matrix of rank 2 is added to $H^{(k)}$ to obtain $H^{(k+1)}$.

ii) Very often, the initial approximation $B^{(0)}$ of the Hessian is chosen to be β id with some $\beta > 0$, or as a diagonal matrix. Other choices are known, but are equally heuristic in nature.

iii) Since $B^{(k)}$ will be dense in general, it cannot be computed and stored explicitly in many applications. For the BFGS update it is possible to formulate the application of $B^{(k)}$ to a vector through a recursion using the vectors $p^0, \ldots, p^{(k-1)}$ and $q^{(0)}, q^{(k-1)}$. By keeping only a small number m of these vectors (e.g. $m \in \{3, \ldots, 20\}$), one can construct an approximation to $B^{(k)}$ (limited memory BFGS update (lBFGS)), that works well in practice, although it may converge slowly on ill-conditioned problems and no theory is available. [NW06, Chapter 7.2].

Although there are theoretical results for Broyden class updates for (at least locally) strongly convex $\mathcal{F}$, proving local superlinear convergence, there is no proof for more general nonlinear functionals $\mathcal{F}$. [NW06]

Because of the lack of theoretical results for the class of problems we want to examine, I will turn to line search based methods next. In practice, both families of methods performed quite similar in the test cases. As we will see in Section 4.2.2, the construction of the approximation $B^{(k)}$ to the Hessian $\mathcal{F}''(\Phi^{(k)})$ can be regarded as choosing a preconditioner for a general descent method.

4.2.2. Line search methods

This section will only give some rough ideas. Refer to [NW06, Chapter 3, Chapter 4] for an extensive presentation.

In this subsection, assume that $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ and that for a given initial guess $\Phi^{(0)} = \Phi_0$ that

$$U_0 := \{\Phi : \mathcal{F}(\Phi) \leq \mathcal{F}(\Phi_0)\} \text{ is compact and} \tag{4.5}$$

$$\exists U_0 \supset U \text{ convex and open} : \mathcal{F} : \mathcal{C}^1(U, \mathbb{R}) \tag{4.6}$$

For $x \in U$, $d \in \mathbb{R}^n$ is called a *descent direction* iff $(d, \mathcal{F}'(x)) < 0$. Then it follows that

$$\exists \alpha > 0 : \mathcal{F}(\Phi + \alpha d) < \mathcal{F}(\Phi).$$

With this, a general descent method is then of the following form:

Algorithm 4.1 General descent algorithm.

For a given Φ_0 and initial descent direction $d^{(0)}$ do $k = 0, \ldots, N$:

1. Compute a step length $\alpha^{(k)}$ such that $\mathcal{F}(\Phi^{(k)} + \alpha d^{(k)}) < \mathcal{F}(\Phi^(k))$.
2. Set $\Phi^{(k+1)} = \Phi^{(k)} + \alpha d^{(k)}$.
3. Compute a new descent direction $d^{(k+1)}$.

In this general form, Algorithm 4.1 will not produce a sequence of iterates that converges to a local minimum. Without any further conditions, the *line search* in step 1 might produce arbitrarily short steps $\alpha^{(k)}$, or the search directions might become nearly orthogonal to the gradient $\mathcal{F}'(\Phi)$. These two issues will need to be taken care of.

Definition 4.3. *Let an iterate Φ and a search direction d be given.*

i) A line search is called exact *iff it returns* $\alpha^* = \operatorname{argmin}_{\alpha \in \mathbb{R}_+} \mathcal{F}(\Phi + \alpha d)$.

ii) A step length α is said to satisfy the Armijo condition *with constant $c_1 \in (0,1)$ iff*

$$\mathcal{F}(\Phi + \alpha d) \leq \mathcal{F}(\Phi) + c_1 \alpha (\mathcal{F}(\Phi), d). \tag{4.7}$$

iii) A step length α is said to satisfy the strong Wolfe conditions *with constant $c_{1,2} \in (0,1)$, $c_1 < c_2$ iff it satisfies the Armijo condition with constant c_1 and*

$$\left|(\mathcal{F}'(\Phi + \alpha d), d)\right| \leq c_2 |(\mathcal{F}'(\Phi), d)|. \tag{4.8}$$

Performing an exact line search will usually decrease the number of iterations the outer general descent method needs, but it will require very many (costly) evaluations of $\mathcal{F}$ and/or $\mathcal{F}'$. An inexact line search addresses this problem, but needs to be made "exact enough" by requiring that some conditions hold, like the strong Wolfe conditions above. There are line searches based on just the Armijo condition, or even the Goldstein conditions ([NW06, Chapter 3.1]) which are similar to the Wolfe conditions.

Lemma 4.6. *Zoutendijk condition*

Assume that $\exists c_l \in \mathbb{R} : \mathcal{F} > c_l$ and consider Algorithm 4.1, assuming that $\forall k \in \mathbb{N} : d^{(k)}$ is a descent direction and $\alpha^{(k)}$ is a step length satisfying the strong Wolfe conditions. If $\mathcal{F}' \in \mathcal{C}^{0,1}(U, \mathbb{R}^n)$, the Zoutendijk condition

$$\sum_{k=0}^{\infty} \frac{\left(-\mathcal{F}'(\Phi^{(k)}), d^{(k)}\right)}{\|d^{(k)}\|_2} < +\infty \tag{4.9}$$

holds.

Proof. See [NW06, Theorem 3.2]. □

The Zoutendijk condition formalises the requirement that all $d^{(k)}$ are descent directions that do not become orthogonal to the gradient $\mathcal{F}'(\Phi^{(k)})$ and can be used to prove global convergence results for general descent methods.

Theorem 4.7. *Convergence of general descent methods*

Assume the assumptions from Lemma 4.6 hold and that the descent directions $d^{(k)}$ additionally satisfy

$$\exists c \in \mathbb{R}^+ : \forall k \in \mathbb{N} : \beta_k := \frac{\left(-\mathcal{F}'(\Phi^{(k)}), d^{(k)}\right)}{\|\mathcal{F}'(\Phi^{(k)})\|_2 \|d^{(k)}\|_2} \geq c > 0. \tag{4.10}$$

If the line search step sizes $\alpha^{(k)}$ satisfy the strong Wolfe conditions (4.7)*,* (4.8) *and the algorithm does not terminate after a finite number of steps, it follows that*

$$\lim_{k \to \infty} \mathcal{F}'(\Phi^{(k)}) = 0.$$

For every accumulation point Φ^* of the sequence, it holds that $\mathcal{F}'(\Phi^*) = 0$. Since the set U_0 is is compact by assumption, the sequence $\left(\Phi^{(k)}\right)_{k \in \mathbb{N}}$ trivially has at least one accumulation point Φ^*.

Proof. The Zoutendijk condition holds and we have

$$+\infty > \sum_{k=0}^{\infty} \frac{\left(-\mathcal{F}'(\Phi^{(k)}), d^{(k)}\right)}{\|d^{(k)}\|_2} = \sum_{k=0}^{\infty} \|\mathcal{F}'(\Phi^{(k)})\|_2 \beta_k > c \sum_{k=0}^{\infty} \|\mathcal{F}'(\Phi^{(k)})\|_2,$$

which implies $\lim_{k \to \infty} \|\mathcal{F}'(\Phi^{(k)})\|_2 = 0$.

Since $\mathcal{F}$ is continuous and U_0 is compact by assumption (4.5), so $\left(\Phi^{(k)}\right)_{k \in \mathbb{N}}$ has at least one accumulation point. If Φ^* is an accumulation point (since $\mathcal{F}'$ is continuous) we obtain for a converging subsequence $(\Phi^{(m_k)})_{k \in \mathbb{N}}$

$$\mathcal{F}'(\Phi^*) = \mathcal{F}'(\lim_{k \to \infty} \Phi^{(m_k)}) = \lim_{k \to \infty} \mathcal{F}'(\Phi^{(m_k)}).$$

□

Remark 4.8.

i) The prerequisites of Lemma 4.6 and Theorem 4.7 are not too restrictive, since $\mathcal{F}$ needs to be bounded from below for the minimisation problem to be well-defined, and the Lipschitz continuity of $\mathcal{F}'$ is often satisfied in practice.

ii) Theorem 4.7 gives a very strong convergence result that is global *in the sense that the initial guess $\Phi^{(0)}$ does not need to be "close" to a stationary point Φ^* for Algorithm 4.1 to converge. But the downside is that convergence can become arbitrarily slow. As opposed to that, Newton's method (see Lemma 4.2) gives a very strong result and fast convergence, but only* locally.

iii) One important point in Theorem 4.7 is that the line search does not have to be exact, as long as every step size fulfils the strong Wolfe conditions, as this is much easier numerically. For theoretical results, it very often remains necessary to assume exact line searches.

iv) Important variants of Algorithm 4.1 are the (nonlinear) steepest descent method *(NLSD) obtained by choosing*

$$d^{(k)} = -1/\|\mathcal{F}'(\Phi^{(k)})\|_2 \mathcal{F}'(\Phi^{(k)}),$$

the (nonlinear) conjugate gradient method *(NLCG) obtained by setting*

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

where β is set according to a conjugacy condition, see Algorithm 4.2, and various variants of Newton's method.

v) In Newton's method, we have $\forall k \in \mathbb{N} : \alpha^{(k)} = 1$ and set the search direction as

$$d^{(k)} = -\left(\mathcal{F}''(\Phi^{(k)})\right)^{-1} \mathcal{F}'(\Phi^{(k)}).$$

The conditions in Lemma 4.2 ensure that $d^{(k)}$ is a descent direction, which is only possible if we start close enough to a stationary point Φ^*. It is possible to replace the fixed step size $\alpha^{(k)} = 1$ by a line search (sometimes called damped Newton's method*).*

For a quasi Newton method *using a matrix $\left(H^{(k)}\right)^{-1}$ to approximate $\left(\mathcal{F}''(\Phi^{(k)})\right)^{-1}$ for computing the search direction, one can show that iff*

$$\lim_{k\to\infty} \frac{\| H^{(k)} - \mathcal{F}''(\Phi^{(k)}) d^{(k)} \|}{\|d^{(k)}\|} = 0$$

and some other assumptions hold, the resulting quasi Newton method converges superlinear locally *(see [NW06, Theorem 3.6]).*

vi) It is possible to precondition *descent methods by applying the inverse of a symmetric positive definite matrix $H^{(k)}$ to the descent direction $d^{(k)}$. We immediately see that choosing $H^{(k)} = \mathcal{F}''(\Phi^{(k)})$ in the NLSD method, we recover a damped Newton method, or even a quasi Newton method if $B^{(k)}$ is a "good" approximation of the Hessian (e.g. using the lBFGS update). In the following, I will call this NLSD-lBFGS.*

The already mentioned important special case of the NLCG algorithm was introduced in [FR64]. It is a variant of the well known linear Conjugate Gradient method (see [HS52]). The main differences are:

i) $\mathcal{F}$ 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.

ii) In the linear case, the conjugacy of the search directions is in the sense that $d^{(k+1)}, d^{(k)}$ are $\mathcal{F}''$-orthogonal. In the nonlinear case, the notion of conjugacy changes in every iteration, as $\mathcal{F}'' = \mathcal{F}''(\Phi)$ 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 4.9.

Algorithm 4.2 Preconditioned nonlinear Conjugate Gradient (NLCG) algorithm.

For a given Φ_0, preconditioner $B : \mathbb{R}^d \to \mathbb{R}^d$ and initial search direction $d^{(0)}$ do $k = 0, \dots, N$:

1. Compute a step length $\alpha^{(k)}$ such that $\mathcal{F}(\Phi^{(k)} + \alpha d^{(k)}) < \mathcal{F}(\Phi)$.
2. Set $\Phi^{k+1} = \Phi^{(k)} + \alpha d^{(k)}$.
3. Compute a new descent direction

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

Remark 4.9. *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 $\mathcal{F}$ is quadratic (and the nonlinear CG method reduces to the linear version). All variants have some influence on the convergence behaviour.*

i) The Hestenes-Stiefel update [HS52] sets

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

ii) The Fletcher-Reeves update [FR64] sets

$$\beta_{FR}^{(k+1)} = \frac{\left(B^{-1}\mathcal{F}'(\Phi^{(k+1)}), \mathcal{F}'(\Phi^{(k+1)})\right)}{\|\mathcal{F}'(\Phi^{(k)})\|}.$$

iii) The Polak-Ribière-Polyak update [PR69, Pol69] sets

$$\tilde{\beta}_{PRP}^{(k+1)} = \frac{\left(B^{-1}\mathcal{F}'(\Phi^{(k+1)}), \mathcal{F}'(\Phi^{(k+1)}) - \mathcal{F}'(\Phi^{(k)})\right)}{\|\mathcal{F}'(\Phi^{(k)})\|}.$$

As this update does not guarantee that $d^{(k+1)}$ is a descent direction, the modified version

$$\beta_{PRP}^{(k+1)} = \max\{0, \tilde{\beta}_{PRP}^{(k+1)}\}$$

is used most of the time.

iv) The Dai-Yuan update [DY99] sets

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

v) The Dai-Yuan-Hestenes-Stiefel hybrid update sets

$$\beta_{DYHS}^{(k+1)} = \max\{0, \min\{\beta_{DY}^{(k+1)}, \beta_{HS}^{(k+1)}\}\}.$$

vi) The Hager-Zhang update [HZ05] sets

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

See the survey article [HZ06] for even more search direction updates.

Remark 4.10. *The preconditioner B*

In literature dealing with nonlinear optimisation, the notion of preconditioning NLCG rarely appears (see [HZ06, Section 8] for one of the exceptions). This may be due to (algebraic) preconditioners being available for NLSD that offer good convergence properties in practice (e.g. lBFGS) 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 4.9 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 Polak-Ribière-Polyak update should read

$$\tilde{\beta}_{PRP}^{(k+1)} = \frac{\left(\left(B^{(k+1)}\right)^{-\frac{1}{2}} \mathcal{F}'(\Phi^{(k+1)}), \left(B^{(k+1)}\right)^{-\frac{1}{2}} \mathcal{F}'(\Phi^{(k+1)}) - \left(B^{(k)}\right)^{-\frac{1}{2}} \mathcal{F}'(\Phi^{(k)}) \right)}{\|\mathcal{F}'(\Phi^{(k)})\|},$$

which 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 4.9 iii for left-only preconditioning.

The remaining question is if this offers any benefit over e.g. simply preconditioning NLSD with $B^{(k)}$. This is an open question, as there are examples where it is not the case, which is illustrated briefly in Section 4.4.

Remark 4.11. *Optimisation in Hilbert spaces*

The presented methods can also be formulated directly in Hilbert spaces (see e.g. [Sac86]), meaning they can even be applied to the continuous problem without first discretising the functional by the finite element method. This is beyond the scope of this work, as mesh optimisation is only relevant in the discrete case, but additional insight might be gained from considering the problem in a more abstract setting.

4.3. Incorporating constraints

If we now want to align the mesh $\mathcal{T}_h$ with a surface Γ, we effectively add a *constraint* to the minimisation problem:

$$\text{Find } \Phi^* = \text{argmin}_{\Phi \in \mathcal{D}_{\phi_0}} \mathcal{F}(\Phi) \quad \text{satisfying } \forall i \in E : c_i(\Phi^*) = 0, \tag{4.11}$$

where $E \subset \mathbb{N}$ is a finite set defining the number of constraints. For such a *constrained minimisation problem* to be well-posed, the constraints c_i will need to satisfy certain conditions. This is usually called *constraint qualification* (see [NW06, Chapter 12]). One example is the the *linear independence constraint qualification* (LICQ).

Remark 4.12. *In this part, I only mention equality constraints, as this is the relevant case for the class of problems we want to consider. Most of the theory mentioned in this section also takes inequality constraints into account, which are much harder to treat.*

Definition 4.4. *The linear independence constraint qualification is said to hold at Φ iff*

$$\{\nabla c_i(\Phi) : i \in E\} \text{ is linearly independent.}$$

Together with the first order necessary conditions for an unconstrained minimum, a constraint qualification forms the first order necessary conditions for the constrained minimisation problem (4.11). One example are the Karush-Kuhn-Tucker (KKT) conditions, using the LICQ ([NW06, Theorem 12.1]). There are many other constraint qualifications, which are usually related, can be derived from or implicate each other, see [Pet73] for an extensive treatment. All of the constraint qualifications from [Pet73] have in common that they require $\nabla c(\Phi^*) \neq 0$ in one variant or the other. I do not cover more details because we will see that the surface alignment constraint from Section 3.5.8 does not have this property.

Recall (3.40):

$$\mathcal{H}(\Phi) = \sum_{K \in \Phi(\mathcal{T}_h)} \left(\sum_{a_{i,j} \in \mathcal{E}^0(K), i \neq j} r(s_\Gamma(a_i) s_\Gamma(a_j)) \right),$$

with

$$\begin{aligned} &r : \mathbb{R} \to \mathbb{R}_{\geq 0}, \quad r \in \mathcal{C}^1(\mathbb{R}) : \\ &\forall x \in \mathbb{R} : r'(x) \geq 0, \quad r(x) \begin{cases} > 0, & x < 0 \\ = 0, & x \geq 0 \end{cases}. \end{aligned}$$

Because of the regularity assumption, we necessarily have $r'(0) = 0$ which already implies

$$\mathcal{H}(\Phi^*) = 0 \Rightarrow \nabla \mathcal{H}(\Phi^*) = 0.$$

The condition $\forall x \geq 0 : r(x) = 0$ is strictly necessary because away from Γ, the constraint should not influence the solution.

This means that most methods for solving constrained minimisation problems are not applicable. One method that does not require any form of constraint qualification is the quadratic penalty method ([NW06, Chapter 17.1]). For this, we define the quadratic penalty function

$$Q(\Phi, \mu) := \mathcal{F}(\Phi) + \frac{\mu}{2} \sum_{i \in E} c_i^2(\Phi) \tag{4.12}$$

which in our case reduces to

$$Q(\Phi, \mu) = \mathcal{F}(\Phi) + \frac{\mu}{2} \mathcal{H}(\Phi)^2.$$

μ is the *penalty parameter*. The idea is now to solve a sequence of minimisation problems $(Q(\mathcal{F}, \mu_n))_{n \in \mathbb{N}}$ with

$$\forall i > k : \mu_i > \mu_k, \mu_k \xrightarrow{k \to \infty} +\infty.$$

obtaining a sequence of minimisers $\Phi^{*(n)}$, so that $\lim_{n \to \infty} \mathcal{H}(\Phi^{*(n)}) = 0$. Because of the regularity of the penalty terms, we can use the described unconstrained minimisation techniques from Section 4.2 to compute

$$\Phi^{*(n)} = \operatorname{argmin}_{\Phi \in \mathcal{D}_{\phi_0}} Q(\Phi, \mu_n)$$

with the hope that $\Phi^{*(n-1)}$ proves a good initial guess so that the number of iterations for the unconstrained minimisation is small.

In [Wei12, Section 6.1.3], two different functions r_1, r_2 are compared with

$$r_1(x) = \begin{cases} x^2, & x < 0 \\ 0, & x \geq 0 \end{cases}, \quad r_2(x) = \begin{cases} 2(\cosh(x) - 1), & x < 0 \\ 0, & x \geq 0 \end{cases}.$$

Although they differ only by $O(x^4)$ for $x \to 0$, the numerical results suggest that r_2 is better suited for our purposes, as the convergence of both the unconstrained minimisation algorithms and the constraint $\mathcal{H}(\Phi^{*(n)})$ is faster.

Remark 4.13. *The choice of the sequence $(\mu_n)_{n\in\mathbb{N}}$ is critical for the overall performance of the quadratic penalty method. If μ_n is increased too slowly, many outer iterations may be necessary to get a sufficient decrease in $\mathcal{H}(\Phi^{*(n)})$, which means we need to solve many unconstrained minimisation problems. If μ_n is increased to quickly, $\Phi^{*(n-1)}$ is not a good initial guess for the minimisation of $Q(\Phi, \mu_n)$, so the unconstrained minimisation will require many iterations or even fail to converge all together, as the penalisation leads to systematic ill-conditioning of the Hessian $Q''(\Phi, \mu)$.*

In practice, a strategy that works well is setting

$$\mu_1 = 1, \quad \mu_{n+1} = c \left(\mu_n \frac{\mathcal{H}(\Phi^{*(n)})}{\mathcal{H}(\Phi^{*(n-1)})} \right)^2 \tag{4.13}$$

where e.g. $c = 5$. See also [Wei12, Algorithm 6.1], [BW13, Algorithm 1].

4.4. Examples of nonlinear minimisation algorithms

There are some well-known test functions for nonlinear minimisation algorithms. As this is not the main focus of this work, I will just use them for illustration purposes, to indicate some of the problems or phenomena we need to expect to face when applying methods from Section 4.2.

The minimisation algorithms used were

i) Newton's method,

ii) NLSD without preconditioning,

iii) NLSD using the Newton direction by preconditioning with $B^{(k)} = (f''(x^{(k)}))$,

iv) NLSD preconditioned with lBFGS, using an lBFGS dimension of 2,

v) NLCG without preconditioning and modified Polak-Ribière-Polyak update,

vi) NLCG using the Newton direction by right preconditioning with $B^{(k)} = (f''(x^{(k)}))$ and modified Polak-Ribière-Polyak update,

with the absolute tolerance $\varepsilon_a = 1.4901\text{e}{-8}$, relative tolerance $\varepsilon_r = \varepsilon_a$ and step length tolerance $\varepsilon_s = 2.2204\text{e}{-16}$ where applicable.

In all cases using a line search, the used algorithm was used an inexact line search with mixed quadratic/cubic interpolation for determining the step sizes, with the constants $c_1 = 10^{-3}$ and

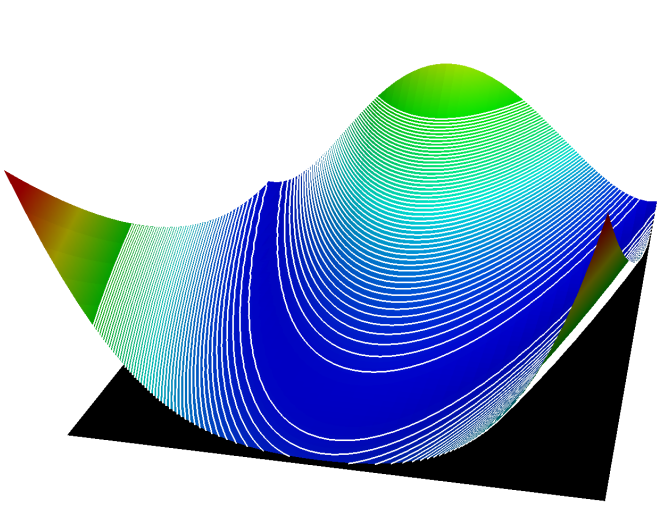
$c_2 = \frac{3}{10}$ in the strong Wolfe conditions (see Definition 4.3). If the preconditioned search direction is not a descent direction, all algorithms perform one step of unpreconditioned steepest descent, as a line search will generally fail along such a direction.

Example 4.14. *Rosenbrock's Function*

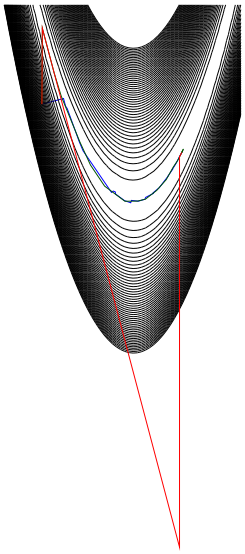
One well-known benchmark problem is finding the minimum $x^* = (1,1)^T$ *of the Rosenbrock function (see [Ros60])*

$$f(x_1,x_2) = 100(-x_1^2+x_2)^2 + (1-x_1)^2.$$

The function's steep valley proves difficult for most minimisation algorithms, but the Hessian is positive definite on the set $M := \left\{(x_1,x_2) \in \mathbb{R}^2 : x_2 < x_1^2 + \frac{1}{200}\right\}$. *If all iterates remain in this set, we can expect very good performance from Newton's method. I shall use it as a small example for illustration purposes. The starting point was* $x^{(0)} = (-1.9,2)^T \in M$.



(i) Illustration of the steep valley with colour according to $f(x)$ from 0 (blue) to 2000 (red) in $[-2,2] \times [-0.72,3.68] \subset \mathbb{R}^d$.



(ii) Isolines and the iterates of Newton's method (red), NLCG-Newton (green) and unpreconditioned NLCG (blue) in $[-3.28,2.22] \times [-7.78,4.06] \subset \mathbb{R}^d$.

Figure 4.1: Rosenbrock's function with 51 isolines from 0 to 1000.

As we can see in Table 4.1, the quadratic convergence of Newton's method wins in this case, and it can be verified that all iterates lie in M. The unpreconditioned NLSD performs very poorly, but preconditioning it using $B^{(k)} = (f''(x^{(k)}))$ *gives good results. However, using lBFGS as a preconditioner requires only slightly more evaluations of* f *and* f'*, and no evaluations of* f''*, so it might be more efficient overall. NLCG without preconditioning needs more evaluations than NLSD-lBFGS, but also requires slightly less computational effort. Preconditioning NLCG using* $B^{(k)} = (f''(x^{(k)}))$ *performs quite similar to NLSD-Newton. It requires less evaluations, but every iteration is slightly more expensive than of NLSD-Newton.*

The points I want to illustrate here are:

i) For this problem, Newton's method is applicable and naturally offers the best convergence behaviour, but we need to keep in mind that this will not be the case for nonconvex problems.

	Newton	NLSD	NLSD-Newton	NLSD-lBFGS	NLCG	NLCG-Newton
# its	5	8853	19	33	32	22
# evals f	6	51206	66	70	84	52
# evals f'	6	51207	66	71	84	52
# evals f''	5	0	19	0	0	22

Table 4.1: Statistics for several minimisation algorithms applied to Rosenbrock's function.

Especially if the problem is of high dimension, computing (let alone solving a linear system with) f'' will in general be prohibitively expensive.

ii) In Figure 4.1, we can see that the second iterate of Newton's method has a higher function value and is far away from the minimum along the search direction. This is due to not performing a line search, but using a fixed step length of 1 - the search direction is indeed a descent direction, just the step size is wrong. In this case, it causes the iteration to leave the steep valley, re-entering in the following iteration.

iii) The quasi Newton method NLSD-lBFGS is applicable even to problems where Newton's method is not, although the constructed preconditioner might not be as effective. But even in this example, the superlinear convergence only appears when the iterates $x^{(k)}$ are already "close enough" to x^*, meaning near the end of the iteration.

iv) All methods using a line search must iterate through the Rosenbrock function's steep valley, which greatly limits the step sizes requires lots of iterations, most notably in the unpreconditioned NLSD method.

Example 4.15. *Himmelblau's function*

Another well-known test function for minimisation algorithms is Himmelblau's function (see [Him72]):

$$f(x) = (x_1^2 + x_2 - 11)^2 + (x_2^2 + x_1 - 7)^2$$

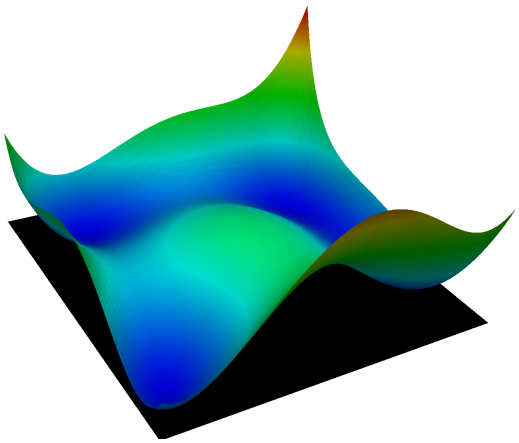
In contrast to Rosenbrock's function, it does not have a particularly steep gradient, but it has multiple local minima at

$$x^{*1} \approx (-3.77931 - 3.28319)^T, \qquad x^{*2} \approx (-2.80512, 3.13131)^T,$$
$$x^{*3} = (3,2)^T, \qquad x^{*3} \approx (3.58443, -1.84813)^T$$

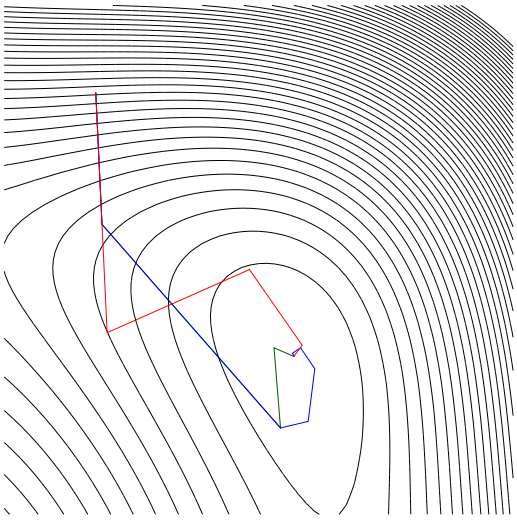
and a local maximum at $x^M \approx (-0.270845, -0.923039)^T$. Here, we cannot expect the Newton direction to always be a descent direction, and indeed the very first iterate is very far from the starting point and any of the known local minima. In the following, Newton's method converges to the local minimum x^{*1} and not to x^{*3} like all other solvers. This is the reason why the iterates of Newton's method are absent from Figure 4.2 (ii) and the iterates of NLSD-Newton are given instead.

The starting point of the minimisation is chosen as $x^{(0)} = \left(\frac{3}{2}, 4\right)^T$.

In this case, we can see that Newton's method does not offer the fastest convergence, which comes as no surprise since the prerequisites are not given. Equipping NLSD with the Newton direction requires only half the iterations, which is very important as the most expensive part of an iteration is the assembly of f'' and the solving with it to obtain the Newton direction. The most efficient solver is NLCG-Newton, which needs the same number of outer iterations (and thus solves



(i) Illustration of the multiple extrema with colour according to $f(x)$ from 0 (blue) to 500 (red) in $[-4.5, 4.5]^2 \subset \mathbb{R}^2$.



(ii) Isolines and the iterates of NLSD-Newton method (red), NLCG-Newton (green) and unpreconditioned NLCG (blue).

Figure 4.2: Himmelblau's function.

	Newton	NLSD	NLSD-Newton	NLSD-lBFGS	NLCG	NLCG-Newton
# its	15	28	7	14	10	7
# evals f	16	152	30	20	21	16
# evals f'	16	152	30	21	21	16
# evals f''	15	0	7	0	0	7

Table 4.2: Statistics for several minimisation algorithms applied to Himmelblau's function.

with f''), but less evaluations of f and f'. Interestingly, the quasi Newton method NLSD-lBFGS outperforms Newton's method, too.

Here, my point is that in the absence of strong convexity of the functional, we cannot expect the Newton direction to have as great an impact as e.g. for the Rosenbrock function. The flexibility offered by discarding non-descent directions in NLCG and NLSD is critical for the use of a line search and greatly improves the stability of the minimisation process, at the cost of computing f'' and then not being able to use the Newton direction. The class of problems we are interested in is nonconvex like this example, but additionally has (many) strong singularities, which is a strong argument for using a line search based minimisation algorithm that takes this into account.

Remark 4.16. *The given examples are to illustrate that for the minimisation of nonlinear functionals, especially for nonconvex functionals, the choice of the most suitable algorithm is even more problem dependent than for the solving of linear problems. There is no single algorithm that performs "best" even just for a significant set of benchmark problems (see [JY13] for a large catalogue).*

The same applies to the search direction update (meaning the computation of $\beta^{(k)}$) in the NLCG method. See [HZ05] for a comparison of different NLCG and NLSD-lBFGS variants on a total of 113 benchmark problems.

5

Numerical results part I: Mesh optimisation

In this chapter I will show some results of mesh optimisation based on the class of functionals derived in Section 3.5 and compare them with the results from a method using the $\mathbf{D}(u) : \mathbf{D}(v)$ bilinear form from Section 3.4.2 where possible. This functional offers significantly better results than the cheaper (graph) Laplacian based functionals, while the computational effort is lower than for more sophisticated functionals leading to linear systems of equations, like the biharmonic equation (see e.g. [Wic11]) or linearised elasticity. Especially the biharmonic equation is challenging to solve numerically ([Han93], [MS04]) if one tries to take advantage of imposing boundary conditions for $\partial_\nu \Phi$, leading to boundary conditions of the third kind.

5.1. 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 ([Tur99], [TGB$^+$10], [KOS$^+$12], [HKT14]). 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, 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 ([KT98],[TBK06]).

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 $\mathcal{F}'$ 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` ([Boc]), for which FEAT3 provides an interface.

5.2. Some mesh quality heuristics

Since we want to compare the minimisers of different mesh quality functionals, we need another notion of mesh quality that is independent of the quality functionals chosen.

For a cell K recall the definitions of the diameter $h(K)$, the in-circle diameter $\rho(K)$ and the aspect ratio $\sigma(K) = \frac{h(K)}{\rho(K)}$ from Definition 2.3. From the definitions of regular families of finite elements (Definition 2.10) and meshes (Definition 2.7) 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 $(\mathcal{T})$ of meshes. With the definitions

$$h_{\min}(K) = \min\{\mathrm{vol}(e) : e \in \mathcal{E}^1(K)\} \tag{5.1}$$

of the minimum edge length and

$$\gamma_{\max}(K) := \max \left\{ \frac{|(v_i - v_j, v_k - v_j)|}{\|v_i - v_j\|_2 \|v_k - v_j\|_2} : \exists e_{ij}, e_{kj} \in \mathcal{E}^1(K) : v_i, v_j \in e_{ij}, v_k, v_j \in e_{jk} \right\} \tag{5.2}$$

of the maximum of the cosine of angles between two edges of a cell K.

To measure this, we chose the following mesh shape quality heuristics:

Definition 5.1. *Shape quality heuristics*

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

$$\mathcal{Q}(K) := \begin{cases} \frac{h_{\min}(K)}{h(K)} \sqrt{1 - \gamma_{\max}}, & K \text{ is convex} \\ 0 & \text{else} \end{cases} \tag{5.3}$$

if K is a 2-hypercube and

$$\mathcal{Q}(K) = \frac{1}{h(K)} \sqrt[d]{\frac{\mathrm{vol}(K)}{\mathrm{vol}(\hat{S}_n)}} \tag{5.4}$$

if K is a d-simplex (see also [CR72, Sections 5 and 6]), and define the shape quality heuristic

$$\mathcal{Q}(\mathcal{T}_h) := \min_{K \in \mathcal{E}^2(\mathcal{T}_h)} \mathcal{Q}(K). \tag{5.5}$$

and the the average shape quality heuristic

$$\mathcal{Q}_a(\mathcal{T}_h) := \frac{1}{\mathrm{card}(\mathcal{E}^2(\mathcal{T}_h))} \sum_{K \in \mathcal{E}^d \mathcal{T}_h} \mathcal{Q}(K). \tag{5.6}$$

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 $\mathcal{Q}(\hat{S}) = \mathcal{Q}(\hat{Q}) = 1$ (hence the additional scaling factor $1/\sqrt[d]{\mathrm{vol} S_n}$ in (5.4)). To the best of my 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 $|R^{-1}|_{1,\infty,K}$ (see Lemma 2.6) where $R : \hat{Q} \to K$ is the local reference mapping.

The hypercube quality heuristic (5.3) 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.

Because it is very visual, another quality heuristic frequently used in $2d$ is the minimum spatial angle $\alpha_{\min}$ between edges. For 2-simplices, this is equivalent to the measure in (5.4) in the sense

that $(\alpha_{\min}(\mathcal{T}_i) \to 0) \Leftrightarrow (\mathcal{Q}(\mathcal{T}_i) \to 0)$ and that its maximum is attained at $\alpha_{\min}(K) = \frac{\pi}{3}$ iff K is an equilateral 2-simplex. For hypercubes, this is not a very useful quality heuristic for the reasons stated above, and the maximum angle is equally important. But it is still useful to compute these angles to recognise if the $\mathcal{Q}$ is low because of bad angles or because of strong anisotropies.

Remark 5.1. *The use of the* minimum angle *to estimate the quality of simplices goes back to results from [Syn57]. The proof for an interpolation error estimate similar to Lemma 2.3 for $m = 1$ contained therein requires the solution u to be of class $\mathcal{C}^2(\Omega)$. This was generalised to $u \in H^2(\Omega)$ in [Jam76], resulting in an estimate that instead depends on the* maximum angle*. See [AD00] for an error estimate for hypercubes in 2d using a maximum angle condition.*

Definition 5.2. *Worst angle*

Define the minimum, maximum and worst angle α_w *of a mesh $\mathcal{T}_h$ according to*

$$\alpha_{\min}(\mathcal{T}_h) := \min_{K\in\mathcal{T}_h}\left\{\frac{(a_i-a_j,a_k-a_j)}{\|a_i-a_j\|_2\,\|a_k-a_j\|_2} : \exists e_{ij}\in\mathcal{E}^1(K) : a_i,a_j\in\overline{e_{ij}}, \exists e_{kj}\in\mathcal{E}^1(K) : a_k,a_j\in\overline{e_{kj}}\right\}$$
$$\alpha_{\max}(\mathcal{T}_h) := \max_{K\in\mathcal{T}_h}\left\{\frac{(a_i-a_j,a_k-a_j)}{\|a_i-a_j\|_2\,\|a_k-a_j\|_2} : \exists e_{ij}\in\mathcal{E}^1(K) : a_i,a_j\in\overline{e_{ij}}, \exists e_{kj}\in\mathcal{E}^1(K) : a_k,a_j\in\overline{e_{kj}}\right\}$$
$$\alpha_w(\mathcal{T}_h) := \min\{\alpha_{\min}(\mathcal{T}_h), |\alpha_{\max}(\mathcal{T}_h)-\pi|\}$$

In addition to this scaling invariant shape quality heuristic, we need a somewhat shape independent size quality heuristic. Assume that we have an optimal cell size distribution $\lambda : \mathcal{E}^d(\mathcal{T}_h) \to [0,1]$ with $\sum_{K\in\mathcal{E}^d(\mathcal{T}_h)}\lambda(K) = 1$ (see Section 3.5.4).

Definition 5.3. *Size distribution defect*

Define the size distribution defect *of a mesh $\mathcal{T}_h$ on a domain Ω_h as*

$$\mathcal{S}(\mathcal{T}_h) = \sum_{K\in\mathcal{E}^2(\mathcal{T}_h)}\left|\lambda(K) - \frac{\mathrm{vol}(K)}{\mathrm{vol}(\Omega_h)}\right|, \tag{5.7}$$

which is just the norm $\|\mathcal{S}\|_1$ of the corresponding vector of differences $\mathcal{S}$. Sometimes the notation $\mathcal{S}_i := \mathcal{S}(K_i)$ for some $K_i \in \mathcal{E}^d(\mathcal{T}_h)$ will be used.

5.3. Refinement of a unit circle mesh

In this case 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.

We obtain this by using

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

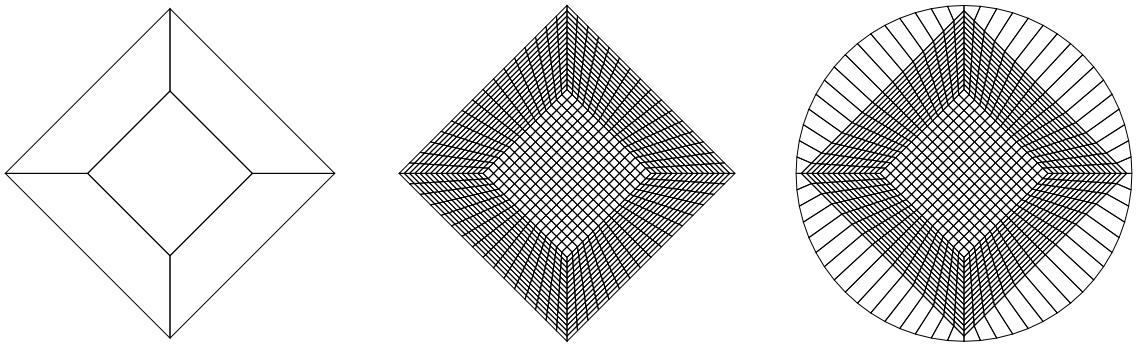
with a mesh $\mathcal{T}_{h,0}$ which we then refined using regular refinement to obtain the mesh $\mathcal{T}_{h,l}$ which in turn defines the boundary $\partial\Omega_{h,l}$ and thus $\Omega_{h,l}$. In the following, I omit the index l if this is unambiguous.

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

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

with which we can define the vertex mapping

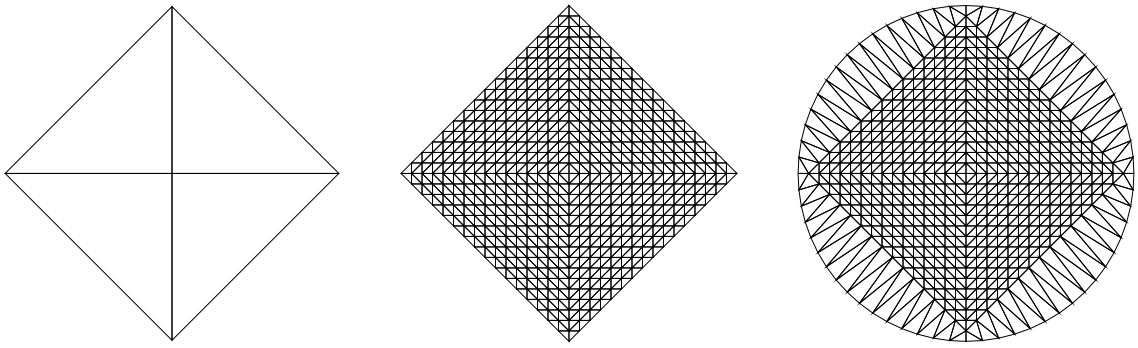
$$\hat{\varphi} : \mathcal{E}^0(\mathcal{T}_h) \to \mathbb{R}^2, \hat{\varphi}(v) = \begin{cases} \mathrm{tr}\,\varphi(v), & v \in \partial\mathcal{T}_h \\ v, & v \notin \partial\mathcal{T}_h(t_k) \end{cases}.$$



(i) $\mathcal{T}_{h,0}$. (ii) $\mathcal{T}_{h,4}$. (iii) $\hat{\varphi}(\mathcal{T}_{h,4})$.

Figure 5.1: Various meshes for the approximation of the unit circle, hypercubes.

The mapping $\hat{\varphi}$ also defines a member of $\mathcal{D}_h(\mathcal{T}_h)$ 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.



(i) $\mathcal{T}_{h,0}$. (ii) $\mathcal{T}_{h,4}$. (iii) $\hat{\varphi}(\mathcal{T}_{h,4})$.

Figure 5.2: Various domains for the approximation of the unit circle, simplices.

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

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

When the mesh $\mathcal{T}_{h,0}$ is refined l times, we only adapt $\partial\mathcal{T}_{h,l}$ (meaning applying $\hat{\varphi}$ after the lth refinement). This leads to cells of very poor quality in $\hat{\varphi}(\mathcal{T}_{h,l})$, so this methodology is not very useful in situations where we could just perform this adaption in every step of refinement.

However, this is very useful for generating a sequence of initial guesses whose quality deteriorates with increasing l to investigate if a regular family of meshes can be recovered by a mesh optimisation technique. More explicitly, we want to see if said technique can recover from an initial mesh of poor quality, which could be the output of an external mesh generation tool. As an additional requirement, we want to have a uniform cell size distribution which provides an additional challenge as the starting solution's distribution is quite nonuniform. This requirement is given by choosing a uniform reference mesh for $\mathcal{G}_h$ and $\lambda \equiv \text{const}$ for $\mathcal{F}_h$, which will be defined below.

Define the spaces

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

The discrete $\mathbf{D}(u) : \mathbf{D}(v)$ functional is defined by

$$\forall (u,v) \in V_h \times W_h : \quad a_h(u,v) = \int_{\hat{\Omega}_h} \mathbf{D}(u) : \mathbf{D}(v) dx$$

and then setting

$$\mathcal{G}_h(u) := \frac{1}{2} a_h(u,u).$$

In this case, we will use the reference domain $\hat{\Omega}_h = \Omega_h$ and the reference mesh $\hat{\mathcal{T}}_h = \mathcal{T}_h$.

So we are looking for

$$\Psi_h^* = \operatorname{argmin}_{u \in V_h} \mathcal{G}_h(u) \Leftrightarrow \forall w \in W_h : \ \frac{\partial \mathcal{G}_h}{\partial w}(\Psi_h^*) = 0.$$

For the hyperelasticity based functional, recall (3.23):

$$\mathcal{F}_h(\Phi) = \int_{\Omega_h} c_f (\|\nabla \Phi\|_F^2 - d)^2 dx + (\det(\nabla \Phi))^{p_d} dx + \frac{c_d}{\left(\det(\nabla \Phi) + \sqrt{\delta_r^2 + (\det(\nabla \Phi))^2} \right)^{p_d}} dx.$$

We are then looking for a discrete optimal variation Φ_h^* in the sense of (3.24):

$$\Phi_h^* = \operatorname{argmin}_{\Phi \in V_h} \mathcal{F}_h(\Phi).$$

In this example, the parameters are

$$c_f = 1, \delta_r = 1\text{e}{-8}, p_d = 1, c_d = \sqrt{\delta_r^2 + 1} + \delta_r^2 + 1.$$

Remark 5.2. *In this section, I give results obtained by using unpreconditioned solvers (CG for $\mathcal{G}_h$ and NLCG for $\mathcal{F}_h$). This is done for illustration purposes, to demonstrate the behaviour of the functionals for $h \to 0$ and to serve as motivation for Chapter 6.*

In the corresponding tables, the computational time is given in seconds of wall clock time. This is meant to give a rough idea about the relative computational effort of the different methods and of different problem sizes and is by no means to be understood as benchmarking information.

Table 5.4 and Table 5.1 summarise the problem sizes and some properties of the initial guess $\mathcal{T}_h$. Because we are using global, regular refinement (where a cell is split into four cells upon refinement), the number of cells increases by a factor of four from level to level, but since the coarse meshes consist of five cells (hypercube mesh) and four cells (simplex mesh), the numbers are slightly different.

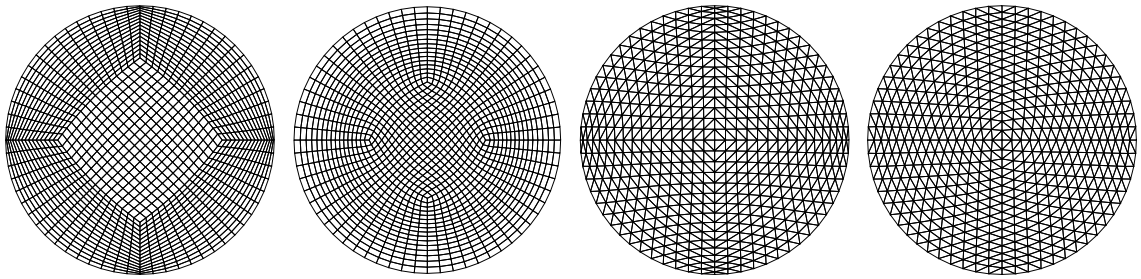
Remark 5.3. *The settings for the solvers were:*

i) *$\mathcal{G}_h$: Unpreconditioned CG with relative stopping criterion $\varepsilon_r = 1\text{e}{-8}$.*

ii) *$\mathcal{F}_h$: Unpreconditioned NLCG with relative stopping criterion $\varepsilon_r = 1\text{e}{-8}$, function value decrease tolerance $\varepsilon_f = 0$ and step length criterion $\varepsilon_s \approx 2.2204\text{e}{-16}$ (machine precision for double precision). The linesearch used $c_1 = 1\text{e}{-3}, c_2 = 0.3$ for the strong Wolfe conditions and a step length criterion of $\varepsilon_s = 5\text{e}{-14}$ 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.*

In all tables, cases in which the 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, which is an indicator for the problem being too badly conditioned for the nonlinear solver to make any further progress. One could set $\varepsilon_s = 0$, but in most cases this would just lead 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 CG or NLCG iterations are given. In the case of NLCG, this is not a good metric, as the number of functional and gradient evaluations can be very different from iteration to iteration due to the linesearch requiring different numbers of iterations. The more relevant quantity is the number of functional evaluations, which is omitted here since the iteration numbers are just stated for motivation purposes. The number of functional evaluations will be given and discussed in Section 7.1.



(i) $\Psi^*(\mathcal{T}_{h,4})$. (ii) $\Phi^*(\mathcal{T}_{h,4})$. (iii) $\Psi^*(\mathcal{T}_{h,4})$. (iv) $\Phi^*(\mathcal{T}_{h,4})$.

Figure 5.3: Optimised meshes on refinement level four.

The solutions Ψ_h^*, Φ_h^* for refinement level four are depicted in Figure 5.3, where some statistics data about them is given in Table 5.5 and in Table 5.6 for the hypercube meshes, and in Table 5.2 and Table 5.3 for the simplex meshes.

Let us first visually examine the meshes $\mathcal{T}_{h,0}, \mathcal{T}_{h,l}$ and the initial guess $\hat{\varphi}(\mathcal{T}_{h,l})$ in Figure 5.1 and Figure 5.2. It can be seen that the coarsest mesh does not capture the geometry very well and that its boundary has singular points (in the sense of Section 3.1), which poses some problems as we will see. The mesh $\mathcal{T}_{h,l}$ on the computational domain on which the functional $\mathcal{G}_h$ is minimised is just a refinement of the initial coarse mesh $\mathcal{T}_{h,0}$ and completely uniform in the simplex case, where the low regularity of the boundary forces us to use a less uniform coarse mesh for the hypercube case. This is due to the considerations from Section 3.5.8, also see Figure 3.19 and Figure 3.20. Note that the mesh $\hat{\varphi}(\mathcal{T}_h)$ only visualises the starting point for the minimisation of $\mathcal{G}_h$ and $\mathcal{F}_h$, but that in the case of $\mathcal{G}_h$ no computations are carried out on these ill-shaped meshes. Note that the particular choice of the initial guess is of no consequence if the solution to the minimisation problem is unique (e.g. for $\mathcal{G}_h$), but that this is not the case for the functionals of the family $\mathcal{F}_h$.

Note that the starting meshes $\hat{\varphi}(\mathcal{T}_{h,l})$ behave quite differently upon refinement. For the hypercube mesh, the worst angle remains the same while the ratio $\frac{h_{\min}(K)}{h(K)}$ deteriorates for the cells K at the boundary (see also Table 5.4). For the simplex mesh, angles of cells at the boundary tend to zero with further refinement (see also Table 5.1). It should also be noted that, because the vertices are projected to the boundary in the direction of the outer normal, the cells at the singular points will always be of the lowest quality. Because of the ratio between cells in the interior and cells at the boundary, $\mathcal{Q}_a$ is not monotone with refinement.

First, let us examine the simplex case. The minimisers of the functional $\mathcal{G}_h$ do not satisfy any useful lower bound on $\mathcal{Q}$ and are not able to recover meshes of good quality with the given

l	CELLS	VERTICES	DOF	$\alpha_w(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}_a(\hat{\varphi}(\mathcal{T}_h))$
3	256	145	290	26.1	$7.13e{-1}$	$7.68e{-1}$
4	1024	545	1090	14.94	$5.46e{-1}$	$7.48e{-1}$
5	4096	2113	4226	8.03	$4.02e{-1}$	$7.45e{-1}$
6	16384	8321	16642	4.16	$2.90e{-1}$	$7.49e{-1}$
7	65536	33025	66050	2.12	$2.07e{-1}$	$7.53e{-1}$
8	262144	131585	263170	1.07	$1.47e{-1}$	$7.56e{-1}$
9	1048576	525313	1050626	0.54	$1.04e{-1}$	$7.57e{-1}$
10	4194304	2099201	4198402	0.27	$7.37e{-2}$	$7.59e{-1}$
11	16777216	8392705	16785410	0.13	$5.21e{-2}$	$7.59e{-1}$

Table 5.1: Number of entities and properties of the initial guess $\mathcal{T}_h$ for the unit circle simplex meshes.

boundary conditions, even though the reference domain has a uniform discretisation and cells of good shape. The minimal cell shape quality heuristic even decreases on refinement level three, four and five (see Table 5.3), as does the worst angle α_w. On all levels, the average cell quality $\mathcal{Q}_a$ slightly improves, which is confirmed visually in Figure 5.3, especially with regard to the uniform cell size distribution of the reference domain. However, the low values for $\mathcal{Q}$ mean that the meshes obtained by this method are less suitable for solving partial differential equations on this domain with the finite element method.

In contrast to this, the functional $\mathcal{F}_h$ is independent of a reference domain, but has a set of (independent) reference cells and measures some energy of the deformation from a reference cell to the corresponding cell in the mesh. The minimisers Φ^* of the functional $\mathcal{F}_h$ show major improvement of α_w, $\mathcal{Q}$ and $\mathcal{Q}_a$ over the initial guess mesh $\hat{\varphi}(\mathcal{T}_h)$ (see Table 5.3). However, the solver does not converge with regard to the relative residual tolerance ε_r from level nine onwards and either stops due to the step length criterion (level ten) or stagnates (levels nine and eleven). The effect of the solver stopping early is very strong, but as long as the solver is able to reach the relative tolerance, the angles and the quality indicator remain in regimes that are acceptable for using the corresponding meshes for finite element discretisations. Also note that even without convergence with regard to ε_r and stopping after very few iterations (most notably on refinement levels ten and eleven), the minimiser Φ^* produces a better mesh with regard to the shape quality heuristic $\mathcal{Q}$ than the minimiser Ψ^* of the functional $\mathcal{G}_h$, in comparable run time.

The iteration numbers for the minimisation of $\mathcal{G}_h$ show the typical behaviour associated with second order elliptic operators and an unpreconditioned CG solver, as the number increases roughly by a factor of two for each level of refinement. The interesting observation here is that even though the functional $\mathcal{F}_h$ is highly nonlinear and is minimised with a nonlinear solver, it shows the same behaviour for the refinement levels three to eight. This will serve as a starting point for deriving a preconditioner in Chapter 6.

Now consider the hypercube case. If we look at the shape quality heuristic $\mathcal{Q}$ in Table 5.5, we note that the worst angle α_w remains nearly the same over all levels of refinement, whereas the shape quality indicator $\mathcal{Q}$ decreases by two orders of magnitude. Examining the solution on level four given in Figure 5.3 we see strong anisotropies in the mesh, especially near the singular points of boundary of the computational domain and mesh $\hat{\mathcal{T}}_h$. As the functional measures the energy of

l	$\alpha_w(\Psi^*)$	$\mathcal{Q}(\Psi^*)$	$\mathcal{Q}_a(\Psi^*)$	# its	$\frac{\|\mathcal{G}'(\mathcal{T}_h)\|_2}{\|\mathcal{G}'(\Psi^*)\|_2}$	time
3	15.92	5.60e−1	8.01e−1	21	8.92e−9	5.87e−4
4	11.05	4.68e−1	8.01e−1	47	8.01e−9	2.08e−3
5	7.54	3.88e−1	8.02e−1	98	7.69e−9	7.47e−3
6	5.07	3.19e−1	8.02e−1	193	9.11e−9	5.02e−2
7	3.38	2.61e−1	8.02e−1	375	9.77e−9	3.83e−1
8	2.23	2.12e−1	8.02e−1	732	9.61e−9	3.59e+0
9	1.46	1.72e−1	8.02e−1	1423	1.00e−8	6.85e+1
10	0.96	1.39e−1	8.02e−1	2770	9.95e−9	6.91e+3
11	0.62	1.12e−1	8.02e−1	5346	9.94e−9	3.74e+4

Table 5.2: Statistics for the minimisers of $\mathcal{G}_h$, simplices, unpreconditioned CG.

l	$\alpha_w(\Phi^*)$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\Phi^*)$	# its	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	29.25	7.21e−1	8.49e−1	44	8.71e−9	6.36e−3
4	27.27	6.94e−1	8.51e−1	103	9.42e−9	4.25e−2
5	25.77	6.76e−1	8.52e−1	208	9.02e−9	2.95e−1
6	24.64	6.63e−1	8.53e−1	400	9.76e−9	2.08e+0
7	23.56	6.53e−1	8.53e−1	1081	9.52e−9	2.92e+1
8	22.63	6.44e−1	8.53e−1	3908	9.86e−9	6.81e+2
9s	10.77	4.63e−1	7.77e−1	1156	6.36e−7	1.38e+3
10*	3.72	2.74e−1	7.13e−1	388	6.66e−7	1.06e+3
11s	5.09	3.40e−1	7.43e−1	359	6.38e−7	3.97e+3

Table 5.3: Statistics for the minimisers of $\mathcal{F}_h$, simplices, unpreconditioned NLCG.

the deformation with regard to the computational domain and the computational domain contains (slight) anisotropies, this seems unavoidable. Table 5.5 only shows the lowest value of $\mathcal{Q}(\Psi^*(\hat{\mathcal{T}}_h))$, which is good for determining whether a mesh has become unsuitable for further computations, but is misleading for judging the average quality of the mesh's cells. As can be observed in Figure 5.3, the application of Ψ^* improves the average quality $\mathcal{Q}_a$ (see Table 5.6), especially when bearing in mind that a uniform size distribution was one of the goals. Still, the low values for $\mathcal{Q}$ make the meshes resulting from this method less desirable (as in the simplex case).

As the functional $\mathcal{F}_h$ is independent of a reference domain and isotropic reference cells were chosen, the functional $\mathcal{F}_h$ tends to reduce anisotropies but cannot remove them entirely due to other constraints as can be observed in Figure 5.3. The solver does not reach the required tolerance on the relative residual from refinement level ten onwards. Instead, it stagnates or stops because of the step size tolerance, which indicates that the problem is too badly conditioned for the linesearch to make any further progress. The effect is noticeable in the quality heuristics $\mathcal{Q}$ and $\mathcal{Q}_a$ of the

solutions from refinement level ten on, although they are still higher than for the minimisers of the functional $\mathcal{G}_h$ (compare Table 5.5 and Table 5.6).

l	CELLS	VERTICES	DOF	$\alpha_w(\mathcal{T}_h)$	$\mathcal{Q}(\mathcal{T}_h)$	$\mathcal{Q}_a(\mathcal{T}_h)$
3	320	337	674	45	$1.80e{-1}$	$4.24e{-1}$
4	1280	1313	2626	45	$1.86e{-1}$	$4.22e{-1}$
5	5120	5285	10570	45	$1.39e{-1}$	$4.26e{-1}$
6	20480	20048	40096	45	$7.18e{-2}$	$4.29e{-1}$
7	82920	82177	164354	45	$3.65e{-2}$	$4.32e{-1}$
8	327680	328193	656386	45	$1.84e{-2}$	$4.33e{-1}$
9	1310720	1311745	2623490	45	$9.23e{-3}$	$4.34e{-1}$
10	5242880	5244929	10489858	45	$4.62e{-3}$	$4.35e{-1}$
11	20971520	20975617	41951234	45	$2.31e{-3}$	$4.35e{-1}$

Table 5.4: Number of entities and properties of the initial guess $\mathcal{T}_h$ for the unit circle hypercube meshes.

l	$\alpha_w(\Psi^*)$	$\mathcal{Q}(\Psi^*)$	$\mathcal{Q}_a(\Psi^*)$	# its	$\frac{\|\mathcal{G}'(\mathcal{T}_h)\|_2}{\|\mathcal{G}'(\Psi^*)\|_2}$	time
3	52.66	$9.81e{-2}$	$4.40e{-1}$	45	$8.50e{-9}$	$9.30e{-4}$
4	52.93	$6.78e{-2}$	$4.58e{-1}$	102	$9.15e{-9}$	$5.72e{-3}$
5	53.04	$4.61e{-2}$	$4.68e{-1}$	215	$9.68e{-9}$	$5.14e{-1}$
6	53.09	$3.10e{-2}$	$4.73e{-1}$	435	$9.83e{-9}$	$4.62e{+0}$
7	53.11	$2.06e{-2}$	$4.75e{-1}$	860	$9.88e{-9}$	$5.24e{+1}$
8	53.12	$1.36e{-2}$	$4.76e{-1}$	1689	$9.93e{-9}$	$7.15e{+2}$
9	53.12	$8.88e{-3}$	$4.77e{-1}$	3315	$9.72e{-9}$	$4.12e{+3}$
10	53.12	$5.77e{-3}$	$4.77e{-1}$	6430	$9.94e{-9}$	$5.60e{+4}$
11	53.12	$3.74e{-3}$	$4.78e{-1}$	12356	$9.98e{-9}$	$3.46e{+5}$

Table 5.5: Statistics for the minimisers of $\mathcal{G}_h$, hypercubes, unpreconditioned CG.

In this example, the nonlinear class $\mathcal{F}_h$ of mesh quality functionals is superior to the quadratic functionals $\mathcal{G}_h$, as it can recover meshes with good shape quality heuristics from badly shaped initial guesses, which could be the input from external mesh generation tools, over a descent range of refinement levels. Solving the nonlinear minimisation problem for $\mathcal{F}_h$ proves to be difficult, as the nonlinear solver tends to stagnate without converging with regard to the relative residual criterion. Even in the cases where this happens, the results are still better than the ones obtained from minimising the quadratic functional $\mathcal{G}_h$, although the nonuniqueness of the solution (see Theorem 3.11) might prove to be problematic. Also, the problem is harder in the case of hypercube meshes, likely due to the different nonuniform mesh and the fact that the transformation Φ is not

l	$\alpha_w(\Phi^*)$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\Phi^*)$	# its	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	55.17	$4.22e{-1}$	$5.70e{-1}$	95	$9.97e{-9}$	$2.78e{+0}$
4	54.84	$4.41e{-1}$	$5.97e{-1}$	190	$7.19e{-9}$	$1.81e{+1}$
5	55.36	$4.52e{-1}$	$6.12e{-1}$	422	$8.80e{-9}$	$1.48e{+2}$
6	56.02	$4.57e{-1}$	$6.20e{-1}$	717	$9.91e{-9}$	$9.86e{+2}$
7	56.6	$4.60e{-1}$	$6.24e{-1}$	1209	$9.53e{-9}$	$6.52e{+3}$
8	57.07	$4.61e{-1}$	$6.26e{-1}$	1991	$9.64e{-9}$	$4.02e{+4}$
9	57.53	$4.65e{-1}$	$6.36e{-1}$	3256	$9.69e{-9}$	$2.55e{+5}$
10s	36.95	$1.12e{-1}$	$4.52e{-1}$	990	$7.12e{-7}$	–
11*	17.07	$2.70e{-2}$	$4.36e{-1}$	1357	$1.14e{-6}$	–

Table 5.6: Statistics for the minimisers of $\mathcal{F}_h$, hypercubes, unpreconditioned NLCG.

linear, meaning that for a cell $K \in \mathcal{E}^d(\mathcal{T}_h) : \nabla\Phi_{|K} \neq \text{const}$.

This example will be revisited in Section 7.1 to compare different nonlinear solvers, the solutions obtained by them, and the aspect of the preconditioner that will be introduced in Chapter 6.

5.4. Computation of extension operators

In this section, I will show how the functionals $\mathcal{G}_h$ and $\mathcal{F}_h$ from Section 5.3 can be modified and used to compute an extension of a boundary movement into the interior of the domain of interest to adjust the mesh's vertices there. This is of practical importance e.g. for fluid-structure interactions, two-phase flow or even flows with free capillary surface and is a slightly more sophisticated example of what was presented in Section 3.4.

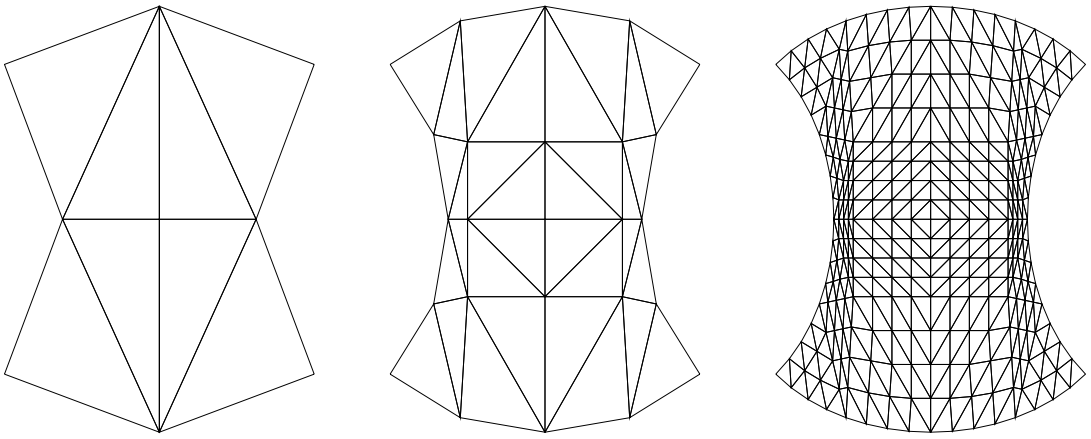
5.4.1. Moving nonconvex shape

The domain of interest is $\Omega = \Omega(0) \subset \mathbb{R}^2$ bounded by the composite cubic Bézier curve given by

$$\gamma_i : [0,1] \to \mathbb{R}^2, \gamma_i(t) := (1-t)^3 P_{i1} + 3(1-t)^2 t P_{i2} + 3(1-t)t^2 P_{i3} + t^3 P_{\mathrm{mod}_4(i+1)1}$$

with

$$\begin{aligned}
P_{11} &= (0,0)^T, P_{12} = \left(\frac{1}{4}, -\frac{1}{4}\right)^T, P_{13} = \left(\frac{3}{4}, -\frac{1}{4}\right)^T, \\
P_{21} &= (1,0)^T, P_{22} = \left(\frac{3}{4}, \frac{1}{4}\right)^T, P_{23} = \left(\frac{3}{4}, \frac{3}{4}\right)^T, \\
P_{31} &= (1,1)^T, P_{32} = \left(\frac{3}{4}, \frac{5}{4}\right)^T, P_{33} = \left(\frac{1}{4}, \frac{5}{4}\right)^T, \\
P_{41} &= (0,1)^T, P_{42} = \left(\frac{1}{4}, \frac{3}{4}\right)^T, P_{43} = \left(\frac{1}{4}, \frac{1}{4}\right)^T,
\end{aligned}$$



(i) $\mathcal{T}_{h,0}$. (ii) $\mathcal{T}_{h,1}$. (iii) $\mathcal{T}_{h,3}$.

Figure 5.4: The initial mesh $\mathcal{T}_{h,l}$ for various refinement levels.

which can be seen as a unit square with deformed boundary. For the sake of brevity, I will only consider simplex meshes in this section (see Figure 5.4) and will only discuss the shape quality heuristic $\mathfrak{Q}$ (see Section 5.2). The smallest angles are still plotted because they are used as criterion for an early stop.

The boundary $\partial\Omega(0)$ is piecewise smooth, but is only Lipschitz continuous at the corner vertices P_{i1}. As in Section 5.3, the analytic description of the boundary given by γ_i is used to adapt the boundary of the polygonal approximation $\Omega_h(0)$ and the initial mesh $\mathcal{T}_{h,l}(0)$ upon initial refinement to level l (the index l will again be omitted where possible). Refer to Table 5.7 for the resulting number of entities. Also note that the starting configuration is already nonconvex.

l	Cells	Vertices	DoF
3	512	256	289
4	2048	1089	2178
5	8192	4225	8450
6	32768	16641	33282
7	131072	66049	132098
8	524288	263169	526338
9	2097152	1050625	2101250

Table 5.7: Numbers of entities for different levels of refinement.

After this, the boundary is deformed using the boundary deformation

$$\operatorname{tr}\varphi : [0,\bar{t}] \times \partial\Omega(0) \to \mathbb{R}^2, \operatorname{tr}\varphi(t,x) = x + t \begin{pmatrix} x_1 - z_1 + (x_2 - z_2)^3 \\ -(x_1 - z_1 + (x_2 - z_2)^3 - z_1 + (x_1 - z_1)^3) \end{pmatrix}.$$

For a partitioning $0 = t_0 < \cdots < t_N = \bar{t}$, we start with the mesh $\mathcal{T}_h(0)$ on $\Omega_h(0)$ and can define

the mappings

$$\hat{\varphi}_0 : \mathcal{E}^0(\mathcal{T}_h(0)) \to \mathbb{R}^2, \hat{\varphi}_0(v) = v,$$
$$\forall l,k \in \mathbb{N}, l < k : \hat{\varphi}_{l,k} : \mathcal{E}^0(\mathcal{T}_h(t_l)) \to \mathbb{R}^2, \hat{\varphi}_k(v) = \begin{cases} \operatorname{tr}\varphi(t_k, (\operatorname{tr}\varphi)^{-1}(t_l, v)), & v \in \partial\mathcal{T}_h(t_l) \\ v, & v \notin \partial\mathcal{T}_h(t_k) \end{cases},$$

which define members of $\mathbb{P}_1(\mathcal{T}_h(t_l))$ (which again will be denoted by $\hat{\varphi}_{l,k}$) and can be used to parametrise $\mathcal{T}_h(t_k)$ directly over $\mathcal{T}_h(t_l)$ instead of $\mathcal{T}_h(0)$. Note that the boundary deformation is not divergence free. In fact the volume of the domain grows over time, which together with the varying deformation speed presents an additional difficulty.

Define the spaces

$$V_{h,l,k} = \left\{ v \in \mathcal{D}_h(\mathcal{T}_h(t_l)) : v_{|\partial\Omega_h(t_l)} = \hat{\varphi}_{l,k} \right\}, W_{h,l,k} = \left\{ w \in \mathcal{D}_h(\mathcal{T}_h(t_l)) : w_{|\partial\Omega_h(t_l)} = 0 \right\}$$

with which we can define the discrete $\mathbf{D}(u) : \mathbf{D}(v)$ functional

$$\forall (l,k,u) \in \mathbb{N} \times \mathbb{N} \times V_{h,l,k} : \mathcal{G}(l,k,u) := \frac{1}{2} \int_{\Omega_h(t_l)} \mathbf{D}(u) : \mathbf{D}(u) dx$$

So we are looking for

$$\Psi^*_{h,l,k} = \operatorname{argmin}_{u \in V_{h,l,k}} \mathcal{G}(l,k,u) \Leftrightarrow \forall w \in W_{h,l,k} : \ \frac{\partial \mathcal{G}_h}{\partial w}(\Psi^*_{h,l,k}) = 0.$$

This might appear complicated, but for arbitrary but fixed $k \in \mathbb{N}$, it contains the two most important cases.

i) $l = 0$: The functional $\mathcal{G}_h$ is formulated with regard to the fixed reference domain $\Omega_h(0)$). This requires the least computational effort, as the system matrix is assembled just once.

ii) $l = k-1$: The functional $\mathcal{G}_h$ is formulated with regard to the moving reference domain $\Omega_h(t)$. Here, the system matrix has to be reassembled in every time step.

Note that the conditions $\Psi^*_{k,l,k} \in \mathcal{D}_h(\mathcal{T}_h(t_l))$ are *not* enforced directly in any way. Instead, the more standard finite element spaces $\mathbb{P}_1(\mathcal{T}_h(t_l))$ are used and the orientation preserving property is checked a posteriori.

For the hyperelasticity based functional, recall (3.23):

$$\mathcal{F}_{h,k}(\Phi) = \int_{\Omega_h(t_k)} c_f(\|\nabla\Phi\|_F^2 - d)^2 dx + (\det(\nabla\Phi))^{p_d} dx + \frac{c_d}{\left(\det(\nabla\Phi) + \sqrt{\delta_r^2 + (\det(\nabla\Phi))^2}\right)^{p_d}} dx.$$

We are then looking for a discrete optimal variation $\Phi^*_{h,k}$ in the sense of (3.24):

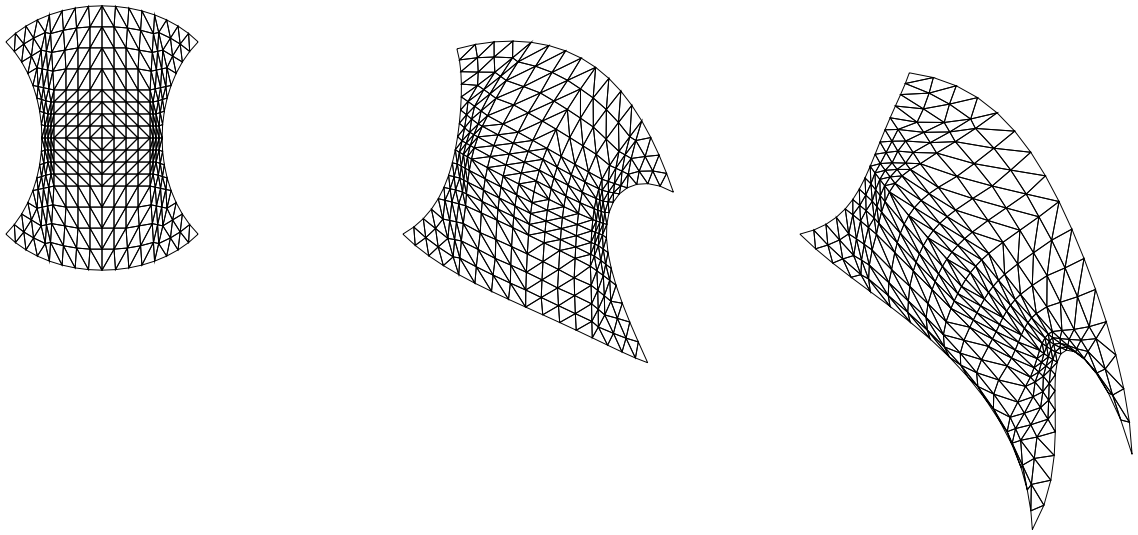
$$\Phi^*_{h,k+1} = \operatorname{argmin}_{\Phi \in V_{h,k,k+1}} \mathcal{F}_{h,k}(\Phi).$$

Since the functional $\mathcal{F}_h$ does not require a reference domain, the choice of $l = k$ has no impact on the computational cost. For all functionals, the initial guess is chosen as $\hat{\varphi}_{k,k+1}$.

After computing the appropriate minimiser, set $\Omega_h(t_{k+1}) = \Psi^*_{h,l,k+1}(\Omega_h(t_l))$ or $\Omega_h(t_{k+1}) = \Phi^*_{h,k+1}(\Omega_h(t_k))$ depending on the method used.

The parameters for $\mathcal{F}_h, \mathcal{G}_h$ used here are the same as in Section 5.3. Additionally, we use

$$\Delta t = 5\text{e}{-4}, \bar{t} = 0.5, z = (0,0)^T, \bar{\alpha} = \frac{\pi}{180} \tag{5.8}$$



(i) $t = 0$. (ii) $t = 0.25$. (iii) $t = 0.49850$.

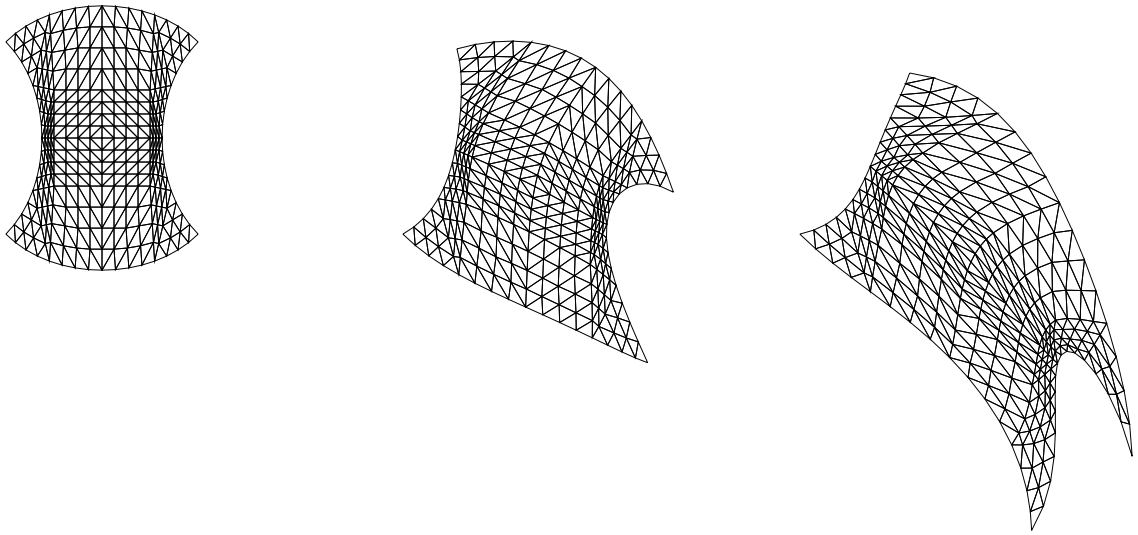
Figure 5.5: The minimisers $\Psi^*(\mathcal{T}_h(t))$ of $\mathcal{G}_{h,0,k}$ for different time steps t, refinement level three.

and stop the computation as soon as $\alpha_w < \bar{\alpha}$. This value is chosen based on numerical experience. The small time step size is chosen because we want to study the behaviour of both methods for several levels of refinement in space and need to ensure that $\hat{\varphi}_{k+1}(\mathcal{T}_h(t_k)) \in \mathcal{D}_h(\mathcal{T}_h(t_k)$, meaning $\hat{\varphi}_{k+1}$ is still orientation preserving.

Let us first consider the functional $\mathcal{G}_{h,0,k}$ and visually examine the meshes in Figure 5.5. As the functional $\mathcal{G}_h$ measures the energy of the deformation from $\hat{\mathcal{T}}_h(t_k)$ to $\Psi(\mathcal{T}_h(t_k))$, the quality the reference mesh $\hat{\mathcal{T}}_h(0)$ plays a crucial role. It can be seen in Figure 5.4, the starting reference meshes are nonuniform, limiting the quality that can be achieved by minimising $\mathcal{G}_{h,0,k}$. Especially in the regions where the boundary is convex at $t = 0$ and loses this property, strong compression of cells and high degrees of anisotropy can be observed. Worse still, the output of one time step is used as the reference domain for the next time step. On all refinement levels, the computation breaks down well before $t = \bar{t}$ because of the angle constraint, and this breakdown occurs sooner on higher levels of refinement (see Figure 5.9 (i)). One explanation for this is that the boundary deformation is more severe relatively to the cell sizes and edge lengths on the finer meshes. As opposed to this, the evolution of the average mesh quality heuristic $\mathcal{Q}_a$ is nearly the same on all levels of refinement (see Figure 5.10 (i)) and within good bounds, which is of no consequence because of the violation of the angle constraint in at least one cell. Also note how a kind of "mesh convergence" can be observed in the plot of $\mathcal{Q}(\Psi^*_{h,0,k})$ in Figure 5.9 (i). In summary, the results are not satisfactory.

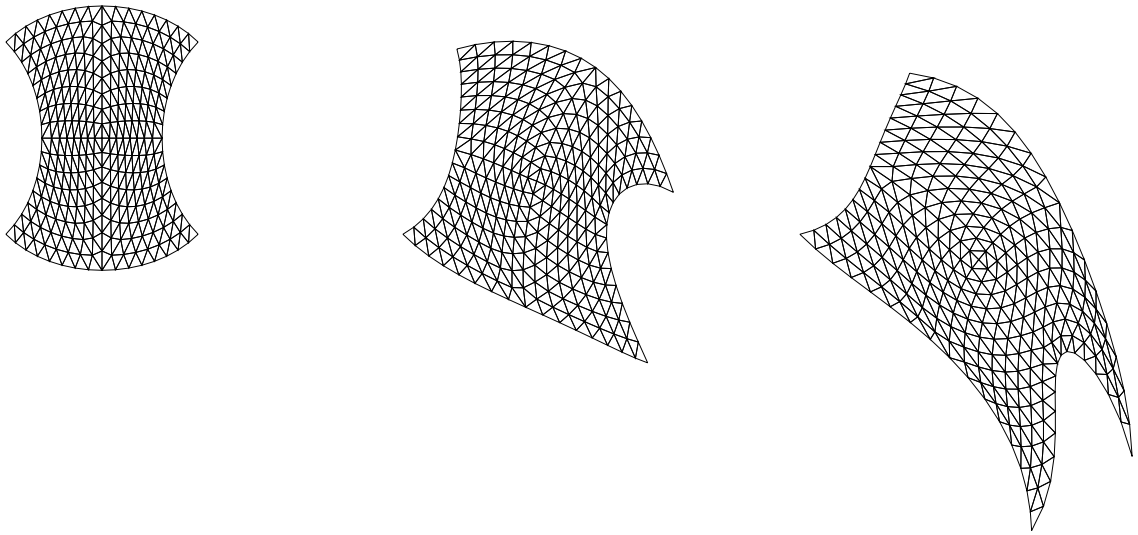
Similar to Section 3.4.2, we can expect to improve the method by using a moving reference domain and mesh. Indeed, the functional $\mathcal{G}_{h,k,k+1}$ offers much better results, as can be seen in Figure 5.6. For the time step $t = 0.25$, there is still no visual difference, but this changes towards the end of the time interval. The minimisers of $\mathcal{G}_{h,k,k+1}$ do not show the compression of cells near the lower boundary, and the angle constraint is far from being violated on all refinement levels (see Figure 5.8 (ii)). Interestingly, the evolution of the average mesh quality heuristic is nearly identical to the case with the fixed reference domain (compare Figure 5.10 (i) and Figure 5.10 (ii)). The higher computational cost of reassembling the linear system in every time step clearly pays off here, as the deformations in each time step are "small".

Last, we consider the minimisers of $\mathcal{F}_h$ to see if this more expensive method offers significant advantages. One advantage certainly is that we can prescribe a cell size distribution directly. In



(i) $t = 0$. (ii) $t = 0.25$. (iii) $t = 0.5$.

Figure 5.6: $\Psi^*(\mathcal{T}_h(t))$ for different time steps t, refinement level three.

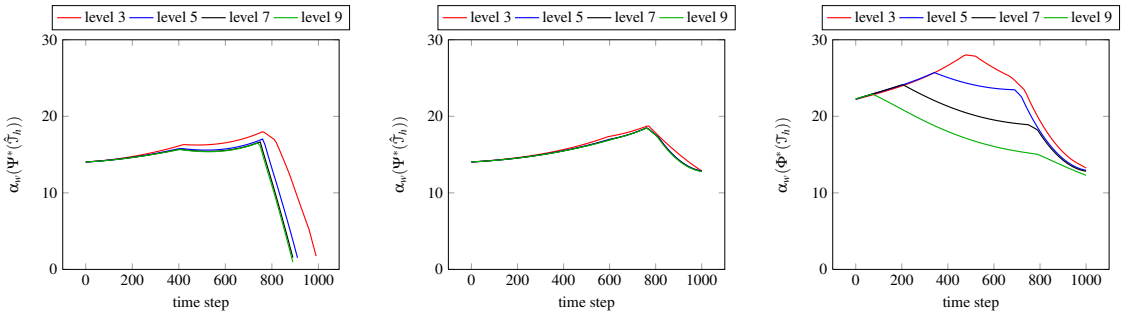


(i) $t = 0$ (ii) $t = 0.25$ (iii) $t = 0.5$

Figure 5.7: Moving mesh optimised using a hyperelasticity based functional, refinement level three.

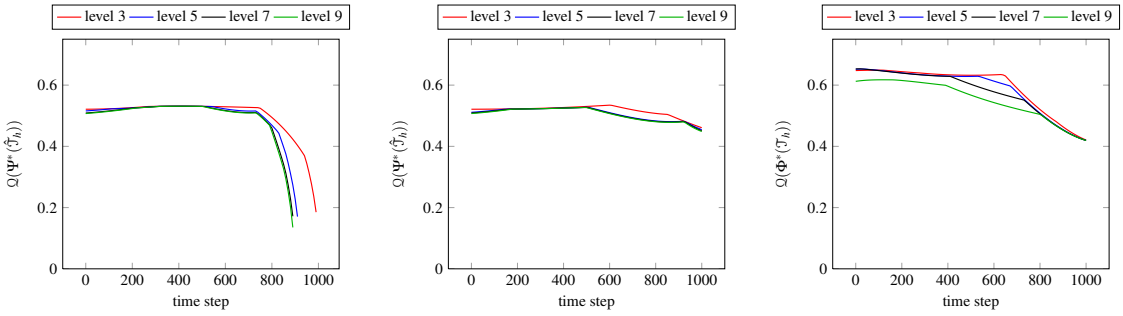
this case we prescribe a uniform distribution of the domain's volume over all cells, which can be seen in Figure 5.7. The meshes obtained by minimising the functional $\mathcal{F}_h$ show similar stability as the meshes obtained by minimising $\mathcal{G}_{h,k,k+1}$ and do not show any deterioration on finer meshes. Again the independence on any reference domain appears to be an advantage. Even near the lower boundary (which starts out convex but loses this property), only moderate cell compression can be observed. This is due to the fact that the functional $\mathcal{F}_h$ offers direct control over $\det(\nabla\Phi)$ and $1/\det(\nabla\Phi)$. The minimal angles stay bounded well above $\tilde{\alpha}$ (see Figure 5.8 (iii)), and in fact it is possible to continue the computations up to $t > 0.94$ without the angle constraint being violated due to the boundary deformation. The compression of the mesh in x_2 direction accompanied by the stretching in x_1 direction appears to pose no difficulty. Apart from the expansion of the domain and

the changes to cell sizes because of the boundary deformation, even the cell size equidistribution remains intact. The average cell quality heuristic $\mathcal{Q}_a$ is significantly higher than for the functionals $\mathcal{G}_{h,l,k+1}$ (compare Figure 5.10 (ii) and Figure 5.10 (iii)).



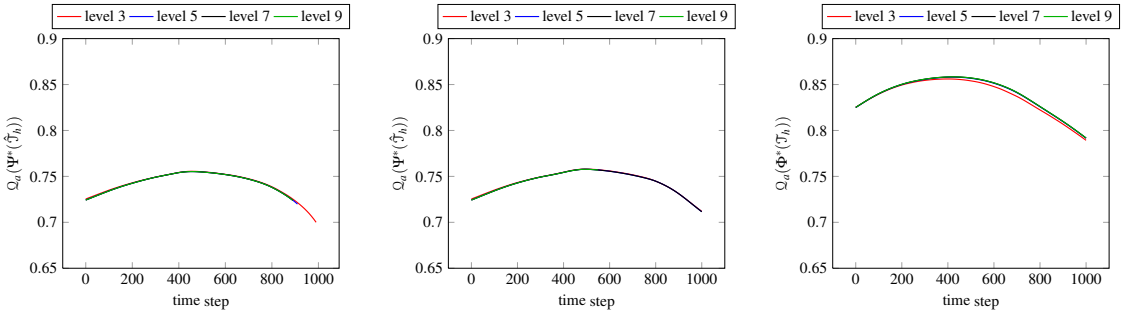
(i) Minimisers of $\mathcal{G}_{h,0}$. (ii) Minimisers of $\mathcal{G}_{h,k,k+1}$. (iii) Minimisers of $\mathcal{F}_h$.

Figure 5.8: Worst angle α_w over time.



(i) Minimisers of $\mathcal{G}_{h,0}$. (ii) Minimisers of $\mathcal{G}_{h,k,k+1}$. (iii) Minimisers of $\mathcal{F}_h$.

Figure 5.9: Minimal shape quality heuristic $\mathcal{Q}$ over time.



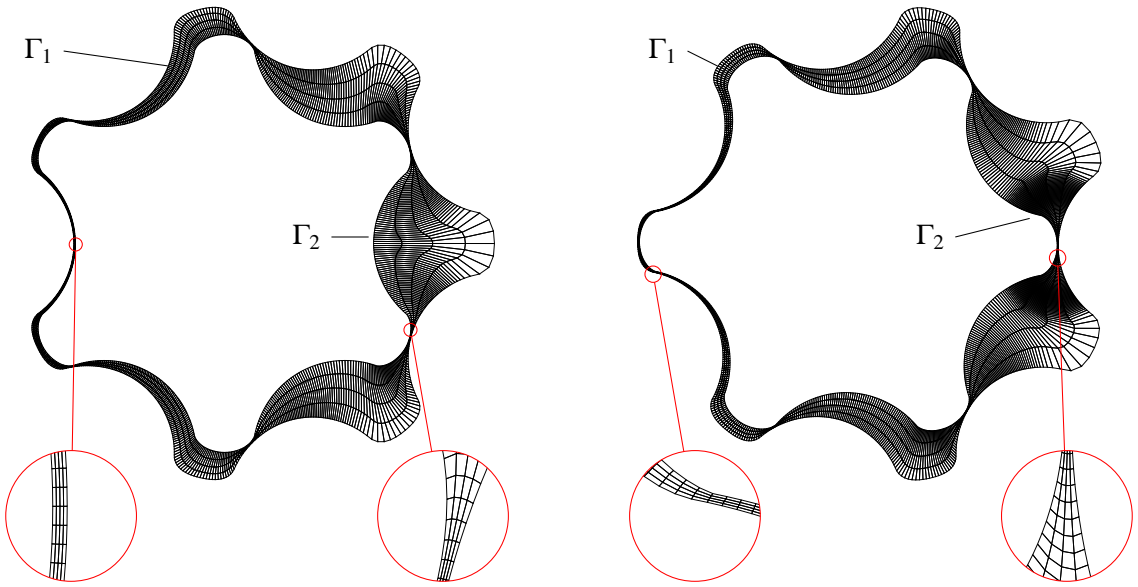
(i) Minimisers of $\mathcal{G}_{h,0}$. (ii) Minimisers of $\mathcal{G}_{h,k,k+1}$. (iii) Minimisers of $\mathcal{F}_h$.

Figure 5.10: Average shape quality heuristic $\mathcal{Q}_a$ over time.

As opposed to the example in Section 3.4, the quadratic mesh quality functional $\mathcal{G}_{h,k,k+1}$ is sufficient for the case considered in this section. As before, the functional $\mathcal{G}_{h,0,k}$ is not. In a case like this, using the far more costly functional $\mathcal{F}_h$ still provides some decisive advantages, as being able to prescribe a cell size distribution and a significantly higher average mesh quality, although these benefits might only warrant the cost for special applications.

5.4.2. Rotating excentric screws

This example is inspired by the geometry of a micro gear pump. Two excentrically placed screws with six (inner) and seven teeth (outer) rotate at different angular velocities, which is a geometric requirement, as the screws must not touch. 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. In this section, the computations for hypercubes were carried out on a suitable $2d$ part of the mesh and then extruded, whereas the computations for simplex meshes were done on a $2d$ mesh.



(i) $t = 0$. (ii) $t = 0.065$.

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

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

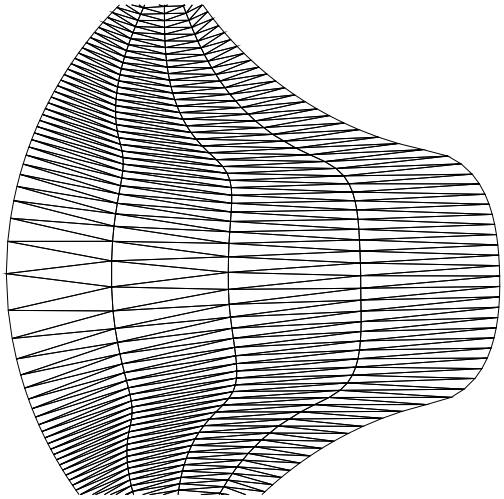
greatly varies between

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

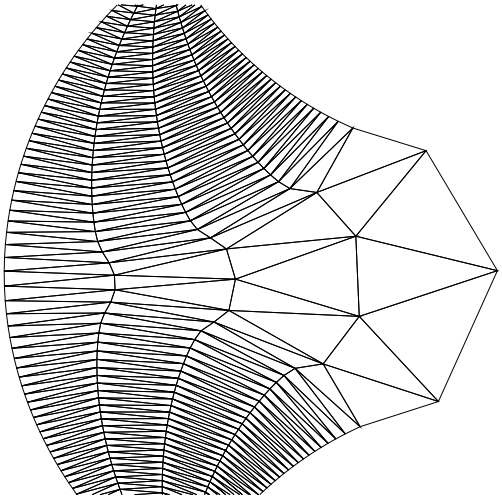
This strong anisotropy already implies low interior angles between edges in the case of simplex meshes and makes the problem quite challenging. It also means that the optimal scale of a cell K needs to be chosen according to its distance from Γ_1, Γ_2, a this changes according to the rotation and cells 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 quickly leads to mesh deterioration, especially when using the functional $\mathcal{G}_h$ (see Figure 5.12 (i)).

Instead, a unilateral boundary condition of place (see Definition 3.1) needs to be used on one or even both boundaries. This is somewhat similar to a slip boundary condition for the Stokes or



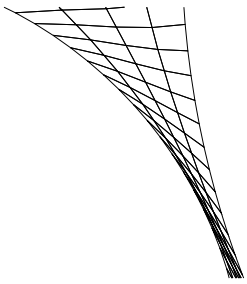
(iii) Unilateral boundary condition of place on Γ_2, displacement boundary condition on Γ_1.



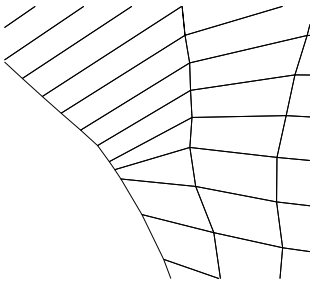
(iv) Unilateral boundary condition of place on Γ_1 and Γ_2.

Figure 5.11: Mesh at $t = 4\text{e}{-3}$ with different sets of boundary conditions.

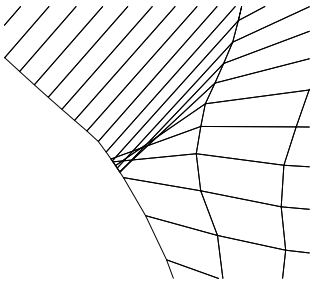
Navier-Stokes equations, and can be implemented in a similar fashion. If this boundary condition is enforced at both boundaries, it means the cells can move freely throughout the mesh, although it might be advantageous to rule out rigid body rotations in the vertex movement (e.g. by enforcing a displacement 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, but it also means that the discretisation of the boundary loses resolution, see Figure 5.11.



(i) Deteriorated mesh at $t = 0.0246$ when using displacement boundary conditions only.



(ii) Unilateral boundary condition of place, before deterioration due to projection.



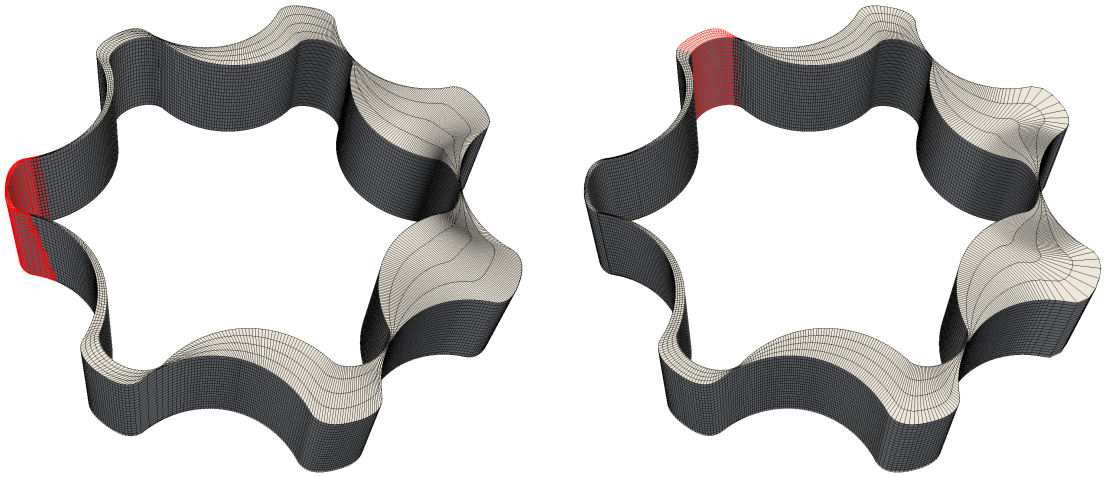
(iii) Unilateral boundary condition of place, intersections at the boundary due to projection.

Figure 5.12: Different types of mesh deterioration when using the functional $\mathcal{G}_h$.

However, if the boundary is curved, the outer unit normal is not constant, which means that even the functional $\mathcal{G}_h$ becomes nonlinear due to the boundary condition. As this is a loss of its significant advantage, one could try to linearise the problem by not updating the outer unit normal over the course of the solver iterations. As the curvature of the boundaries is quite strong, this quickly lead to mesh deterioration by intersections at the boundary after projecting the solution to the space satisfying the unilateral boundary condition of place (see Figure 5.12 (ii) and Figure 5.12 (iii)).

Using the parameters

$$\omega = 2\pi, \bar{t} = 1, \delta_t = 1\text{e}{-4}, \tag{5.9}$$



(i) Case 1: Unilateral boundary condition of place on Γ_2, displacement boundary condition on Γ_1.

(ii) Case 2: Unilateral boundary condition of place on Γ_1 and Γ_2.

Figure 5.13: $3d$ mesh for both cases at $t = 0.5$ with the same set of cells marked.

means that the outer screw performs one full rotation on the time interval $[0,\bar{t}]$, while the inner screw rotates by $\frac{7}{6}2\pi$. For $\mathcal{F}_h$, the parameters

$$c_f = 0.01, \delta_r = 1\text{e}{-8}, p_d = 2, c_d = 2\delta_r^2 + (\delta_r^2+2)\sqrt{\delta_r^2+1}+2$$

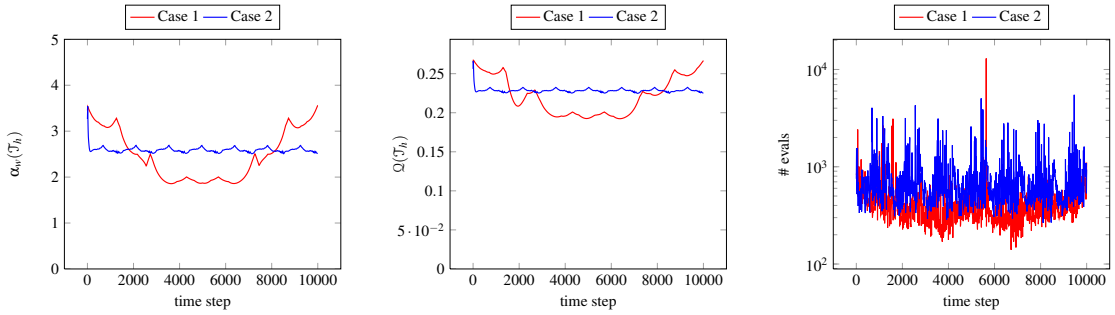
were used, with c_f chosen small to allow for a greater change in edge lengths. In the following, we will regard the following cases

1. Displacement boundary condition on Γ_1 and unilateral boundary condition of place on Γ_2.
2. Unilateral boundary condition of place on both Γ_1 and Γ_2.

As the domain forces a high degree of anisotropy for the mesh's cells, we cannot expect any significant improvement in the quality over the initial mesh. The realistic goal is to keep the quality indicator $\mathcal{Q}(\mathcal{T}_h)$ (or the minimum angle in the case of simplex meshes) somehow bounded from below and the mesh from totally distorting, e.g. by violating the orientation preserving constraint $\det(\nabla\Phi*) > 0$, which does not seem possible using the functional $\mathcal{G}_h$.

Because of the large deformations and high ratio of compression and expansion of cells, this test case is a good test for the robustness of the method. In Figure 5.18, some cells have been marked to show they move freely in the domain due to unilateral boundary conditions of place on both boundaries. It can be observed how the cells move from a region with small gap width to a region of large gap width over time, partially "moved along" with the boundaries. This means that, after a full rotation of the outer boundary, cells will in general not be located at the same points where they were at $t = 0$. In Figure 5.13, the same set of cells was marked for the meshes resulting from case 1 and case 2 for a comparison at $t = 0.5$. For case 2 it can be seen that the marked cells moved freely through the mesh, so they occupy a different region in $\Phi^*(\mathcal{T}_h(0.5))$ than in case 1.

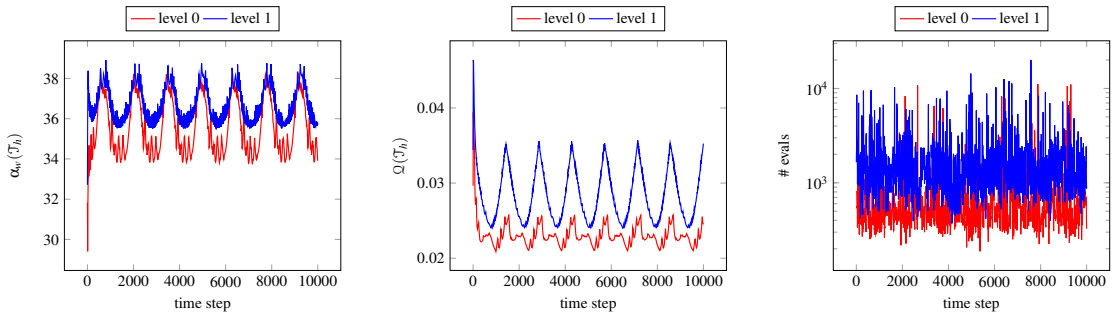
Note that the placement of Γ_1 and Γ_2 relative to each other is somewhat periodic, as they occupy the same region of space with a period of $\frac{1}{7}$, but with different parts. This means the geometric situation is "similar" every $\frac{10000}{7} \approx 1429$ time steps. However, the vertex distribution on $\Gamma_{1,2}$ is not completely uniform, causing the edges to have different lengths.



(i) Worst angle over time. (ii) Quality indicator over time. (iii) Number of functional evaluations per time step.

Figure 5.14: Comparison of different quantities for different sets of boundary conditions, simplices, $l = 0$.

First, let us examine the plots in Figure 5.14, comparing the results of case 1 and 2 for a simplex mesh on refinement level $l = 0$. The coarsest level is already sufficient to resolve the geometry, and global regular refinement will not change the general difficulty of cell compression and expansion. In the plots of α_w and $\mathfrak{Q}$ in Figure 5.14, we can observe the aforementioned periodicity for case 2 because of the unilateral boundary conditions of place on both Γ_1 and Γ_2. The vertices can move freely on those boundaries so that the minimisers Φ^* of $\mathcal{F}$ are able to create these "similar" states. For case 1, we can still see the local extrema in both plots to correspond to these time steps, but there is no periodic structure. Because the different edge lengths on $\Gamma_{1,2}$ (as mentioned above) remain the same on Γ_2 for all t due to the displacement boundary condition, the states are not very "similar" even if the geometric situation is.



(i) Worst angle α_w over time. (ii) Quality heuristic $\mathfrak{Q}$ over time. (iii) Number of functional evaluations per time step.

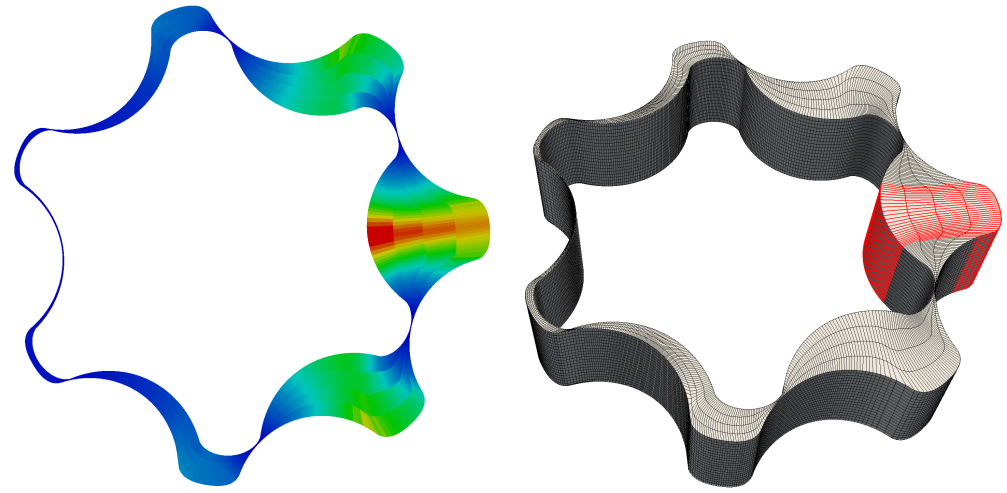
Figure 5.15: Comparison of different quantities for different levels of refinement, hypercubes, case 2.

Nonetheless, the meshes $\Phi^*(\mathcal{T}_h(t))$ do not deteriorate even in this situation of extreme compression and expansion. The quality heuristic $\mathfrak{Q}$ remains in an acceptable range for both cases and we cannot expect more, based on the initial mesh.

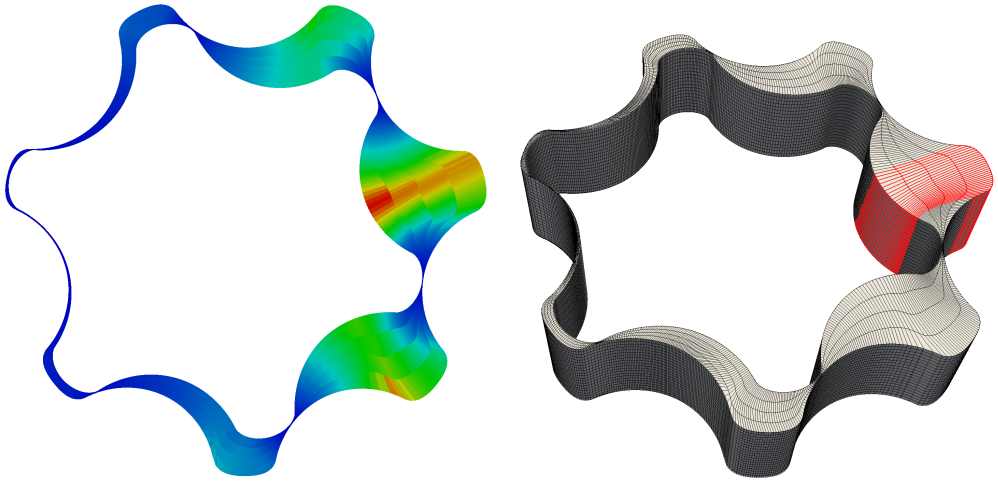
For hypercube meshes, I will only discuss case 2. For this cell type, the degree of anisotropy directly enters the shape quality heuristic $\mathfrak{Q}$ and we already know the gap width varies between 0.02 and 1.15. For $l = 1$, the edges on $\Gamma_{1,2}$ have lengths in the range of $O(1\mathrm{e}{-2})$, while the gap is split into four layers of elements. This already results in a ratio of $\frac{h_{\min}}{h_{\max}} \approx \frac{1\mathrm{e}{-2}}{\frac{1}{4}1.15} \approx 0.035$, so we cannot expect much more for $\mathfrak{Q}$, which is confirmed in Figure 5.15 (ii). The gap width δ_g is depicted in Figure 5.17, while the resulting cell size can be found in Figure 5.16. The use

of unilateral boundary conditions of place on $\Gamma_{1,2}$ can reduce this ratio in some regions, but not globally (see Figure 5.16 for case 1 and Figure 5.17 for case 2). But even with the freely moving cells, the method can keep the angles from becoming too small.

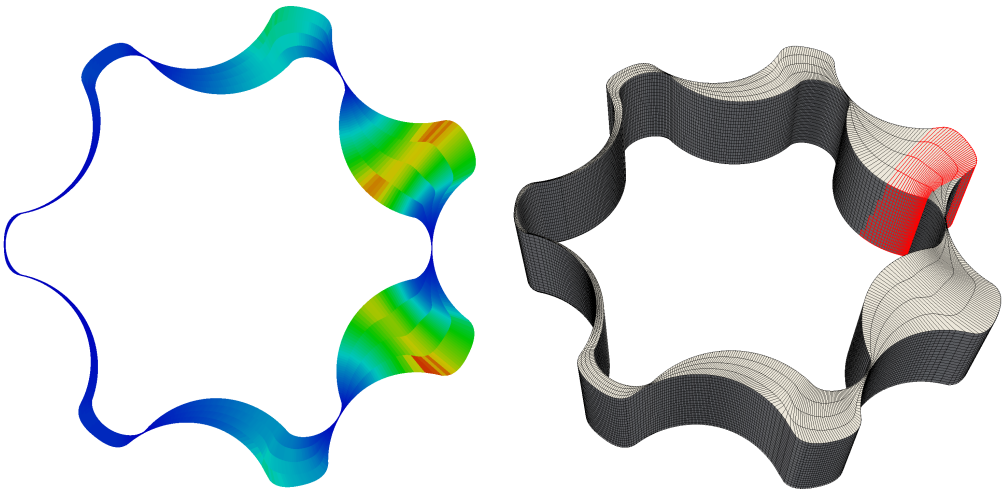
This example was chosen to demonstrate the robustness of the method. Visual examination of the resulting meshes shows that they remain intact and of acceptable quality, but that they may be too coarse for practical flow simulations in the expanded regions. Here, some manual work, or fine tuning of the mesh quality functional may be needed. Examining the resulting meshes also inspires the idea of performing the mesh optimisation on the coarsest mesh that resolves the geometric features and then refining and adapting the mesh as in Section 5.3 to obtain a finer mesh that resolves the features of the original PDE.



(i) $t = 0.005$

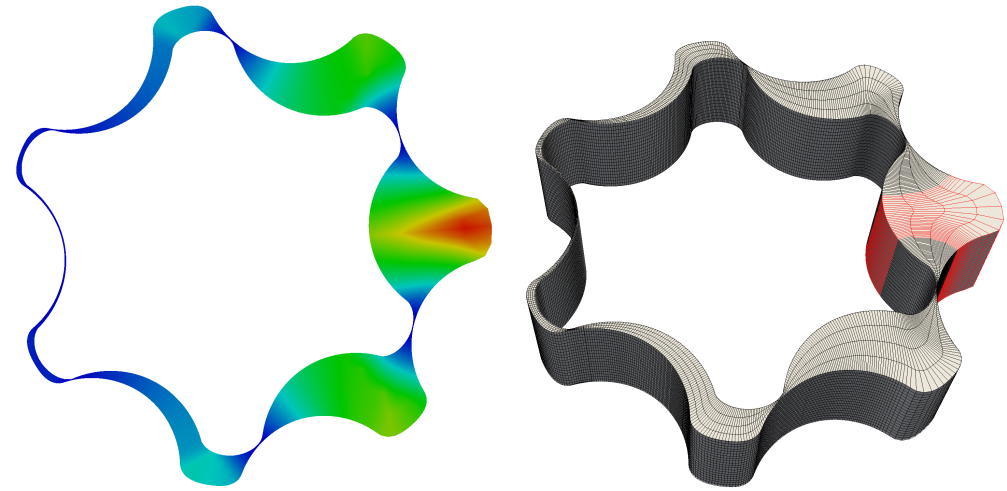


(ii) $t = 0.03$

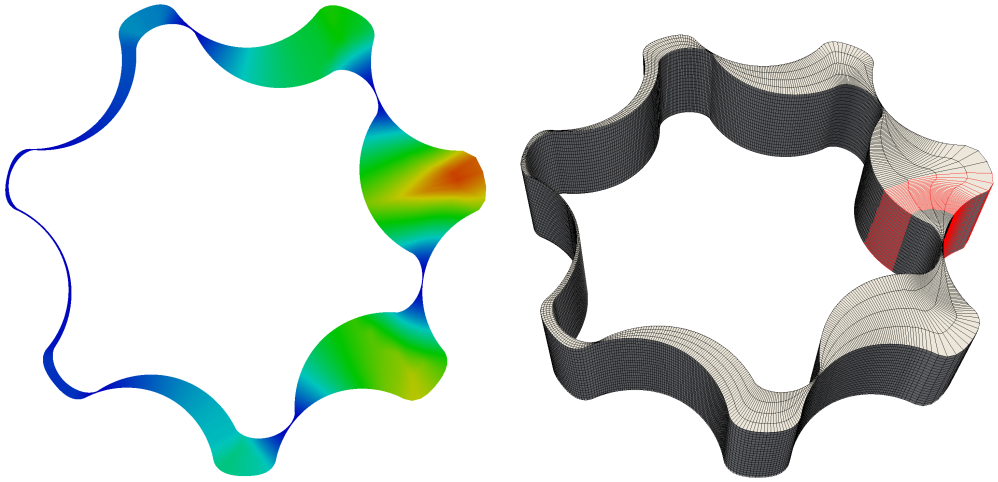


(iii) $t = 0.065$

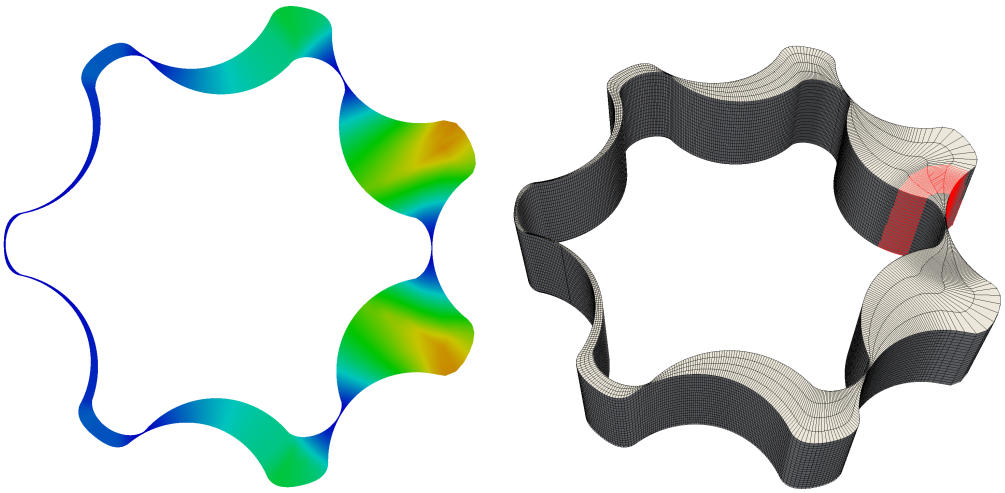
Figure 5.16: Case 1: The mesh with $\forall K \in \mathcal{E}^2(\Phi^*(\mathcal{T}_h)) : \mathrm{vol}_2(K) \in [5\mathrm{e}{-5}, 0.01]$ (left) and the extruded mesh with some marked cells (right).



(i) $t = 0.005$

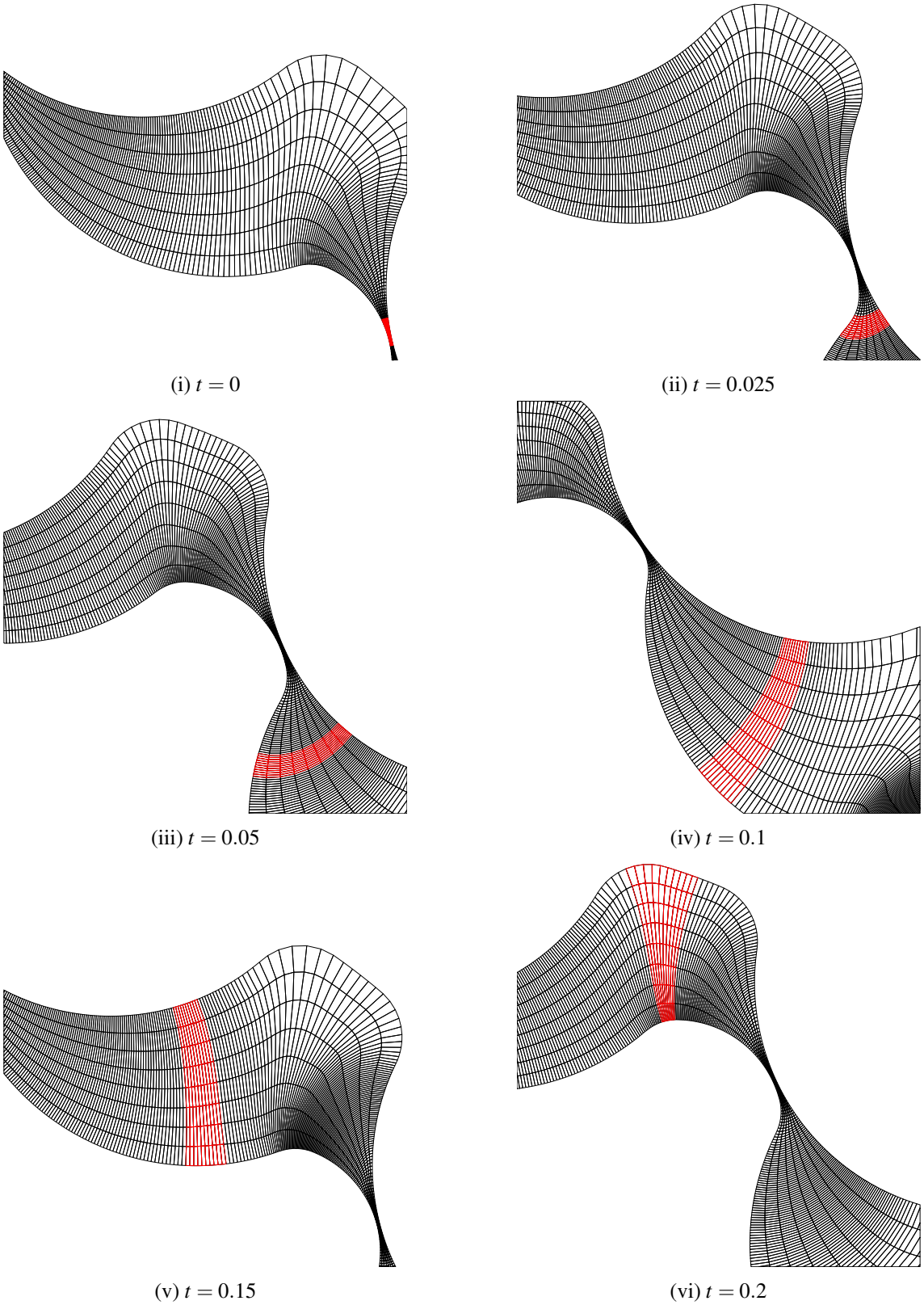


(ii) $t = 0.03$



(iii) $t = 0.065$

Figure 5.17: Case 2: The mesh with $\delta_g \in [0.02, 1.15]$ (left), and the extruded mesh with some marked cells (right).



(i) $t = 0$ (ii) $t = 0.025$

(iii) $t = 0.05$ (iv) $t = 0.1$

(v) $t = 0.15$ (vi) $t = 0.2$

Figure 5.18: Part of the domain with marked cells in case 2 at different time steps on refinement level 2.

5.5. *r*-adaptivity

In this section I will present results obtained by minimising functionals of the family from Section 3.5 when using a mesh concentration function to influence the cell size distribution.

The choice of the mesh concentration function (see Section 3.5.4) is crucial for concentrating cells in the region of interest and is a concept similar to using monitor functions (see [HR11, Chapter 5]), although setting an optimal scale is more direct and allows a quantitative measurement of the cell size distribution defect $\mathcal{S}$.

For simplicity, I will restrict myself to a very simple class of mesh concentration functions defined by

$$c(K) = f(\operatorname{dist}(s_{\Phi^*(K)}, \Gamma)), f : \mathbb{R} \to \mathbb{R}_+, f(t) = (c_1 + |t|)^{c_2}, c_{1,2} > 0 \tag{5.10}$$

for some surface $\Gamma \subset \mathbb{R}^d$, possibly with $\Gamma \not\subset \Omega_h$.

The parameter c_1 defines the relative minimum value of the function f and cannot be chosen too small so that the prerequisites of Theorem 3.11 (most notably the coerciveness) still hold. c_2 defines how quickly f changes away from Γ and the same constraints as for c_1 apply. Note that the relative optimal cell sizes λ and the optimal scales h are invariant to multiplying the concentration function with a constant.

A feasible way to determine $c_{1,2}$ for a specific situation is to first chose c_2 according to the characteristic we want the cell size distribution to have. For example, if the optimal cell size should increase quickly away from Γ, chose some $c_2 < 1$. If we want a whole region around Γ to consist of smaller cells, chose some $c_2 > 1$. After fixing c_2, fix some desired ratio r between the smallest and the largest cell size. After that, one can compute

$$c_1 = \frac{d_{\min} + d_{\max} r^{1/c_2}}{1 - r^{1/c_2}}, \quad d_{\min} = \inf_{x \in \Omega_h} \operatorname{dist}(x, \Gamma), \quad d_{\max} = \sup_{x \in \Omega_h} \operatorname{dist}(x, \Gamma). \tag{5.11}$$

In the following, denote

$$\lambda_{\min}(\Phi^*(\mathcal{T})) = \min_{K \in \mathcal{T}} \lambda(K), \qquad \lambda_{\max}(\Phi^*(\mathcal{T})) = \max_{K \in \mathcal{T}} \lambda(K),$$
$$\operatorname{vol}_{\min}(\Phi^*(\mathcal{T})) = \min_{K \in \mathcal{T}} \operatorname{vol}(K), \qquad \operatorname{vol}_{\max}(\Phi^*(\mathcal{T})) = \max_{K \in \mathcal{T}} \operatorname{vol}(K).$$

5.5.1. Moving circle

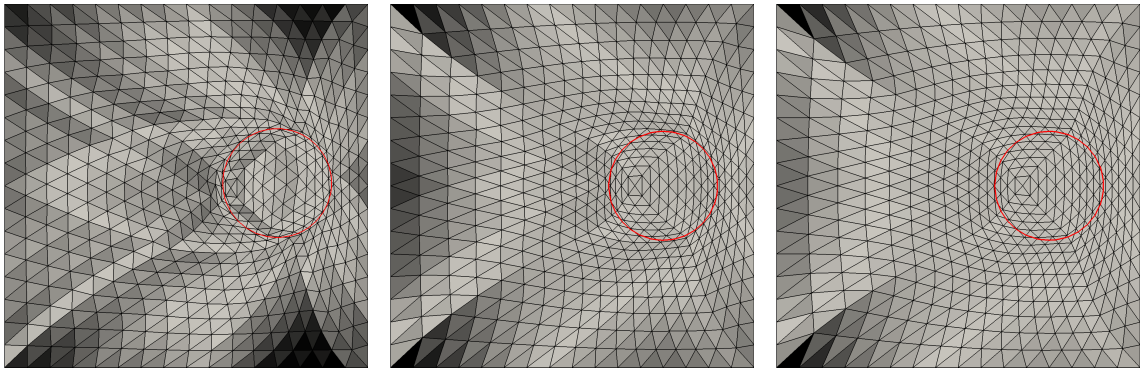
In this section, we consider the unit square $\Omega = \Omega_h = [0,1]^2 \subset \mathbb{R}^2$ with the surface

$$\Gamma = \Gamma(t) := \partial B_{\frac{3}{20}}(x_c(t)), \quad x_c(t) = \frac{1}{4} \begin{pmatrix} 2 + \cos(t) \\ 2 + \sin(3t) \end{pmatrix}.$$

and use

$$c_f = 0.01, \delta_r = 1\text{e}{-8}, p_d = 2, c_d = 2\delta_r^2 + (\delta_r^2 + 2)\sqrt{\delta_r^2 + 1} + 2$$

as parameters for $\mathcal{F}$. We will consider the time interval $[0,1]$ and compute 100 time steps with $\delta_t = 1\text{e}{-2}$. On $\partial\Omega_h$ we enforce a no displacement boundary condition. The setting is chosen to be particularly simple: The distance function is smooth (except at the point $x_c(t)$) and the set inscribed by Γ is convex. Moreover, $c_f = 1\text{e}{-2}$ is chosen so that differing edge lengths have less influence on the functional value (this might be interpreted as the hyperelastic material used being less "stiff" with regard to size changes of one dimensional entities), as we are less interested in maximising $\mathcal{Q}$ in this setting, as long as it stays bounded from below.



(i) $c_2 = \frac{1}{2}$,
$\mathcal{S} \in [1.54\text{e}{-8}, 1.14\text{e}{-4}]$.

(ii) $c_2 = 1$,
$\mathcal{S} \in [9.90\text{e}{-8}, 1.81\text{e}{-4}]$.

(iii) $c_2 = 2$,
$\mathcal{S} \in [5.7\text{e}{-8}, 2.85\text{e}{-4}]$.

Figure 5.19: $\Phi^*(\mathcal{T}_{h,0})$ for different values of c_2 with Γ in red, colouring by $|\lambda(K) - \text{vol}(K)|$ from white to black.

The goal is to examine the influence of the choice of the concentration function first and then see how the mesh quality heuristics behave over time as Γ moves. The solver will be a ALGLIB's lBFGS solver with absolute tolerance $\varepsilon_a = 1\text{e}{-8}$, step length tolerance $\varepsilon_s \approx 2.204\text{e}{-16}$ and lBFGS dimension 10. In this example, the solver almost always stopped due to the step length criterion with a relative residual in the order of $O(10^{-3})$ to $O(10^{-6})$.

c_1	7.5e−3	8.3e−2	3.5e−1
c_2	$\frac{1}{2}$	1	2
$\lambda_{\min}(\Phi^*(\mathcal{T}_{h,0}))$	2.01e−4	2.95e−4	3.73e−4
$\lambda_{\max}(\Phi^*(\mathcal{T}_{h,0}))$	1.96e−3	2.78e−3	3.30e−3
$\text{vol}_{\min}(\Phi^*(\mathcal{T}_{h,0}))$	1.98e−4	2.85e−4	3.60e−4
$\text{vol}_{\max}(\Phi^*(\mathcal{T}_{h,0}))$	2.02e−3	2.75e−3	3.23e−3
$\mathcal{Q}(\mathcal{T}_{h,0})$	7.60e−1	7.60e−1	7.60e−1
$\mathcal{Q}(\phi^*(\mathcal{T}_{h,0}))$	3.92e−1	6.02e−1	5.90e−1
$\mathcal{S}(\mathcal{T}_{h,0})$	2.98e−1	4.01e−1	4.57e−1
$\mathcal{S}(\phi^*(\mathcal{T}_{h,0}))$	2.35e−2	2.33e−2	2.28e−2

Table 5.8: Different values for c_2 and associated c_1.

First, let us examine how the choice of concentration function influences $\Phi^*(\mathcal{T}_{h,0})$. For visualisation reasons, a refinement level of four was chosen, resulting in 1024 cells and 990 DoF. We first fix a ratio $r = \frac{1}{10}$ and compute different sets of parameters $c_{1,2}$. Some data for $c_2 \in \left\{\frac{1}{2}, 1, 2\right\}$ including the resulting cell sizes vol can be found in Table 5.8 and visualisations of $\Phi^*(\mathcal{T}_{h,0})$ in Figure 5.19.

In Table 5.8, we can see the general behaviour that the application of Φ^* reduces the cell size distribution defect $\mathcal{S}$ roughly by one order of magnitude, at the cost of decreasing the shape quality indicator $\mathcal{Q}$. In the case $c_2 = \frac{1}{2}$, the distance function varies somewhat sharply in the direct vicinity of Γ, so the cells in that region are of lower shape quality. This is not the case for $c_2 \in \{1, 2\}$, and Table 5.8 confirms the visual impression from Figure 5.19 (ii) and Figure 5.19 (iii) that $\mathcal{Q}$ does not

decrease significantly with the application of Φ^*.

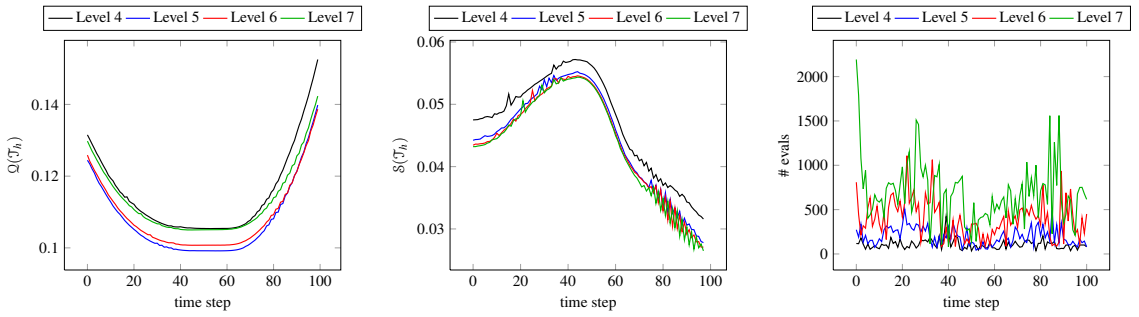
l	$\overline{\#\text{ its}}$	$\overline{\#\text{ evals}}$	$\overline{\mathcal{Q}(\Phi^*)}$	$\overline{\alpha_{\min}(\Phi^*)}$	$\overline{\mathcal{S}(\Phi^*)}$
4	21	107	$1.24e{-1}$	28.44	$4.72e{-2}$
5	42	200	$1.17e{-1}$	27.49	$4.47e{-2}$
6	87	391	$1.18e{-1}$	27.61	$4.40e{-2}$
7	172	686	$1.22e{-1}$	28.12	$4.37e{-2}$

Table 5.9: Quantities for the moving circle on multiple levels, $c_2 = 2$, hypercube case.

Figure 5.19 also shows that the higher values $\mathcal{S}(K)$ naturally occur near the boundary $\partial\Omega_h$ where a no displacement boundary condition is enforced. Enforcing a unilateral boundary condition of place instead could help to reduce these errors, but only in cases where the optimal scales vary significantly for cells lying at the corresponding boundary. However, this is not the case in the current setting.

For the sake of brevity, only the cases $c_2 \in \left\{\frac{1}{2}, 2\right\}$ will be discussed for the whole time interval and for multiple levels of refinement.

First, consider the hypercube case with $c_2 = 2$. Table 5.9 shows the averages of the number of solver iterations, functional evaluations and mesh quality heuristics over all time steps on multiple levels of refinement. The averages of the numbers of iterations and evaluations show the same qualitative behaviour as in the case of uniform size distributions, where these numbers double with each level of refinement. All three mesh quality heuristics remain nearly constant over all levels of refinement.



(i) Shape quality indicator over time.

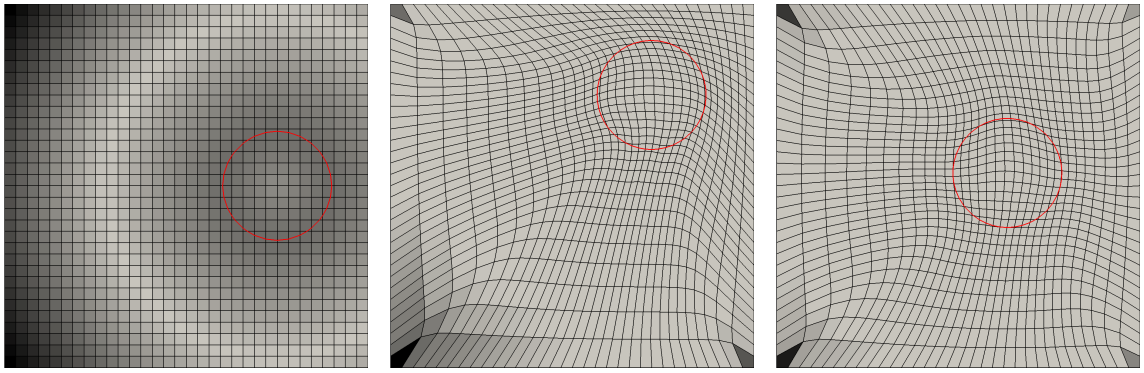
(ii) Size quality defect over time.

(iii) Needed number of functional evaluations per time step.

Figure 5.20: $c_2 = 2$, hypercube case.

The results for the case $c_2 = \frac{1}{2}$ are similar qualitatively, but they can be used to illustrate a problem that occurs in practice. Since the minimiser Φ^* is not unique and a local minimiser only, the discrete solution obtained by the minimisation process might not be a very useful one, even in this comparably simple setting. In such a case, usually the nonlinear solver stops due to the step length criterion, which means the local information at the current state is insufficient to make any further progress. This effect can be observed in Figure 5.23, where the minimisers Φ^* are not very useful up until $t = 0.03$. There, the solver converges to a better local minimum and continues to do so for all subsequent time steps.

Behaviour like this can be observed when the situation is "badly conditioned", meaning the problem is very hard numerically. Here, we add up the difficulty of strongly varying optimal scales



(i) $\mathcal{T}_{h,0}$, $\mathcal{S} \in [4.66\text{e}{-6}, 1.55\text{e}{-3}]$.

(ii) $t = 0.5$, $\mathcal{S} \in [2.00\text{e}{-9}, 9.92\text{e}{-4}]$.

(iii) $t = 1$, $\mathcal{S} \in [2.00\text{e}{-9}, 9.92\text{e}{-4}]$.

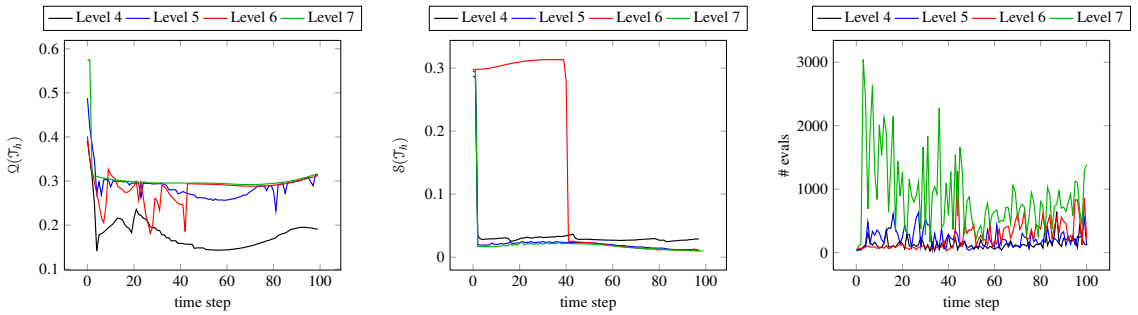
Figure 5.21: Hypercube mesh at different time steps, $c_2 = 2$.

l	$\overline{\#\text{ its}}$	$\overline{\#\text{ evals}}$	$\overline{\mathcal{Q}(\Phi^*)}$	$\overline{\alpha_{\min}(\Phi^*)}$	$\overline{\mathcal{S}(\Phi^*)}$
4	23	138	$1.91e{-1}$	32.73	$3.93e{-2}$
5	50	220	$2.98e{-1}$	41.22	$3.04e{-2}$
6	56	251	$2.93e{-1}$	43.25	$1.41e{-1}$
7	279	885	$3.14e{-1}$	45.73	$2.28e{-2}$

Table 5.10: Quantities for the moving circle on multiple levels, $c_2 = \frac{1}{2}$, hypercube case.

h while still using isotropic reference cell with the inherently higher polynomial degree of $\mathcal{F}$ in the case of hypercubes.

Unfortunately, this behaviour occasionally makes the method unreliable depending on the situation and is likely inherent, since the solvers usually converge to "close" local minima. The behaviour persists on all levels of refinement, especially on refinement level six (see Figure 5.22) and different solvers (e.g. NLCG and NLSD-lBFGS). It might be possible to recognise these situations and to modify the starting point accordingly, but this is beyond the scope of this work.



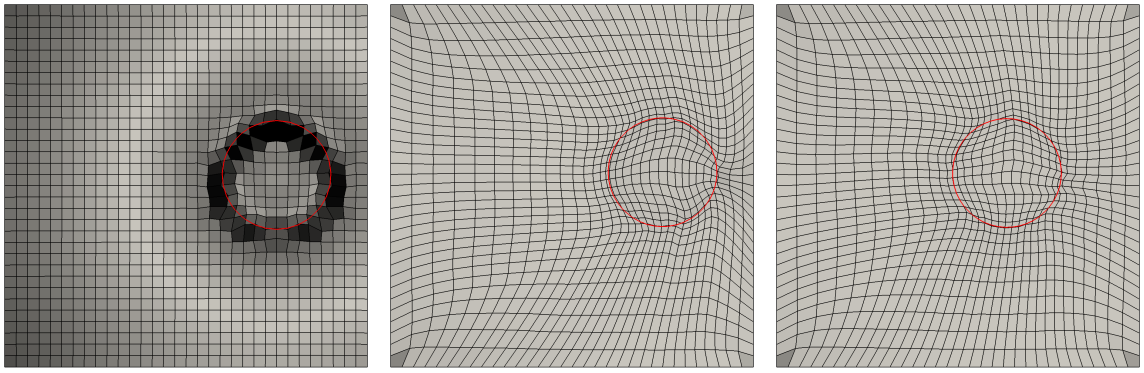
(i) Shape quality indicator over time.

(ii) Size quality defect over time.

(iii) Needed number of functional evaluations per time step.

Figure 5.22: $c = \frac{1}{2}$, hypercube case.

In the case of simplices, the behaviour does not seem to appear very frequently, which might be related to the problem being simpler in the sense that $\mathcal{F}'$ is piecewise constant. The minimisers



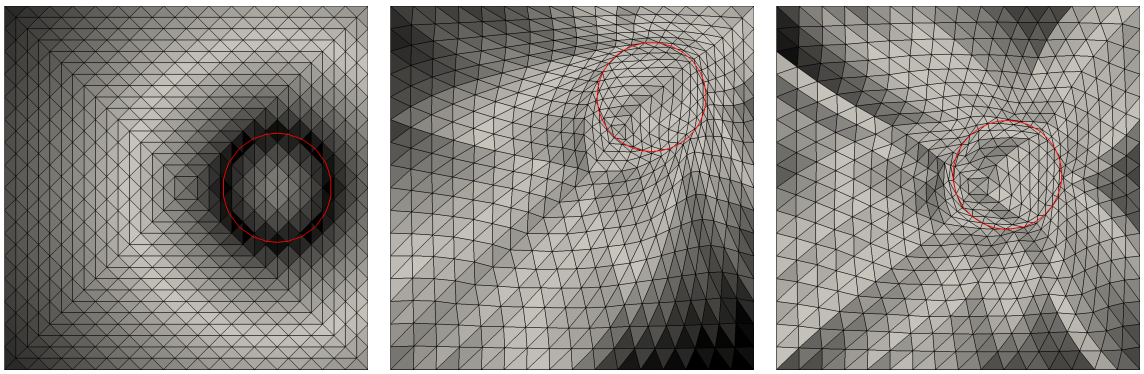
(i) $t = 0.03$, $\mathcal{S} \in [1e{-}6, 1e{-}3]$. (ii) $t = 0.04$, $\mathcal{S} \in [1e{-}6, 1e{-}3]$. (iii) $t = 1$, $\mathcal{S} \in [1e{-}6, 1e{-}3]$.

Figure 5.23: Hypercube mesh at different time steps, $c_2 = \frac{1}{2}$.

exhibit a qualitatively different behaviour as well, as can be seen in Table 5.11 and Figure 5.24. The numbers of both iterations and evaluations increase by a factor of two per level of refinement as well, but the cell size defect $\mathcal{S}$ decreases approximately by a factor of two as opposed to the hypercube case, where it remained nearly constant.

l	$\overline{\#\text{ its}}$	$\overline{\#\text{ evals}}$	$\overline{\mathcal{Q}(\Phi^*)}$	$\overline{\alpha_{\min}(\Phi^*)}$	$\overline{\mathcal{S}(\Phi^*)}$
3	85	349	$4.76e{-}1$	16.46	$3.98e{-}2$
4	190	574	$5.46e{-}1$	18.99	$2.05e{-}2$
5	520	1423	$5.70e{-}1$	20.21	$1.09e{-}2$
6	1275	3290	$5.80e{-}1$	20.93	$5.85e{-}3$
7	2710	7390	$5.77e{-}1$	21.39	$3.24e{-}3$

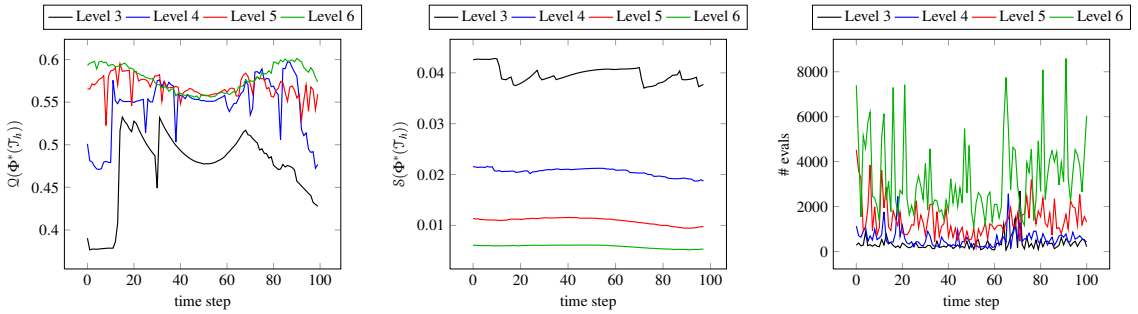
Table 5.11: Quantities for the moving circle on multiple levels, $c_2 = \frac{1}{2}$, simplex case.



(i) $\mathcal{T}_{h,0}$, $\mathcal{S} \in [1.16e{-}6, 8.10e{-}4]$. (ii) $t = 0.5$, $\mathcal{S} \in [1.54e{-}8, 1.14e{-}4]$. (iii) $t = 1$, $\mathcal{S} \in [1.54e{-}8, 1.14e{-}4]$.

Figure 5.24: Simplex mesh at different time steps, $c_2 = \frac{1}{2}$.

This can be observed for the case $c_2 = 2$ as well, so the corresponding results are not presented in detail.



(i) Shape quality indicator over time.

(ii) Size quality defect over time.

(iii) Needed number of functional evaluations per time step.

Figure 5.25: $c_2 = \frac{1}{2}$, simplex case.

5.5.2. Rotating shape

In this section, we consider the unit square $\Omega = [0,1]^2 \subset \mathbb{R}^2$ and a surface $\Gamma \subset \Omega$. The surface is given by the same composite cubic Bézier curve as the boundary of the domain in 5.4.1, scaled by $\frac{3}{4}$ and displaced by $\left(\frac{1}{4}, \frac{1}{4}\right)^T$ to fit into Ω. On the time interval $[0,1]$, Γ is rotated with an angular velocity of 2π so it performs one full rotation. We compute 100 time steps and use the parameters

$$\delta_r = 1\text{e}{-8}, p_d = 2, c_d = 2\delta_r^2 + (\delta_r^2 + 2)\sqrt{\delta_r^2 + 1} + 2$$

for the functional $\mathcal{F}$. We enforce no displacement boundary conditions and use the coefficients

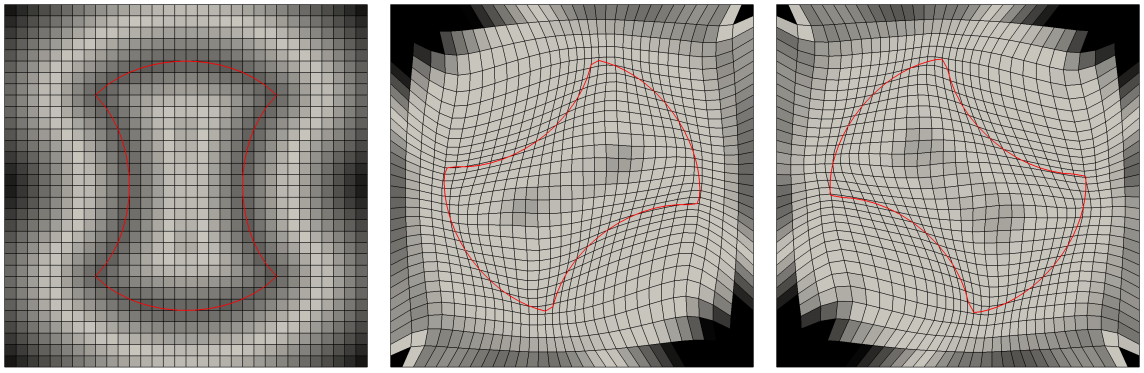
$$c_1 = 0.1, \quad c_2 = 1$$

for the mesh concentration function, which result in a ratio $\frac{\lambda_{\min}}{\lambda_{\max}} \approx \frac{1}{4}$. Because of $d_{\max} \approx 0.35$ this is moderate but still considerable. The goal is to see how the method behaves if the distance function is not smooth and to try the values of $c_f = 0.01, 0.1, 1$ to change how the functional penalises changes in the edge lengths, (which also allow varying degrees of anisotropy in the case of hypercubes). Additionally, the rotation of the shape means that the distance of point on Γ to the domain boundary $\partial\Omega_h$ varies.

The solver will be again ALGLIB's lBFGS solver with relative tolerance $\varepsilon_r = 1\text{e}{-8}$, step length tolerance $\varepsilon_s = 2.204\text{e}{-16}$ and lBFGS dimension 10.

As in Section 5.5.1, we can see in Figure 5.26 that the cell size distribution error $\mathcal{S}$ is high near the boundary, as is expected due to the no displacement boundary conditions. Another observation from Figure 5.26 (ii) and Figure 5.26 (iii) is that $\mathcal{S}$ is higher at the centre of Γ than near Γ itself, where λ is smallest. Also note that cells near Γ or $\partial\Omega_h$ are more anisotropic in the regions where λ takes its extremal values to account for the extremal optimal scales.

In Figure 5.27, the evolution of $\mathcal{Q}, \mathcal{S}$ and the number of functional evaluations per time step are depicted over time. $\mathcal{Q}$ starts at what appears to be a local maximum at $t = 0$ and has its first local minimum at approximately $t = 0.13$. From then on, this behaviour continues with $\delta_t = 0.25$ between two local maxima or minima. A similar evolution can be observed for $\mathcal{S}$, which is maximal at approximately the same time steps that $\mathcal{Q}$ is minimal and vice versa. This is simply due to the fact at around $t = 13$, $\text{dist}(\Gamma, \partial\Omega)$ is maximal because of the position of the concave parts of Γ with regard to the corners of Ω_h. At $t = 0.13$, we get $r = \frac{\lambda_{\min}(\mathcal{T}_{h,0.13})}{\lambda_{\max}(\mathcal{T}_{h,0.13})} \approx 0.18$, which means the mesh optimisation problem is more difficult. In Figure 5.26 we can see that both the cell size distribution error and the degree of anisotropy are high near the boundary, especially near the corner points of $\partial\Omega$. All of this comes as no surprise.

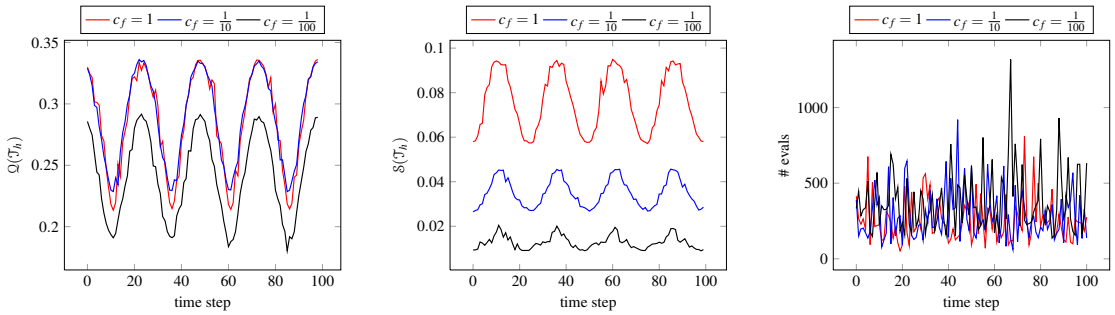


(i) $\mathcal{T}_{h,0}$, $\mathcal{S} \in [1e{-}5, 1e{-}3]$.

(ii) $\Phi^*(\mathcal{T}_{h,0.15})$, $\mathcal{S} \in [1e{-}5, 1e{-}4]$.

(iii) $\Phi^*(\mathcal{T}_{h,0.35})$, $\mathcal{S} \in [1e{-}5, 1e{-}4]$.

Figure 5.26: The domain with the rotating object at different time steps, $c_f = 0.01$, hypercube case.



(i) Shape quality indicator over time.

(ii) Size quality defect over time.

(iii) Needed number of functional evaluations per time step for various c_f.

Figure 5.27: Various quantities over time for the hypercube mesh.

More interesting is the comparison between the different values of c_f. The value c_f determines how deviating of the reference cell's edge lengths is penalised, which indirectly governs how anisotropy is penalised. However, since the boundary restricts the vertex movement, some degree of anisotropy will be required to reduce the cell size defect. In Figure 5.27 (i), we see that $\mathcal{Q}$ is nearly the same for $c_f = 1$ and $c_f = 0.1$, but notably lower for $c_f = 0.01$. In contrast to this, the cell size defect $\mathcal{S}$ decreases approximately by a factor of 2 for every reduction of c_f. The time averages in Table 5.12 confirm this.

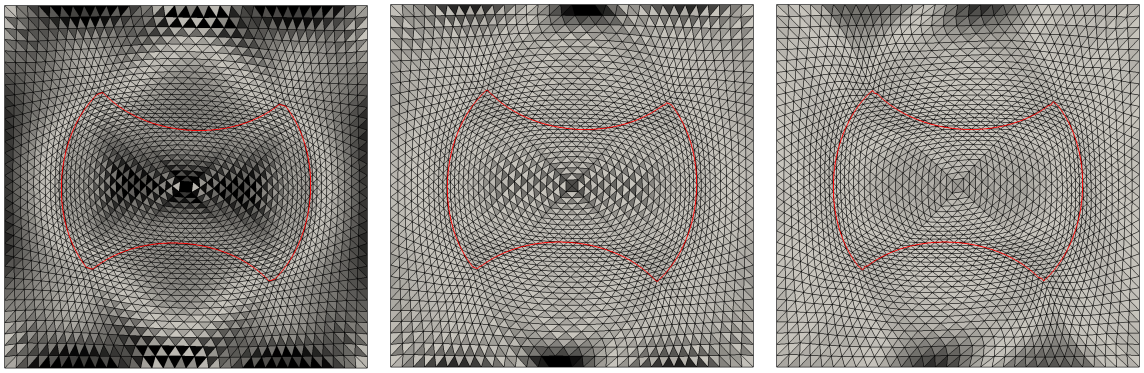
Another interesting observation is that the number of iterations per time step increases significantly with decreasing c_f, but the number of needed functional evaluations only to a lesser extent. More precisely, the average number of functional evaluations *per iteration* decreases from ≈ 6 to ≈ 3. This means that the linesearch needs less iterations, which is an indicator for the problem with higher c_f being more difficult. There is not much information to be gained from the number of evaluations per time step (see Figure 5.27 (iii)), as this number is strongly varying with time. There are accumulations of high evaluation counts near time steps with local maximal in $\mathcal{S}$, but they are not outstanding.

For the simplex case, in Figure 5.28 $\mathcal{S}(\Phi^*(\mathcal{T}_{h,0.75}))$ is shown for different c_f, with the same colour scaling in each picture. We see that the magnitude is quite different, although the distribution is similar. Clearly we see that for $c_f = 1$, the simplices tend to keep the equilateral shape of their reference cell more often than for the smaller values. Especially for $c_f = 0.01$ we see that

c_f	$\overline{\#\text{ its}}$	$\overline{\#\text{ evals}}$	$\overline{\mathfrak{Q}(\Phi^*)}$	$\overline{\alpha_{\min}(\Phi^*)}$	$\overline{\mathfrak{S}(\Phi^*)}$
0.01	123	379	$2.50e{-1}$	41.31	$1.56e{-2}$
0.1	79	285	$2.96e{-1}$	45.2	$3.76e{-2}$
1	42	254	$2.93e{-1}$	44.67	$7.79e{-2}$

Table 5.12: Quantities for the rotating shape for various c_f, hypercube mesh.

the near-symmetry to the x_2-axis we see for $c_f = 1$ is lost.



(i) $c_f = 1$ (ii) $c_f = \frac{1}{10}$ (iii) $c_f = \frac{1}{100}$

Figure 5.28: The domain with the rotating object at $t = 0.75$ for different values of c_f, $\mathfrak{S} \in [0, 5e{-5}]$.

c_f	$\overline{\#\text{ its}}$	$\overline{\#\text{ evals}}$	$\overline{\mathfrak{Q}(\Phi^*)}$	$\overline{\alpha_{\min}(\Phi^*)}$	$\overline{\mathfrak{S}(\Phi^*)}$
0.01	344	1064	$6.33e{-1}$	27.81	$1.78e{-2}$
0.1	204	1247	$6.64e{-1}$	29.8	$2.60e{-2}$
1	87	357	$6.61e{-1}$	29.44	$6.87e{-2}$

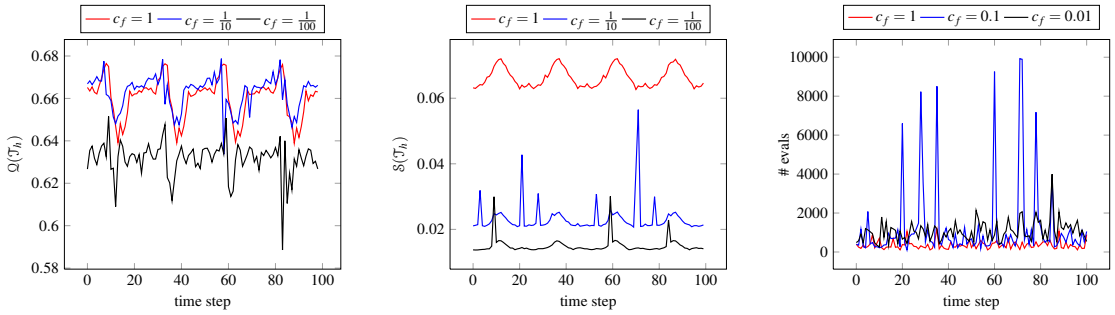
Table 5.13: Quantities for the rotating shape for various c_f, simplex mesh.

The evolution of $\mathfrak{Q}, \mathfrak{S}$ and the number of functional evaluations over time is similar to the hypercube case, with some noteworthy differences. $\mathfrak{Q}$ behaves similar when it comes to comparing it over the different values of c_f, but is not nearly monotone in between local extrema as in the hypercube case. The local extrema still occur in the same time steps, but the evolution in between is very different.

When comparing $\mathfrak{S}$, we again see the aforementioned reduction, although no longer a factor of nearly two between $c_f = 1$ and $c_f = 0.1$. However, there are some peaks in $\mathfrak{S}$, for example at $t = 0.2, 0.31, 0.56$ and others for $c_f = 0.1$. Note that those coincide with some of the peaks in the number of functional evaluations. In general, these are time steps where the solver either stopped at a "bad" local minimum, or (where the number of functional evaluations is very high) failed to converge with regard to $\varepsilon_r, \varepsilon_a$ or ε_s. In those cases, we see the solver performing hundreds of iterations with the linesearch returning very small steps (e.g. $O(1e-12)$) after performing the maximum number of iterations without satisfying the strong Wolfe conditions. As the lBFGS preconditioned steepest descent has no stagnation criterion like NLCG, it simply performs its

maximum number of iterations (which was 500) without converging.

This is the same situation as was described in Section 5.5.1 and appears hard to avoid in practise, as it is very much related to the initial guess the solver starts from. On the positive side we see that the time steps with "bad" solutions appear isolated in this context and have no impact on the subsequent time steps, although they do provide bad initial guesses for the solution process in the next time step.



(i) Shape quality indicator over time.

(ii) Size quality defect over time.

(iii) Needed number of functional evaluations per time step for various c_f.

Figure 5.29: Various quantities over time for the simplex mesh.

As we have seen in this section, a mesh quality functional of the type presented in Section 3.5 offers a great deal of flexibility when it comes to achieving different goals in r-adaptivity. The weighting of the parts using the terms $\|\nabla\Phi\|_F, \det(\nabla\Phi)$ and (in $3d$) $\mathrm{Cof}(\nabla\Phi)$ is crucial and also balances the cell quality against the cell size distribution. However, care must be taken, as the problem's difficulty also depends on the choice of the mentioned parameters, and might lead to prohibitive computation times and/or the occasional "bad" minimiser Φ^*.

5.6. Surface alignment

In this section, I want to present the results of various test cases concerning the alignment of the mesh with some surfaces. In all cases, the surface displayed in the images is not Γ itself, but the zero levelset of the finite element (meaning $\mathbb{P}_1(\mathcal{T}_h)$ or $\mathbb{Q}_1(\mathcal{T}_h)$) interpolant of the distance function

$$\tilde{\Gamma} = \{x \in \mathbb{R}^d : \Pi_h(\mathrm{dist}(x,\Gamma))(x) = 0\}. \tag{5.12}$$

5.6.1. Rotating ellipse

In this case we again consider $\Omega = \Omega_h = [0,1]^2 \subset \mathbb{R}^2$ and a surface Γ embedded in Ω by composite cubic Bézier curve given by

$$\gamma_i : [0,1] \to \mathbb{R}^2, \gamma_i(t) := (1-t)^3 P_{i1} + 3(1-t)^2 t P_{i2} + 3(1-t)t^2 P_{i3} + t^3 P_{\mathrm{mod}_4(i+1)1}$$

with

$$P_{11} = \left(\frac{7}{32}, \frac{1}{2}\right)^T, P_{12} = \left(\frac{7}{32}, \frac{5}{16}\right)^T, P_{13} = \left(\frac{5}{16}, \frac{1}{8}\right)^T,$$
$$P_{21} = \left(\frac{1}{2}, \frac{1}{8}\right)^T, P_{22} = \left(\frac{11}{16}, \frac{1}{8}\right)^T, P_{23} = \left(\frac{25}{32}, \frac{5}{16}\right)^T,$$
$$P_{31} = \left(\frac{25}{32}, \frac{1}{2}\right)^T, P_{32} = \left(\frac{25}{32}, \frac{11}{16}\right)^T, P_{33} = \left(\frac{11}{16}, \frac{7}{8}\right)^T,$$
$$P_{41} = \left(\frac{1}{2}, \frac{7}{8}\right)^T, P_{42} = \left(\frac{5}{16}, \frac{7}{8}\right)^T, P_{43} = \left(\frac{7}{32}, \frac{11}{16}\right)^T,$$

which is an approximation of the boundary ∂E of the ellipse

$$E := \left\{ x \in \mathbb{R}^2 : \left(\frac{x_1 - x_1^M}{\frac{1}{2}}\right)^2 + \left(\frac{x_1 - x_1^M}{\frac{3}{4}}\right)^2 \leq 1 \right\}, \quad x^M = \left(\frac{1}{2}, \frac{1}{2}\right)^T.$$

E is then rotated by 2π on the time interval $[0,1]$ (see Figure 5.30). In this example, the parameters are

$$\delta_r = 1\text{e}{-8}, p_d = 1, c_d = \sqrt{\delta_r^2 + 1} + \delta_r^2 + 1$$

and I will compare results for $c_f = 1$ and $c_f = 0.01$. As solvers, the quadratic penalty iteration with absolute tolerance $\varepsilon_a = 10^{-14}$ with NLCG with $\varepsilon_r = 1\text{e}{-8}$ and $\varepsilon_s = 2.204\text{e}{-16}$ were used. Also, the quadratic penalty iteration aborts if $\mu_k \geq 1\text{e}77$.

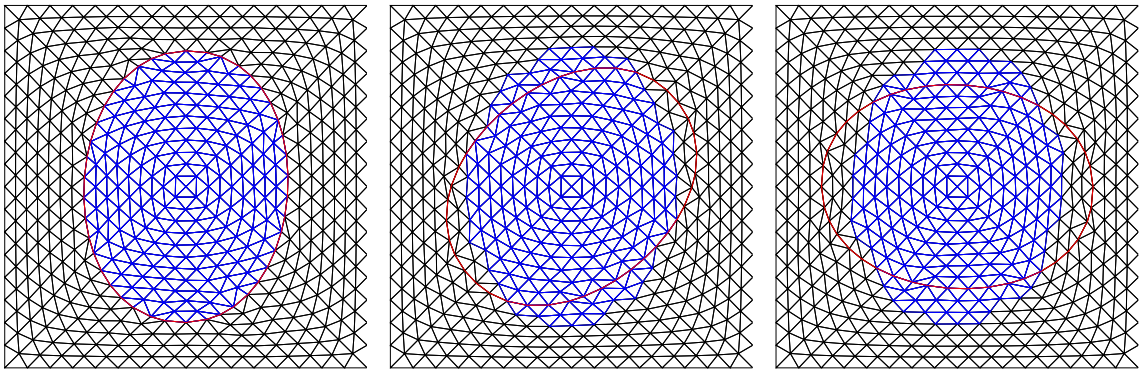
This simple case is already an example where an interface tracking method that fixes the vertices on Γ through a displacement boundary condition will invariably fail. If we are only interested in the part of the domain that is on the outside of Γ (meaning $\Omega \setminus E$), we could just mesh this domain of interest, rotate E and impose unilateral boundary condition of place on ∂E, together with e.g. a no displacement boundary condition on $\partial\Omega$. This would be a similar situation as Section 5.4.2.

However, in many situations we need a mesh on the whole of Ω, e.g. in two phase flow simulations like the rising droplet. This is a commonly used benchmark problem (see [HTK+09]), where a sharp interface representation of the free surface is advantageous due to the better order of convergence in space (see Section 3.2.3). A levelset-based surface alignment method for exactly this case was presented in [BW13], while another method based on front-tracking with occasional re-parametrisation with time-discontinuous ALE mappings can be found in [Bas16].

If Γ is interpreted as a phase boundary, the greatest benefit of aligning $\mathcal{T}_h$ with Γ without explicitly tracking the corresponding part of $\mathcal{T}_h$ is that cells are allowed to change phase. This allows for good mesh quality even in the face of large boundary deformations and changes of the phases' volume fraction. In Figure 5.30, cells of the set $\{K \in \Phi^* \mathcal{T}_h(0) : K \subset E(0)\}$ are coloured blue to visualise how they change from the interior to the exterior of E over time.

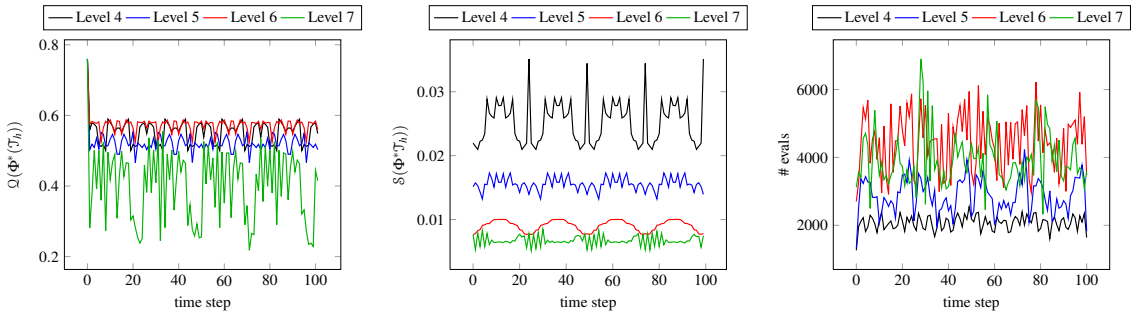
Let us first consider the case of simplex meshes with $c_f = 0.01$ to make the material less "stiff" with regard to changes in edge lengths and examine the same mesh quality heuristics as before. In addition to $\mathcal{Q}(\Phi^*(\mathcal{T}_h))$ and $\mathcal{S}(\Phi^*(\mathcal{T}_h))$ we now need to consider $\mathcal{H}(\Phi^*(\mathcal{T}_h))$, which is nothing else than the residual of the quadratic penalty iteration. Here, Φ^* is the solution of the quadratic penalty iteration. We can expect the surface alignment constraint to have a negative impact on $\mathcal{Q}$ and $\mathcal{S}$, as can be expected from the concept of the quadratic penalty iteration with the constraint becoming dominant.

Figure 5.31 shows that both $\mathcal{Q}$ and $\mathcal{S}$ stay with in very good bounds for the refinement levels $l = 4,5,6$, with the reduction of $\mathcal{S}$ with refinement we have already seen in the previous sections on r-adaptivity. On level seven, the shape quality heuristic $\mathcal{Q}$ is markedly lower and varies greatly from



(i) $\Phi^*(\mathcal{T}_h(0))$. (ii) $\Phi^*(\mathcal{T}_h(0.13))$. (iii) $\Phi^*(\mathcal{T}_h(0.25))$.

Figure 5.30: Surface aligned mesh at different time steps with the cells initially comprising the interior coloured blue.



(i) Shape quality indicator over time. (ii) Size quality defect over time. (iii) Total number of functional evaluations per time step.

Figure 5.31: Various quantities over time for the simplex mesh using $c_f = 0.01$.

time step to time step. Closer examination of the meshes $\Phi^*(\mathcal{T}_{h,7}(t))$ for the corresponding time steps shows situations like the one shown in Figure 5.32 (i). As the vertex marked blue was moved to $\Gamma(t)$, the surrounding cells were distorted. Recall that every mesh $\mathcal{T}_{h,l}(t)$ $\frac{\hat{h}_l}{2}$-resolves Γ, with $\hat{h}_l = \max_{e \in \mathcal{E}^0(\mathcal{T}_{h,l}(0))} \mathrm{vol}_1(e)$. As the surface alignment constraint $\mathcal{H}$ also depends on the distance (see (3.40), the absolute value for every edge decreases as the mesh better resolves Γ. This means that higher penalty parameters μ_k are required for the penalty term to have a significant influence on Q. Depending on how μ_k is increased, this may only happen when $\mathcal{H}$ already dominates $\mathcal{F}$ in the quadratic penalty function Q, resulting in badly shaped cells. Depending on where the alignment constraint is violated in iteration i, even vertices that will not be aligned with Γ after the final iteration will be moved towards it to reduce the constraint violation. In iteration $i+1$, if the $\mathcal{H}$ dominates Q, these vertices will not be adjusted again because $\mathcal{F}$ has almost no influence then. To retain a good shape quality, it seems advantageous if the alignment penalty term has its influence earlier in the penalty iteration when $\mathcal{F}$ and $\mathcal{H}$ still balance out.

Now we can examine the number of iterations needed for solving the penalised unconstrained minimisation problem in each iteration of the quadratic penalty iteration. These numbers are given in table Table 5.14. The first iteration requires the minimisation of a nearly unpenalised problem and requires many iterations as the initial guess is the aligned mesh of the previous time step, which might be of poor quality due to the aforementioned effects. This first iteration is crucial, though, as this is the point where cells may move freely so that the surface Γ might be resolved by different edges.

l	$\overline{i=1}$	$\overline{i=2}$	$\overline{i=3}$	$\overline{i=4}$	$\overline{i=5}$	$\overline{i=6}$	$\overline{i=7}$
4	664	2	23	375	464	9	4
5	1438	2	2	291	642	10	5
6	2409	3	2	593	728	7	0
7	2784	3	2	2	677	29	7

Table 5.14: Averages of the number of iterations for the penalised unconstrained minimisation problem in each penalty iteration i for $c_f = 0.01$.

We can see that this is (in average) the iteration requiring the most inner solver iterations. The outer iterations two and three require very few inner iterations, as the alignment penalty term is still the minor factor in Q. The main work appears to occur in the penalty iterations four and five, where usually $\mu_4 = O(10^5), \mu_5 = O(10^{11}), \mu_6 = O(10^{22})$ with the proposed strategy from (4.13). The penalty iterations six and seven then align the vertices to Γ with $\mathcal{F}$ having very little influence. Here we can also see that for level seven, very few inner iterations are performed in the outer iteration four, as the penalty parameter is still too small due to effects described above.

l	$\overline{i=1}$	$\overline{i=2}$	$\overline{i=3}$	$\overline{i=4}$	$\overline{i=5}$	$\overline{i=6}$	$\overline{i=7}$
4	201	3	3	73	172	9	5
5	354	6	3	23	190	19	13
6	575	3	3	23	221	9	4
7	750	3	2	2	209	30	11

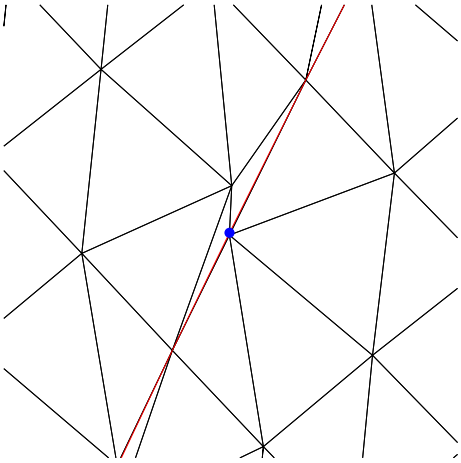
Table 5.15: Averages of the number of iterations for the penalised unconstrained minimisation problem in each penalty iteration i for $c_f = 1$.

To further look into this, we now set $c_f = 1$ to make the functional $\mathcal{F}$ penalise deviations in edge lengths even more. This should cause the term $\mathcal{H}$ to have its influence even later in the quadratic penalty iteration, which we can see in Table 5.14, as the number of inner iterations required in the fourth outer iteration is now very small. However, we note that for the outer iterations six and seven, the number of inner iterations required increased, most notably on levels 5 and 7.

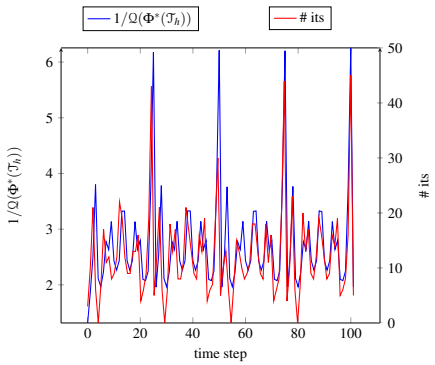
Looking at the plots of $\mathcal{Q}$ and $\mathcal{S}$ in Figure 5.33 (i) and Figure 5.33 (ii), we see about the same qualitative behaviour as in the case $c_f = 0.01$ for the levels four, six and seven, and some time steps with low $\mathcal{Q}$ on level five. With the idea that the low quality might be caused by vertices not being aligned with Γ until μ_k is very large, we can look at Figure 5.32 (ii), where both the number of inner iterations in the last penalty iteration and $1/\mathcal{Q}$ are plotted and see their relation. Unfortunately, this only helps to recognise the problem without giving any hints on how to solve it. All this indicates that there is still room for improvement when choosing the strategy for increasing μ_k.

Another important observation is that the total number of iterations and functional evaluations no longer follows the typical second order operator scheme as e.g. in Section 5.3, due to the unpredictable number of iterations the quadratic penalty iteration needs.

For the hypercube case, as discussed in Section 3.5.8, aligning edges to Γ results in a conflict if, for mesh topological reasons, three vertices of a cell K have to be aligned with a part of Γ with

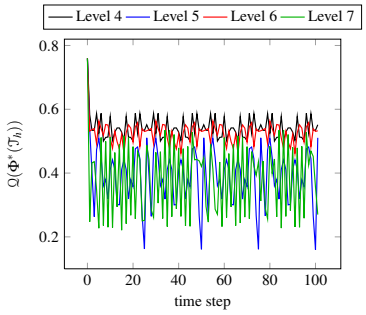


(i) $\Phi^*(\mathcal{T}_{h,7}(0.06))$ for $c_f = 0.01$.

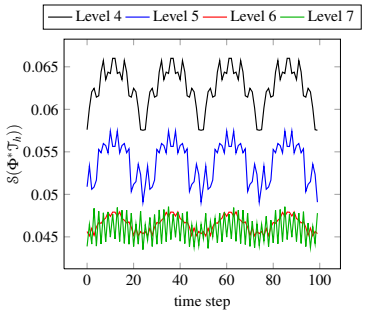


(ii) Shape quality indicator with number of inner iterations in penalty iteration seven over time for $c_f = 1$ and refinement level five.

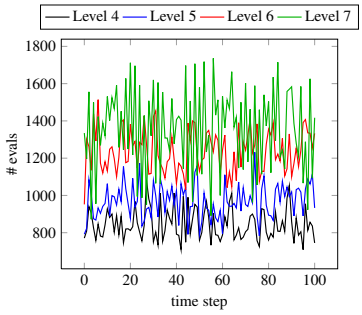
Figure 5.32: Distorted mesh due to interface alignment and the relation between $\mathcal{Q}$ and #its.



(i) Shape quality indicator over time.



(ii) Size quality defect over time.



(iii) Total number of functional evaluations per time step.

Figure 5.33: Various quantities over time for the simplex mesh, $c_f = 1$.

l	# evals	$\overline{\mathcal{Q}(\Phi^*)}$	$\overline{\alpha_w(\Phi^*)}$	$\overline{\mathcal{S}(\Phi^*)}$
4	2073	$5.54e{-1}$	25.04	$2.51e{-2}$
5	2885	$5.22e{-1}$	21.87	$1.56e{-2}$
6	4467	$5.73e{-1}$	24.63	$9.02e{-3}$
7	3927	$4.02e{-1}$	13.44	$6.72e{-3}$

(i) $c_f = 0.01$.

l	# evals	$\overline{\mathcal{Q}(\Phi^*)}$	$\overline{\alpha_w(\Phi^*)}$	$\overline{\mathcal{S}(\Phi^*)}$
4	845	$5.40e{-1}$	23.73	$6.26e{-2}$
5	971	$3.92e{-1}$	13.03	$5.38e{-2}$
6	1240	$5.25e{-1}$	22.79	$4.65e{-2}$
7	1356	$3.81e{-1}$	13.15	$4.59e{-2}$

(ii) $c_f = 1$.

Table 5.16: Quantities for the rotating ellipse for the simplex mesh on various levels of refinement.

low curvature, degenerating K to a triangle. Nevertheless, the stability property (3.32) of the local functional's integrand

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

(see Section 3.5.2) can ensure that we stay in the space of orientation preserving deformations.

This is controlled by the regularisation parameter δ_r in the functional

$$\mathcal{F}(\Phi) = \int_{\hat{\Omega}_h} c_f(\|\nabla\Phi\|_F^2 - d)^2 dx + (\det(\nabla\Phi))^{p_d}\, dx + \frac{c_d}{\left(\det(\nabla\Phi) + \sqrt{\delta_r^2 + (\det(\nabla\Phi))^2}\right)^{p_d}} dx,$$

but if $\delta_r > 0$, we can still cause the mesh to deteriorate by sending the penalty parameter $\mu_k \to \infty$. Increasing δ_r improves the degree of alignment in this case by allowing us to get "closer" to the set of orientation preserving deformations that allow $\det(\nabla\Phi) = 0$, but of course this would be detrimental to the overall mesh quality.

To demonstrate this, I chose a lower absolute tolerance $\varepsilon_a = 1\mathrm{e}{-8}$ for the quadratic penalty iteration, so it stops when the mesh is still not completely aligned with Γ and $c_f = 1$. See Figure 5.34 for examples of edges that cannot be aligned for the given reasons. However, it can be observed that, even if the alignment fails, $\Phi^*(\mathcal{T}_h)$ tends to be δ-aligned with Γ with a much smaller δ than $\delta = \frac{\hat{h}}{2}$.

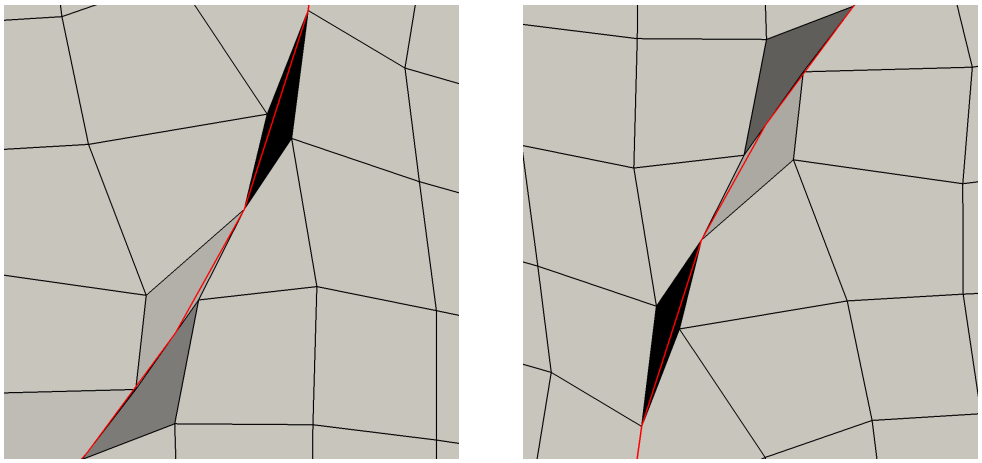


Figure 5.34: Hypercube cells remaining unaligned to Γ (red) in $\Phi^*(\mathcal{T}_{h,5}(0.22))$.

The conflict between the goals of alignment and good cell shapes is of course detrimental for the solver for the time-dependent problem, as the (partially) aligned mesh of t_n with the aforementioned nearly deteriorated cells is used as the initial guess for minimising the penalty function $Q(\Phi,\mu_k)$ for t_{n+1}, starting with $\mu_1 = 1$. One could start with a higher μ_1, but this might rule out cells changing phase (meaning that the set of aligned edges changes), sacrificing the most important advantage of the method. If the domain Ω_h is time independent, one might use $\mathcal{T}_h(0)$ as initial guess for all time steps, but this is not always the case.

The sum of all the mentioned problems means that in general, for hypercubes, we can only get good alignment at the cost of mesh quality, which is not very useful since we want to use the meshes $\Phi^*(\mathcal{T}_h)$ for the discretisation of other PDEs with finite elements. Moreover, the computational cost is even higher than one might have expected at first, as a solution from the previous time step is usually a poor initial guess for the only mildly penalised first quadratic penalty iteration. In view of this, only aligning vertices to Γ and then locally modifying the mesh (as discussed in Section 3.5.8 and in the references given there) might prove to be better to take advantage of this method. But since this is a step away from the variational approach with the desire to avoid combinatorial testing, modifications of the finite element spaces or the introduction of new DoF, it will not be covered in this work.

5.6.2. Merging circles, topology changes

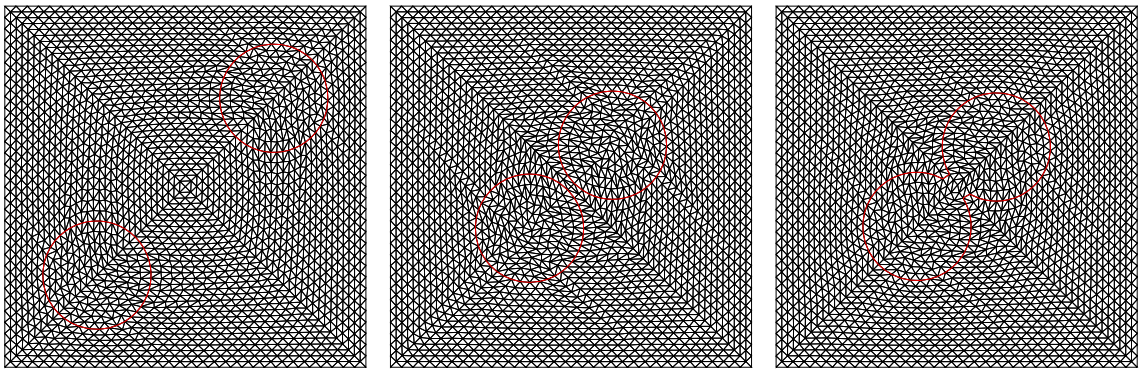
In this section, consider again $\Omega = \Omega_h = [0,1]^2 \subset \mathbb{R}^2$, this time with two moving surfaces

$$\Gamma_1 = \Gamma_1(t) = \partial B_{\frac{3}{20}}(x_1(t)), \Gamma_2 = \Gamma_2(t) = \partial B_{\frac{3}{20}}(x_2(t)),$$

on the time interval $[0,1]$, where

$$x_1(t) = \left(\frac{1}{4}, \frac{1}{4}\right)^T + \frac{t}{2}(1,1)^T, \quad x_2(t) = \left(\frac{3}{4}, \frac{3}{4}\right)^T - \frac{t}{2}(1,1)^T.$$

This means that the two surfaces move through each other, creating a topology change in the set $\Gamma(t) = \Gamma_1(t) \cup \Gamma_2(t)$. For the sake of brevity, I will only discuss the case of simplex meshes here, as all drawbacks of using hypercube meshes in conjunction with surface alignment given in Section 5.6.1 still apply.



(i) $\Phi^*(\mathcal{T}_h(0))$. (ii) $\Phi^*(\mathcal{T}_h(0.26))$. (iii) $\Phi^*(\mathcal{T}_h(0.27))$.

Figure 5.35: Surface aligned mesh at different time steps on refinement level five.

Here, we use

$$c_f = 0.01, \delta_r = 1\text{e}{-8}, p_d = 2, c_d = 2\delta_r^2 + \delta_r^2\sqrt{\delta_r^2+1} + 2 + 2\sqrt{\delta_r^2+1}$$

for the functional $\mathcal{F}$. As solvers, the quadratic penalty iteration with absolute tolerance $\varepsilon_a = 1\text{e}{-14}$ with NLCG with $\varepsilon_r = 1\text{e}{-8}$ and $\varepsilon_s = 2.204\text{e}{-16}$ were used.

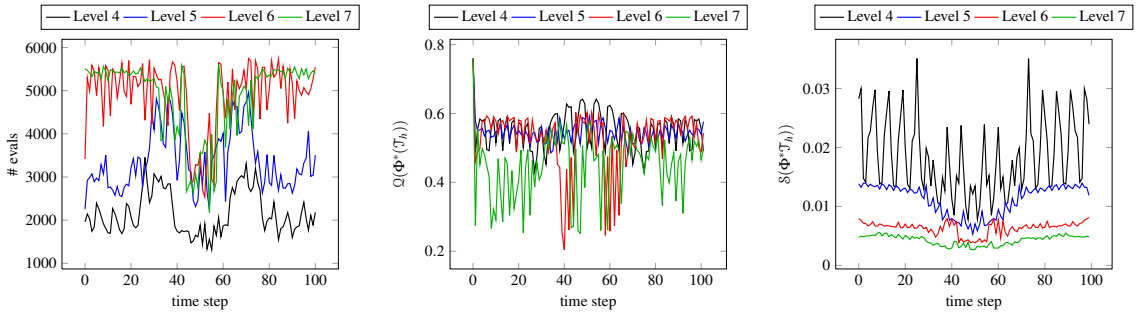
First, we again prescribe a uniform cell size distribution. The solution $\Phi^*(\mathcal{T}_h)$ is visualised in Figure 5.35 for several time steps. One can easily compute that $\Gamma_1(t)$ and $\Gamma_2(t)$ touch (or separate) if

$$\|x_1(t) - x_2(t)\|_2 = \frac{3}{10} \Leftrightarrow t_{1,2} = \frac{1}{2} \mp \sqrt{2}\frac{3}{10}, t_1 \approx 0.2878, t_2 \approx 0.7121.$$

Because of the mesh resolution, Γ will in general not be $\frac{h}{2}$ resolved in a time interval $(t_1 - \delta_1, t_1 + \delta_1) \ni t_{\text{con}}$, at which the contact usually happens for the discrete mesh and the representation of the distance function on that mesh. This can be seen in Figure 5.35, where $t_{\text{con}} = t_{27} = 0.27$ for refinement level five.

Apart from the surface Γ not being sufficiently resolved by the mesh, the topology change does not pose any additional problems to the method, although the outer quadratic penalty iteration causes a great variance in the number of inner iterations and functional evaluations. It should be noted that the inner solver usually stops because of the step length criterion.

The shape quality indicator $\mathcal{Q}$ stays in an acceptable range, even though some time steps with low values of $\mathcal{Q}$ can be observed for the higher refinement levels (e.g. for level six and $t = 0.4$).



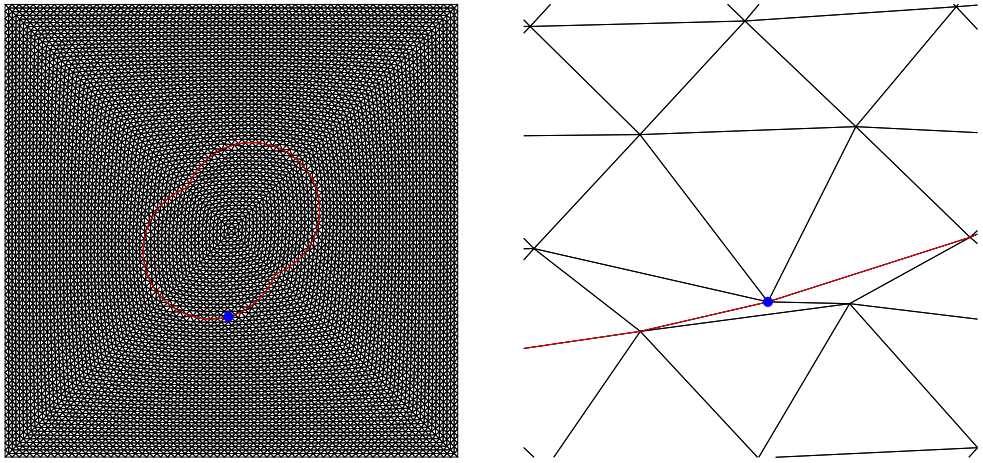
(i) Total number of functional evaluations per time step.

(ii) Shape quality indicator over time.

(iii) Size quality defect over time.

Figure 5.36: Various quantities over time for the simplex mesh with uniform cell size distribution.

Closer examination reveals that this is again due to vertices and edges getting aligned with Γ, which is depicted in Figure 5.37 for $t = 0.40$ on refinement level six. The vertex marked blue was aligned with Γ, unilaterally expanding or compressing adjacent cells, leading a badly shaped cell, where the lowest interior angle is $\alpha \approx 5.5°$. Because of this, the corresponding correction can only happen for a large penalty parameter μ_k, meaning $\nabla\mathcal{H}$ is the dominating part of $\mathcal{F}'$. The same effect can be observed for other time steps and occurs more often on refinement level seven.



(i) $\Phi^*(\mathcal{T}_h(32))$.

(ii) $\Phi^*(\mathcal{T}_h(0.32))$, focused on the cells with low $\mathcal{Q}$

Figure 5.37: Surface aligned mesh on refinement level six at $t = 0.40$ with focus on cells with bad shape quality indicator.

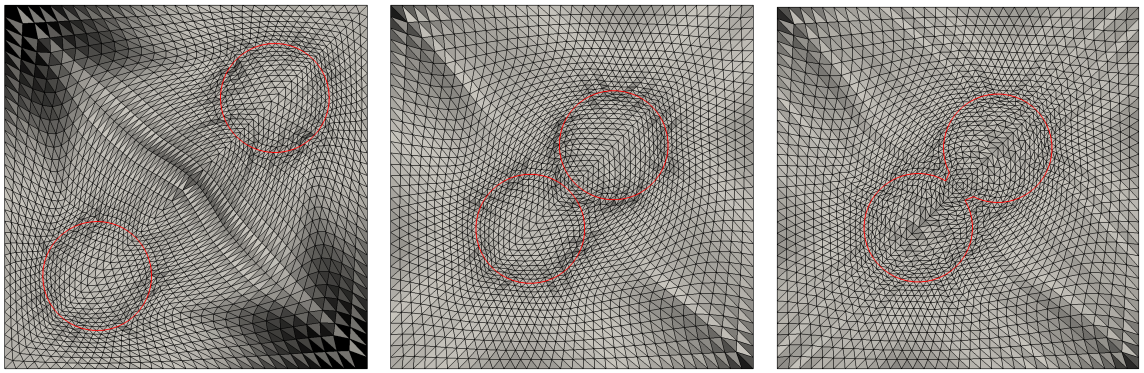
The size quality defect $\mathcal{S}$ also remains in a satisfactory range, and we see some reduction with further refinement as the cells near Γ (that get deformed in the alignment process) become smaller so that the size deviation becomes smaller, too.

Now we want to combine the alignment to Γ with r-adaptivity to see if the scheme remains stable and how the interaction between alignment and r-adaptivity influences the quality indicators $\mathcal{Q}$ and $\mathcal{S}$. The coefficients

$$c_1 = 0.08, \quad c_2 = 1$$

are used for the simple mesh concentration as defined in (5.10). This results in a ratio $\frac{\lambda_{\min}}{\lambda_{\max}} \approx \frac{1}{7}$. The domain with a visualisation of $\mathcal{S}$ can be found in Figure 5.38.

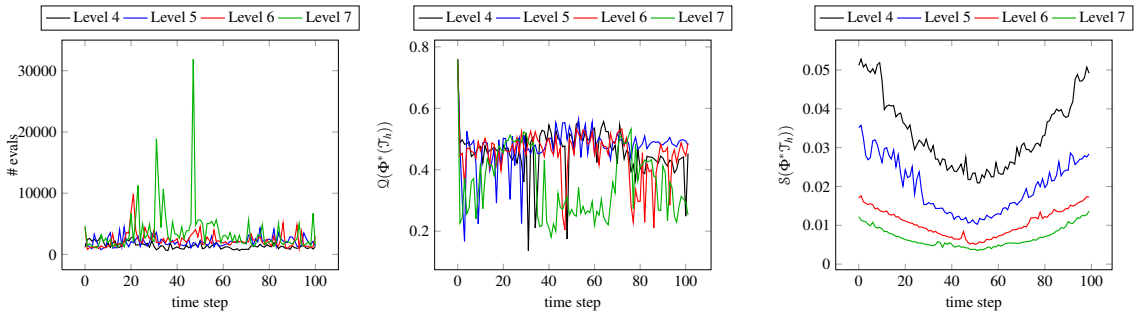
In the plot in Figure 5.39 we can see, that the number of functional evaluations is even more



(i) $\Phi^*(\mathcal{T}_h(0))$. (ii) $\Phi^*(\mathcal{T}_h(0.26))$. (iii) $\Phi^*(\mathcal{T}_h(0.27))$.

Figure 5.38: $\mathcal{S} \in [0, 5e{-}5]$ from white to black on the surface aligned mesh at different time steps on refinement level five with *r*-adaptivity.

unpredictable than in the case without surface alignment. As the local cell sizes influence the condition number of the problem to solve (see the discussion in Section 5.5.2), this was expected. The shape quality indicator $\mathcal{Q}$ stays in the same range with similar peaks as in the uniform case.



(i) Total number of functional evaluations per time step. (ii) Shape quality indicator over time. (iii) Size quality defect over time.

Figure 5.39: Various quantities over time for the simplex mesh with *r*-adaptivity.

A comparison of the number of needed functional evaluations, $\mathcal{Q}$ and $\mathcal{S}$ is given in Table 5.17. The table confirms the assessment that the combination of *r*-adaptivity and interface alignment makes the behaviour of the mentioned quantities unpredictable with regard to refinement.

	uniform			*r*-adaptivity		
l	# evals	$\overline{\mathcal{Q}(\Phi^*(\mathcal{T}_h))}$	$\overline{\mathcal{S}(\Phi^*(\mathcal{T}_h))}$	# evals	$\overline{\mathcal{Q}(\Phi^*(\mathcal{T}_h))}$	$\overline{\mathcal{S}(\Phi^*(\mathcal{T}_h))}$
4	2154	$5.52e{-}1$	$1.84e{-}2$	1418	$4.58e{-}1$	$3.38e{-}2$
5	3375	$5.46e{-}1$	$1.12e{-}2$	2061	$4.74e{-}1$	$2.00e{-}2$
6	4901	$5.26e{-}1$	$6.34e{-}3$	2358	$4.52e{-}1$	$1.07e{-}2$
7	4880	$4.51e{-}1$	$4.28e{-}3$	3649	$3.56e{-}1$	$6.89e{-}3$

Table 5.17: Quantities for the merging circles test, simplex meshes.

6

Preconditioning of the nonlinear system

In this chapter, I want to present a preconditioner for the class of nonlinear problems that is the focus of this work.

Recall that both Algorithm 4.1 and Algorithm 4.2 contained a descent direction $d^{(k+1)}$ which was written as

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

for the NLCG algorithm, where setting $\beta^{(k)} = 0$ recovers the NLSD method. The operator B^{-1} is the preconditioner and needs to be symmetric and positive definite to guarantee that $-B^{-1}\,\mathcal{F}'(\Phi^{(k)})$ is a descent direction. Choices for B were already discussed in Section 4.2 and included

i) $B = \mathcal{F}''$, which means using the Newton direction as a search direction,

ii) quasi Newton methods, which construct an approximation $B^{-1} \approx (\mathcal{F}'')^{-1}$.

Quasi Newton methods like lBFGS or DFP are algebraic in nature, as they use the discrete representations $\mathcal{F}'(\Phi^{(k)})$. I now want to introduce a simple preconditioner motivated by $\mathcal{F}$ on the continuous level.

6.1. Preconditioning with a second order operator

Recall the functional (3.23):

$$\mathcal{F}(\Phi) = \int_{\hat{\Omega}_h} c_f(\|\nabla\Phi\|_F^2 - d)^2 dx + (\det(\nabla\Phi))^{p_d}\,dx + \frac{c_d}{\left(\det(\nabla\Phi)) + \sqrt{\delta_r^2 + (\det(\nabla\Phi))^2}\right)^{p_d}} dx.$$

As we have seen in e.g. in Section 5.3, even the iteration numbers of the nonlinear solvers used exhibit the characteristic behaviour of a second order operator. Indeed, recall

$$\left(\|\nabla\Phi\|_F^2 - d)^2\right)' \eta = 4\left(\|\nabla\Phi\|_F^2 - d\right)\nabla\Phi : \nabla\eta$$

from Section 3.4.3. This indicates that it might be 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

$$a : \mathcal{D}_{\Phi_0} \times \mathcal{D}_0 \to \mathbb{R}, \quad a(\Phi, \eta) = \int_{\hat{\Omega}} D(\Phi) : D(\eta) dx, \tag{6.1}$$

where $\hat{\Omega}$ is some reference domain. The discrete form

$$a_h : \mathcal{D}_{h,\phi_0} \times \mathcal{D}_{h,0} \to \mathbb{R}, \quad a_h(\Phi_h, \eta_h) = \int_{\hat{\Omega}_h} D(\Phi_h) : D(\eta_h) dx, \tag{6.2}$$

can be used to define the operator

$$A_h : \mathcal{D}_{h,\phi_0}(\hat{\mathcal{T}}_h) \to \mathcal{D}_{h,0}(\hat{\mathcal{T}}_h)', (A_h \Phi_h, \eta_h) := a_h(\Phi_h, \eta_h),$$

which can be identified by a $\mathbb{R}^{n \times n}$ matrix. From now on, I will just use this discrete form and identify all quantities with their associated $\mathbb{R}^n$ coefficient vectors. This is the motivation for defining a class of preconditioners by

$$B^{-1} : \mathcal{D}_{h,0}(\hat{\mathcal{T}}_h)' \to \mathcal{D}_{h,\phi_0}(\hat{\mathcal{T}}_h), \qquad B^{-1} d_h := \tilde{A}_h^{-1} d_h \tag{6.3}$$

by 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 6.1.

i) 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.

ii) As it was mentioned in Remark 4.11, the solvers from Section 4.2 can also directly be formulated in Hilbert spaces, including Newton's method using the Hessian $\mathcal{F}''$. With this in mind, the preconditioner presented here is nothing but a very simple approximation of the Hessian.

iii) In the current form where

$$a_h(\Phi_h, \eta_h) = \int_{\hat{\Omega}_h} D(\Phi_h) : D(\eta_h) dx,$$

the preconditioner does not depend on the current iterate Φ_h^k and the current mesh $\Phi_h^{(k)}(\mathcal{T}_h)$ since the integral is taken on Ω_h. It is possible to obtain a variable metric method in the sense of Remark 4.4 by using the bilinear form

$$a_h^{(k)}(\Phi_h, \eta_h) = \int_{\Phi^{(k)}(\Omega_h)} D(\Phi_h) : D(\eta_h) dx,$$

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.

iv) For local functionals of the form (3.23) *(as repeated above), it is easy to bound*

$$\forall A \in \mathrm{SL}_d : c_1 \|A\|_F^2 \leq \mathcal{F}'(x,A).$$

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

$$\mathcal{F}'(x,A) \leq c\|A\|_F^2$$

is not possible in general. This means that the preconditioner will fail to give good results if the stability term of the functional $\mathcal{F}$ 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.

6.2. Expected limitations

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

Constant coefficients

The preconditioner is constructed from a second order operator with constant coefficients. This means that we cannot expect it to approximate the behaviour of $\mathcal{F}$ in the case of *r*-adaptivity, which is akin to having an inhomogeneous material response function, or highly varying optimal scales h. However, it is easy to incorporate non-constant coefficients $\beta : \Omega_h \to \mathbb{R}, \forall K \in \mathcal{T}_h : \beta_{|K} = \beta_k$ into the bilinear form $a_h(\cdot,\cdot)$, which should be chosen as some approximation $\beta_K = \left(\|\nabla R_K(id)\|_F^2 - d\right)$.

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 $\mathcal{F}$ depending on the other invariants of $\nabla\Phi$ become dominant. The same is true if a quadratic penalty iteration from Section 4.3 is used to enforce the surface alignment constraint (3.40). It might be possible to incorporate other terms into the bilinear form $a_h(\cdot,\cdot)$ to at least account for the other invariants of $\nabla(\Phi)$ account for this.

Boundary conditions

If unilateral boundary conditions are to be imposed on some parts of $\partial\Omega_h$, $a_h(\cdot,\cdot)$ has to be defined on the same spaces. But as mentioned in Section 5.4.2, imposing this boundary condition by the use of "simple" projection operators is not very stable with regard to preserving the orientation. More sophisticated projection operators are needed, which is beyond the scope of this work.

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 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.

7

Numerical results part II: Performance of the preconditioner

In this chapter, I will present results and discuss the behaviour of the class of preconditioners introduced in Chapter 6. Sometimes, I will just refer to a functional $\mathcal{F}$ or $\mathcal{G}$ without any more indices or details, which means the whole class of hyperelasticity-based functionals or the discrete $\mathbf{D}(u) : \mathbf{D}(v)$ functional (Section 5.4.1).

Recall that for an application of the preconditioner A_h^{-1}, we need to solve the system

$$A_h y = d, \tag{7.1}$$

which is in general done by an iterative method. One interesting question is now if it is sufficient to solve the system (7.1) only approximately, obtaining only an approximation $\tilde{A}_h^{-1}$ to the real preconditioner. In the case of a moving domain, I also want to investigate different reference domains for assembling the operator A_h on (see Section 5.4.1).

For a moving domain $\Omega = \Omega(t)$ and its polygonal approximation $\Omega_h = \Omega_h(t)$ with meshes $\mathcal{T}_h = \mathcal{T}_h(t)$, define the spaces

$$V_{h,l,k} = \left\{ v \in \mathcal{D}_h(\mathcal{T}_h(t_l)) : v_{|\partial\Omega_h(t_l)} = \hat{\Phi}_{t_k} \right\}, W_{h,l,k} = \left\{ w \in \mathcal{D}_h(\mathcal{T}_h(t_l)) : w_{|\partial\Omega_h(t_l)} = 0 \right\}$$

and the operator

$$A_{h,l,k} : V_{h,l,k} \to W'_{h,l,k}, \quad (A_h \Phi_h, \eta_h) := a_h(\Phi_h, \eta_h),$$

as in Section 5.4.1. We can then identify the operator with its matrix representation also denoted by $A_{h,l,k}$.

Remark 7.1. *I want to define some configurations for obtaining an approximation $\tilde{A}_h^{-1}$.*

i) $\left(\tilde{A}_h^{w,F}\right)^{-1}$*: Solve the linear system given by the operator $A_{h,0,k+1}$ approximately by applying exactly one multigrid V-cycle ("weak" preconditioner on a fixed reference domain).*

ii) $\left(\tilde{A}_h^{s,F}\right)^{-1}$*: Solve the linear system given by the operator $A_{h,0,k+1}$ approximately by with PCG-MGV ("strong" preconditioner on a fixed reference domain).*

iii) $\left(\tilde{A}_h^{w,M}\right)^{-1}$*: Solve the linear system given by the operator* $A_{h,k,k+1}$ *approximately by applying exactly one multigrid V-cycle ("weak" preconditioner on a moving reference domain).*

iv) $\left(\tilde{A}_h^{s,M}\right)^{-1}$*: Solve the linear system given by the operator* $A_{h,k,k+1}$ *approximately by with PCG-MGV (see Section A, "strong" preconditioner on a moving reference domain).*

If the problem is not time dependent, obviously the moving reference domain variants do not apply.

Remark 7.2. *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.

7.1. Refinement of a unit circle mesh

In this section, I want to revisit the example of Section 5.3. As mentioned in the preface of this chapter, I want to investigate the qualitative behaviour of the (preconditioned) nonlinear system arising from $\mathcal{F}$ as well as whether the use of a preconditioner allows the nonlinear solver to make further progress than in the unpreconditioned case. For comparison, I also used the NLSD-lBFGS solver from `ALGLIB` (see [Boc]). Because the academic edition only supports serial mode (meaning no MPI parallelism), all computations in this sections were done in serial, with a few exceptions that will be marked explicitly. The number of DoF increases from 256 (level three) to 16777216 (level eleven) for the simplex meshes (see Table 5.1) and from 320 (level three) to 20971520 (level eleven) for the hypercube meshes (see Table 5.4).

The configurations of the nonlinear solvers can be found in Table 7.1. The settings for the line search are the same as in Section 5.3, see Remark 5.3. Different preconditioners from Remark 7.1 will be used with the NLCG solver, their configurations can be found in Table 7.1.

Let us start by examining the results for simplex meshes. Again, levels for which a solver stagnated are marked with the letter s and if the solver stopped early because of the step length criterion or functional value criterion they are marked with an asterisk.

Simplex meshes

For all solvers for applying the preconditioners $(\tilde{A}_h^{\cdot,\cdot})^{-1}$, the solver configuration from Table 7.2 (i) was used.

Table 7.3 contains data from the same computation as Table 5.3 in Section 5.3, but is used to discuss solver aspects. We can see the number of iterations approximately doubling for every

Solver	NLSD-lBFGS
	$\varepsilon_r = 1e{-}8$
	$\varepsilon_f = 0$
	lBFGS_dim = 10
	max_iter = 10000

(i) Configuration 1.

Solver	NLCG
	$\varepsilon_r = 1e{-}8$
	$\varepsilon_s = 2.204e{-}8$
	$\varepsilon_f = 0$
	stag_iter = 10
	max_iter = 10000
Update	Dai-Yuan-Hestenes-Stiefel

(ii) Configuration 2.

Table 7.1: Different nonlinear solver configurations.

Solver	PCG-MGV $\varepsilon_r = 1e{-}8$ max_iter = 1000
MGV	coarsest level = 1
Smoother	Richardson-Jacobi
Smoother iterations	4 pre, 4 post
Coarse grid solver	PCG-Jacobi
Jacobi	$\omega = 0.7$
PCG-Jacobi	$\varepsilon_r = 1e-8$ max_iter = 1000

(i) Configuration 1.

Solver	PCG-MGV $\varepsilon_r = 1e{-}8$ max_iter = 1000
MGV	coarsest level = 1
Smoother	CG
Smoother iterations	4 pre, 4 post
Coarse grid solver	PCG-Jacobi
Jacobi	$\omega = 0.5$
PCG-Jacobi	$\varepsilon_r = 1e-8$ max_iter = 1000

(ii) Configuration 2.

Table 7.2: Different solver configurations for applying $(\tilde{A}_h^{\cdot,F})^{-1}$.

l	# its	# evals	$\mathcal{Q}(\hat{\Phi}(\mathcal{T}_h))$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\hat{\Phi}(\mathcal{T}_h))$	$\mathcal{Q}_a(\Phi^*)$	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	44	75	$7.13e{-}1$	$7.21e{-}1$	$7.68e{-}1$	$8.49e{-}1$	$8.71e{-}9$	$6.36e{-}3$
4	103	147	$5.46e{-}1$	$6.94e{-}1$	$7.48e{-}1$	$8.51e{-}1$	$9.42e{-}9$	$4.25e{-}2$
5	208	257	$4.02e{-}1$	$6.76e{-}1$	$7.45e{-}1$	$8.52e{-}1$	$9.02e{-}9$	$2.95e{-}1$
6	400	453	$2.90e{-}1$	$6.63e{-}1$	$7.49e{-}1$	$8.53e{-}1$	$9.76e{-}9$	$2.08e{+}0$
7	1081	1608	$2.07e{-}1$	$6.53e{-}1$	$7.53e{-}1$	$8.53e{-}1$	$9.52e{-}9$	$2.92e{+}1$
8	3908	9229	$1.47e{-}1$	$6.44e{-}1$	$7.56e{-}1$	$8.53e{-}1$	$9.86e{-}9$	$6.81e{+}2$
9s	1156	4602	$1.04e{-}1$	$4.63e{-}1$	$7.57e{-}1$	$7.77e{-}1$	$6.36e{-}7$	$1.38e{+}3$
10*	388	748	$7.37e{-}2$	$2.74e{-}1$	$7.59e{-}1$	$7.13e{-}1$	$6.66e{-}7$	$1.06e{+}3$
11s	359	797	$5.21e{-}2$	$3.40e{-}1$	$7.59e{-}1$	$7.43e{-}1$	$6.38e{-}7$	$3.97e{+}3$

Table 7.3: NLCG, simplex meshes.

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	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}_a(\Phi^*)$	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	40	72	$7.13e{-1}$	$7.21e{-1}$	$7.68e{-1}$	$8.49e{-1}$	$9.66e{-9}$	$6.23e{-3}$
4	94	127	$5.46e{-1}$	$6.94e{-1}$	$7.48e{-1}$	$8.51e{-1}$	$8.11e{-9}$	$3.86e{-2}$
5	184	259	$4.02e{-1}$	$6.76e{-1}$	$7.45e{-1}$	$8.52e{-1}$	$8.57e{-9}$	$3.11e{-1}$
6	377	719	$2.90e{-1}$	$6.63e{-1}$	$7.49e{-1}$	$8.53e{-1}$	$9.93e{-9}$	$3.40e{+0}$
7	791	1613	$2.07e{-1}$	$6.53e{-1}$	$7.53e{-1}$	$8.53e{-1}$	$1.00e{-8}$	$3.00e{+1}$
8	1915	4546	$1.47e{-1}$	$6.44e{-1}$	$7.56e{-1}$	$8.53e{-1}$	$9.94e{-9}$	$3.65e{+2}$
9s	389	1924	$1.04e{-1}$	$1.63e{-1}$	$7.57e{-1}$	$7.54e{-1}$	$2.85e{-5}$	$5.84e{+2}$
10*	507	3102	$7.37e{-2}$	$2.08e{-1}$	$7.59e{-1}$	$7.31e{-1}$	$9.78e{-7}$	$4.12e{+3}$
11*	759	1331	$5.21e{-2}$	$4.92e{-1}$	$7.59e{-1}$	$7.31e{-1}$	$3.20e{-8}$	$7.43e{+3}$

Table 7.4: NLSD-lBFGS, simplex meshes.

The iteration and functional evaluation numbers when using NLSD-lBFGS (see Table 7.4) 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 $\varepsilon_f = 0$, 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 ([NW06, Chapter 6.4]), the limited memory variant applied to this nonconvex functional actually behaves quite similar to unpreconditioned NLCG (Table 7.3) up to level seven. This indicates that using the Newton direction (4.3) (in the cases where it actually is a descent direction) e.g. by appropriately preconditioning NLSD or NLCG will most likely not give decisively different convergence rates.

If we apply the preconditioner $(A_h^{w,F})^{-1}$ (the results can be found in Table 7.5), the number of iterations only increases slightly with each level of refinement, up to level seven. Like in the unpreconditioned case (see Table 7.3), 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 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, $A_h^{w,F}$ has stopped being a good preconditioner, either because the approximation $(A_h^{w,F})^{-1}$ 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 $(A_h^{s,F})^{-1}$ gives much better results. Note that the same increase in functional evaluations per NLCG iteration as before occurs.

Applying the strong preconditioner $\tilde{A}_h^{s,F}$ yields iteration number that increase only slightly

l	# its	# evals	$\mathcal{Q}(\hat{\Phi}(\mathcal{T}_h))$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\hat{\Phi}(\mathcal{T}_h))$	$\mathcal{Q}_a(\Phi^*)$	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	26	71	7.13e−1	7.21e−1	7.68e−1	8.49e−1	5.69e−9	1.82e−2
4	33	61	5.46e−1	6.94e−1	7.48e−1	8.51e−1	5.72e−9	3.67e−2
5	41	72	4.02e−1	6.76e−1	7.45e−1	8.52e−1	8.02e−9	1.50e−1
6	47	76	2.90e−1	6.63e−1	7.49e−1	8.53e−1	9.97e−9	6.70e−1
7	48	123	2.07e−1	6.53e−1	7.53e−1	8.53e−1	6.96e−9	5.49e+0
8	255	2313	1.47e−1	6.45e−1	7.56e−1	8.53e−1	9.98e−9	2.29e+2
9*	333	4090	1.04e−1	5.98e−1	7.57e−1	8.52e−1	9.85e−8	2.36e+3
10*	503	6621	7.37e−2	6.43e−1	7.59e−1	8.48e−1	6.17e−8	8.55e+3
11s	125	2095	5.21e−2	1.42e−2	7.59e−1	8.00e−1	2.98e−1	1.02e+4

Table 7.5: NLCG-$\tilde{A}_h^{w,F}$, simplex meshes.

l	# its	# evals	$\mathcal{Q}(\hat{\Phi}(\mathcal{T}_h))$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\hat{\Phi}(\mathcal{T}_h))$	$\mathcal{Q}_a(\Phi^*)$	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	24	48	7.13e−1	7.21e−1	7.68e−1	8.49e−1	9.10e−9	2.38e−2
4	30	61	5.46e−1	6.94e−1	7.48e−1	8.51e−1	7.76e−9	7.87e−2
5	32	61	4.02e−1	6.76e−1	7.45e−1	8.52e−1	8.18e−9	3.45e−1
6	36	66	2.90e−1	6.63e−1	7.49e−1	8.53e−1	6.97e−9	1.41e+0
7	38	68	2.07e−1	6.53e−1	7.53e−1	8.53e−1	7.90e−9	1.35e+1
8	41	90	1.47e−1	6.45e−1	7.56e−1	8.53e−1	8.27e−9	4.99e+1
9	40	161	1.04e−1	6.30e−1	7.57e−1	8.53e−1	8.80e−9	2.73e+2
10	79	724	7.37e−2	6.17e−1	7.59e−1	8.48e−1	8.25e−9	4.93e+3
11	28	280	5.21e−2	5.24e−1	7.59e−1	8.41e−1	7.37e−9	2.45e+3

Table 7.6: NLCG-$\tilde{A}_h^{s,F}$, simplex meshes.

with increasing refinement level (see Table 7.6). 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 run time or 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 $(\tilde{A}_h^{\cdot,\cdot})^{-1}$, the solver configuration from Table 7.2 (ii) was used because the configuration from Table 7.2 (i) resulted in the multigrid V-cycle not reducing the norm of the defect of the system it was applied to. The same effect was observed in other cases and will be discussed in Section 7.2.

l	# its	# evals	$\mathfrak{Q}(\hat{\mathfrak{p}}(\mathcal{T}_h))$	$\mathfrak{Q}(\Phi^*)$	$\mathfrak{Q}_a(\hat{\mathfrak{p}}(\mathcal{T}_h))$	$\mathfrak{Q}_a(\Phi^*)$	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	95	145	$1.80e{-1}$	$4.22e{-1}$	$4.24e{-1}$	$5.70e{-1}$	$9.97e{-9}$	$2.78e{+0}$
4	190	239	$1.86e{-1}$	$4.41e{-1}$	$4.22e{-1}$	$5.97e{-1}$	$7.19e{-9}$	$1.81e{+1}$
5	422	482	$1.39e{-1}$	$4.52e{-1}$	$4.26e{-1}$	$6.12e{-1}$	$8.80e{-9}$	$1.48e{+2}$
6	717	781	$7.18e{-2}$	$4.57e{-1}$	$4.29e{-1}$	$6.20e{-1}$	$9.91e{-9}$	$9.86e{+2}$
7	1209	1324	$3.65e{-2}$	$4.60e{-1}$	$4.32e{-1}$	$6.24e{-1}$	$9.53e{-9}$	$6.52e{+3}$
8	1991	2109	$1.84e{-2}$	$4.61e{-1}$	$4.33e{-1}$	$6.26e{-1}$	$9.64e{-9}$	$4.02e{+4}$
9	3256	3481	$9.23e{-3}$	$4.65e{-1}$	$4.34e{-1}$	$6.36e{-1}$	$9.69e{-9}$	$2.55e{+5}$
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 7.7: NLCG, hypercube meshes.

Table 7.7 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, so no time is given. 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.

Using NLSD preconditioned with lBFGS (Table 7.8) 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 (see Section 4.2.1) do not come into play.

When using the "weaker" preconditioner $\tilde{A}_h^{w,F}$ (the results are in Table 7.9), 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 $\tilde{A}_h^{s,F}$ results in the solver being able to converge with regard to the relative residual norm for all levels of refinement (see Table 7.10). 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,

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}_a(\Phi^*)$	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	86	138	1.80e−1	4.22e−1	4.24e−1	5.70e−1	1.48e−8	2.43e+0
4	167	238	1.86e−1	4.41e−1	4.22e−1	5.97e−1	8.63e−9	1.66e+1
5	330	507	1.39e−1	4.52e−1	4.26e−1	6.12e−1	9.01e−9	1.43e+2
6	564	932	7.18e−2	4.57e−1	4.29e−1	6.20e−1	9.85e−9	1.06e+3
7	1337	3156	3.65e−2	4.60e−1	4.32e−1	6.24e−1	9.78e−9	1.55e+4
8*	385	884	1.84e−2	4.00e−2	4.33e−1	6.12e−1	2.84e−6	1.69e+4
9*	746	1225	9.23e−3	5.49e−2	4.34e−1	6.09e−1	3.00e−7	9.10e+4
10*	567	1924	4.62e−3	1.82e−1	4.35e−1	5.30e−1	1.38e−7	5.55e+5
11*	817	1457	2.31e−3	1.71e−1	4.35e−1	5.03e−1	5.59e−8	1.63e+6

Table 7.8: NLSD-lBFGS, hypercube meshes.

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}_a(\Phi^*)$	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	59	136	1.80e−1	4.22e−1	4.24e−1	5.70e−1	1.30e−8	2.47e+0
4	82	126	1.86e−1	4.41e−1	4.22e−1	5.97e−1	7.90e−9	9.22e+0
5	102	150	1.39e−1	4.52e−1	4.26e−1	6.12e−1	7.97e−9	4.39e+1
6	117	164	7.18e−2	4.57e−1	4.29e−1	6.20e−1	8.35e−9	1.86e+2
7	105	154	3.65e−2	4.60e−1	4.32e−1	6.24e−1	9.96e−9	7.38e+2
8	191	1450	1.84e−2	4.61e−1	4.33e−1	6.26e−1	9.32e−9	2.80e+4
9	766	10050	9.23e−3	4.62e−1	4.34e−1	6.27e−1	9.36e−9	7.08e+5
10s	95	1329	4.62e−3	1.05e−2	4.35e−1	4.67e−1	5.13e−2	3.88e+5
11s	48	853	2.31e−3	3.01e−3	4.35e−1	4.47e−1	4.78e−1	9.78e+5

Table 7.9: NLCG-$\tilde{A}_h^{w,F}$, hypercube meshes.

with only nine iterations for refinement level eleven. But the shape quality heuristic $\mathcal{Q}(\Phi^*(\mathcal{T}_h))$ shows 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 $\varepsilon_r = 1\text{e}{-12}$ and can be found in the last row of Table 7.10. 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. Because of the expected run time, this computation was done in parallel, so no timing information is provided..

For the example of the refinement of the unit circle mesh, the effect of the preconditioners from Chapter 6 is very strong, especially of the NLCG-$\tilde{A}_h^{s,F}$ 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

l	# its	# evals	$\mathcal{Q}(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}(\Phi^*)$	$\mathcal{Q}_a(\hat{\varphi}(\mathcal{T}_h))$	$\mathcal{Q}_a(\Phi^*)$	$\frac{\|\mathcal{F}'(\mathcal{T}_h)\|_2}{\|\mathcal{F}'(\Phi^*)\|_2}$	time
3	60	101	$1.80e{-1}$	$4.22e{-1}$	$4.24e{-1}$	$5.70e{-1}$	$8.70e{-9}$	$2.14e{+0}$
4	78	121	$1.86e{-1}$	$4.41e{-1}$	$4.22e{-1}$	$5.97e{-1}$	$9.72e{-9}$	$1.02e{+1}$
5	86	123	$1.39e{-1}$	$4.52e{-1}$	$4.26e{-1}$	$6.12e{-1}$	$9.95e{-9}$	$4.26e{+1}$
6	91	137	$7.18e{-2}$	$4.57e{-1}$	$4.29e{-1}$	$6.20e{-1}$	$8.02e{-9}$	$1.85e{+2}$
7	82	118	$3.65e{-2}$	$4.60e{-1}$	$4.32e{-1}$	$6.24e{-1}$	$8.51e{-9}$	$6.63e{+2}$
8	66	101	$1.84e{-2}$	$4.61e{-1}$	$4.33e{-1}$	$6.26e{-1}$	$8.84e{-9}$	$2.38e{+3}$
9	45	75	$9.23e{-3}$	$4.61e{-1}$	$4.34e{-1}$	$6.27e{-1}$	$9.89e{-9}$	$6.82e{+3}$
10	29	63	$4.62e{-3}$	$4.63e{-1}$	$4.35e{-1}$	$6.30e{-1}$	$9.77e{-9}$	$2.31e{+4}$
11	9	29	$2.31e{-3}$	$9.69e{-2}$	$4.35e{-1}$	$5.54e{-1}$	$6.71e{-9}$	$3.68e{+4}$
11	196	620	$2.31e{-3}$	$4.62e{-1}$	$4.35e{-1}$	$6.27e{-1}$	$9.71e{-13}$	–

Table 7.10: NLCG-$\tilde{A}_h^{s,F}$, hypercube meshes.

than for linear problems, and that the relative residual norm might not be a good indicator for the quality of the corresponding solution.

7.2. Moving nonconvex shape

In this section I want to discuss the use of the preconditioner from Chapter 6 in the example of the moving nonconvex shape from Section 5.4.1. This example is useful for illustrating some characteristics associated with the solvers and preconditioners on moving reference domains. Because of the number of time steps that need to be computed, all computations in this section were done in parallel with eight to 16 processes.

First, I want to discuss the minimisers

$$\Psi^*_{h,l,k} = \operatorname{argmin}_{u \in V_{h,l,k}} \mathcal{G}(l,k,u)$$

for two cases.

i) $l = 0$: Fixed reference domain. This type of functional will be denoted $\mathcal{G}_F^i$.

ii) $l = k-1$: Moving reference domain. This type of functional will be denoted $\mathcal{G}_M^i$.

Here, i is an index for the solver configuration used. Two different solver configurations will be used, which are given in Table 7.11. The only difference is the smoother in the multigrid V-cycle. The results of this part will also be relevant for preconditioning the minimisation process for the hyperelasticity-based functional.

For this (and later, for the hyperelasticity-based mesh quality functional $\mathcal{F}$), these are the questions I want to investigate in this section:

i) In the case of a fixed reference domain, the linear system for $\mathcal{G}_F$ is the same for all time steps, so the solver should require the same number of iterations in each time step. Is this still the case near the break down?

Solver	PCG-MGV $\varepsilon_r = 1e{-}8$ max_iter = 1000
MGV	coarsest level = 1
Smoother	Richardson-Jacobi
Smoother iterations	4 pre, 4 post
Coarse grid solver	PCG-Jacobi
Jacobi	$\omega = 0.5$
PCG-Jacobi	$\varepsilon_r = 1e-8$ max_iter = 1000

(i) Configuration 1.

Solver	PCG-MGV $\varepsilon_r = 1e{-}8$ max_iter = 1000
MGV	coarsest level = 1
Smoother	CG
Smoother iterations	4 pre, 4 post
Coarse grid solver	PCG-Jacobi
Jacobi	$\omega = 0.5$
PCG-Jacobi	$\varepsilon_r = 1e-8$ max_iter = 1000

(ii) Configuration 2.

Table 7.11: Different solver configuration for $\mathcal{G}_F, \mathcal{G}_M$.

ii) In the case of a moving reference domain, the mesh becomes nonuniform and distorted. How does this influence the solvers for $\mathcal{G}_M$?

iii) Even though the functional $\mathcal{G}_F$ cannot be used to compute meshes for the whole time interval, is it still feasible to formulate a preconditioner on the fixed reference domain?

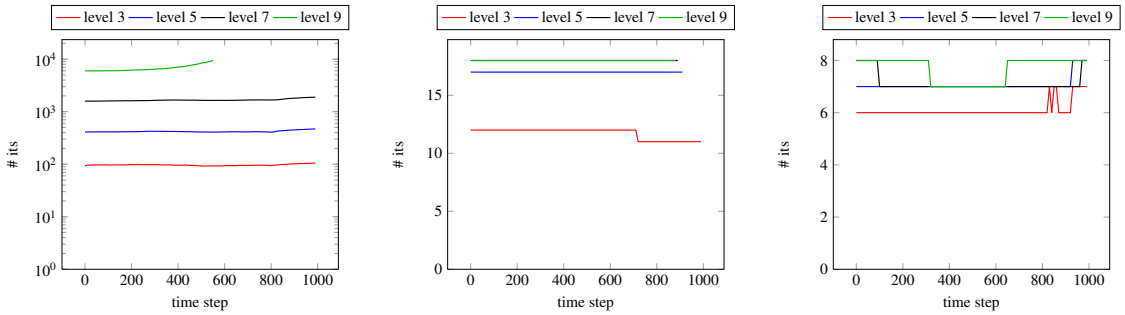
iv) The boundary deformation was chosen to greatly change the size of cells. This means the terms based on $\det(\nabla\Phi)$ in the hyperelasticity-based functional should become more important as the domain evolves. Which impact on the preconditioner does this have?

The functional $\mathcal{G}_F$ is of practical relevance because it is the computationally least expensive method presented in this work. Even though the results in Section 5.4.1 for this functional were not satisfactory, it is sufficient for cases with only small boundary deformations. In Table 7.12, we can see that the number of iterations for the case $\mathcal{G}_F^1$ is nearly mesh independent, as is expected from preconditioning CG with MGV for this type of problem. However, the early breakdown discussed in Section 5.4.1 (see also Figure 5.8 (i)) makes this method unsuitable for the example at hand.

For the case $\mathcal{G}_M^1$, we see that the number of iterations is no longer (nearly) mesh independent. Because the reference domain is moving, the linear system is reassembled in each time step on the current reference domain. However, due to the boundary movement, the mesh hierarchy is slightly disturbed (see Figure 7.2 (ii)), reducing the effectiveness of the MGV preconditioner in solver configuration 1. It turns out that the smoother in the V-cycle is not strong enough to handle this systematic error. With increasing refinement level, the time step from which on this systematic error has an influence occurs earlier (see Figure 7.2 (i)), resulting in greatly increased computation times. This can be avoided by using solver configuration 2, which uses CG as smoother in the multigrid V-cycle. Table 7.12 shows that the average number of iterations per time step is nearly independent of the refinement level. The average of the shape quality heuristic $\mathcal{Q}$ over time is also given as a reminder the variant $\mathcal{G}_M^2$ also gives the better meshes. The number of iterations for very time step can be found in Figure 7.1 for different solver configurations, including an unpreconditioned CG solver for $\mathcal{G}_M$ with $\varepsilon_r = 1e{-}8$ and max_iter $= 50000$. This is given to illustrate how effective the multigrid V-cycle is as a preconditioner, and because it can be seen that the number of iterations required increases with time as the domain deforms. For refinement level nine, the mesh deteriorates at time $t = 0.276$ (time step 552).

	$\mathcal{G}_F^1$		$\mathcal{G}_M^1$		$\mathcal{G}_M^2$	
l	# its	$\overline{\mathcal{Q}(\Psi^*(\mathcal{T}_h))}$	# its	$\overline{\mathcal{Q}(\Psi^*(\mathcal{T}_h))}$	# its	$\overline{\mathcal{Q}(\Psi^*(\mathcal{T}_h))}$
3	11	$5.00e{-1}$	14	$5.17e{-1}$	6	$5.17e{-1}$
4	14	$5.02e{-1}$	42	$5.12e{-1}$	7	$5.12e{-1}$
5	17	$5.01e{-1}$	175	$5.09e{-1}$	7	$5.09e{-1}$
6	18	$5.00e{-1}$	318	$5.07e{-1}$	7	$5.07e{-1}$
7	18	$4.99e{-1}$	424	$5.04e{-1}$	7	$5.06e{-1}$
8	18	$4.99e{-1}$	476	$4.99e{-1}$	8	$5.05e{-1}$
9	18	$4.98e{-1}$	482	$4.99e{-1}$	8	$5.05e{-1}$

Table 7.12: Average number of iterations and functional evaluations for different refinement levels.



(i) $\mathcal{G}_M$, number of unpreconditioned CG iterations in each time step.

(ii) $\mathcal{G}_F^1$, number of PCG-MGV iterations in each time step.

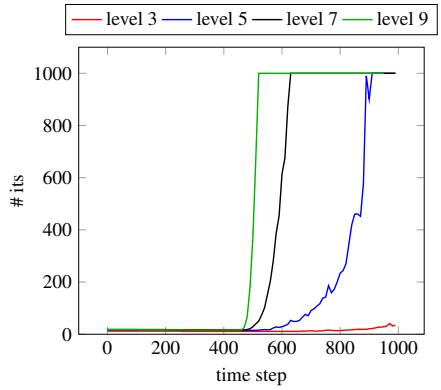
(iii) $\mathcal{G}_M^2$, number of PCG-MGV iterations in each time step.

Figure 7.1: $\mathcal{G}$, number of iterations in each time step for different solver configurations.

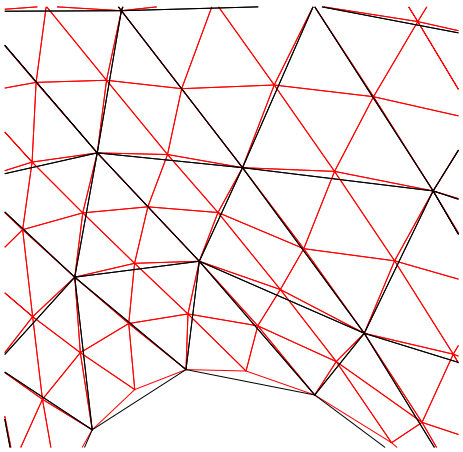
Now we want to examine the functional $\mathcal{F}_{h,k}$ from Section 5.4.1 and compare the results of different solver configurations as well. Here, $\mathcal{F}_F$ denotes any variant of the hyperelasticity-based functional using a preconditioner (if any) that uses a fixed reference domain. $\mathcal{F}_M$ is used analogously for moving reference domains. Note that the reference domain only matters for the preconditioner, so results for the unpreconditioned NLCG are given for comparison purposes where applicable.

The solver configurations are given in Table 7.13, where the outer solver is NLCG (with the same configuration as in Table 7.1 (ii) except for max_iter = 50000) and the preconditioner is always a variant of $\tilde{A}_h$. The preconditioners differ in their reference domain (fixed or moving) and the solver used. For the preconditioners $\tilde{A}_h^{w,M}$ and $\tilde{A}_h^{s,M}$ using the moving reference domain, the stronger multigrid V-cycles from Table 7.11 (ii) are used. Using the weaker multigrid V-cycle from Table 7.11 (i) results in a nonconverging preconditioner from a certain time step on, so that the resulting search direction is discarded and the NLCG degenerates into NLSD.

First, let us consider the unpreconditioned NLCG, for which the average number of iterations, functional evaluations and the mesh quality heuristic $\mathcal{Q}$ can be found in Table 7.14. We again see that the number of iterations more than doubles for each level of refinement, as does the number of functional evaluations, just as in the stationary case of the refinement of the unit circle



(i) $\mathcal{G}_M^1$, number of PCG-MGV iterations in each time step.



(ii) Nonhierarchical meshes on level three (black) and level four (red) for $t = 0.5$, $\mathcal{G}_M^2$.

Figure 7.2: Iteration numbers and nonhierarchical meshes associated with the use of a moving reference domain.

Solver	NLCG-$\tilde{A}_h^{w,F}$ $\varepsilon_r = 1e{-}8$ max_iter $= 1000$
$\tilde{A}_h^{w,F}$	Configuration 1 from Table 7.11 (i)

(i) Configuration 1.

Solver	NLCG-$\tilde{A}_h^{s,F}$ $\varepsilon_r = 1e{-}8$ max_iter $= 1000$
$\tilde{A}_h^{s,F}$	Configuration 1 from Table 7.11 (i)

(ii) Configuration 2.

Solver	NLCG-$\tilde{A}_h^{w,M}$ $\varepsilon_r = 1e{-}8$ max_iter $= 1000$
$\tilde{A}_h^{w,M}$	Configuration 2 from Table 7.11 (ii)

(iii) Configuration 3.

Solver	NLCG-$\tilde{A}_h^{s,M}$ $\varepsilon_r = 1e{-}8$ max_iter $= 1000$
$\tilde{A}_h^{s,M}$	Configuration 2 from Table 7.11 (ii)

(iv) Configuration 4.

Table 7.13: Different solver configuration for $\mathcal{F}_F, \mathcal{F}_M$.

mesh (Section 7.5). The important observation comes from Figure 7.3 (i): These numbers do not increase with time as the domain gets deformed. This indicates that the initial guess is not too far from a local minimiser of $\mathcal{F}$.

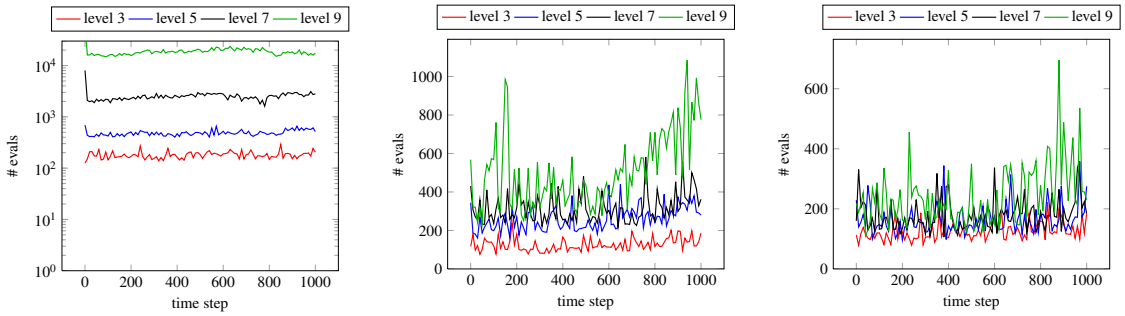
For the cases $\mathcal{F}_F$, the averages of the number of solver iterations, functional evaluations and the quality heuristic $\mathcal{Q}$ are given in Table 7.14. There we can see that both variants of the preconditioner drastically decrease the dependence of these numbers on the refinement level. For the configuration NLCG-$\tilde{A}_h^{w,F}$, we observe an increase by a factor of five over six levels of refinement, for the configuration NLCG-$\tilde{A}_h^{s,F}$ it is just a factor of two. The latter configuration is significantly more expensive than the previous, as every application of the preconditioner requires a significant number (10 to 30) multigrid V-cycles. Another thing to note is that the number of iterations increases with time. I attribute this to the deformed domain, where the terms based on $\det(\nabla\Phi)$ become more important (without being dominant), which the preconditioner does not capture.

These results indicate that even in the face of large deformations, the preconditioner from Chapter 6 is still a good choice, although its performance does indeed decrease as other parts of

	NLCG			NLCG-$\tilde{A}_h^{w,F}$			NLCG-$\tilde{A}_h^{s,F}$		
l	# its	# evals	$\overline{\mathcal{Q}(\Phi^*(\mathcal{T}_h))}$	# its	# evals	$\overline{\mathcal{Q}(\Phi^*(\mathcal{T}_h))}$	# its	# evals	$\overline{\mathcal{Q}(\Phi^*(\mathcal{T}_h))}$
3	98	185	$5.94e{-1}$	57	141	$5.94e{-1}$	42	127	$5.94e{-1}$
4	197	288	$5.90e{-1}$	86	187	$5.90e{-1}$	52	142	$5.90e{-1}$
5	385	482	$5.87e{-1}$	114	234	$5.87e{-1}$	59	155	$5.87e{-1}$
6	821	952	$5.83e{-1}$	140	277	$5.83e{-1}$	65	169	$5.83e{-1}$
7	1955	2252	$5.78e{-1}$	162	316	$5.78e{-1}$	71	177	$5.78e{-1}$
8	5105	5960	$5.71e{-1}$	184	361	$5.71e{-1}$	79	192	$5.71e{-1}$
9	13889	16427	$5.54e{-1}$	219	476	$5.54e{-1}$	88	235	$5.54e{-1}$

Table 7.14: $\mathcal{F}_F$, average number of iterations and functional evaluations on different refinement levels.

the functional become more relevant.



(i) Unpreconditioned NLCG. (ii) NLCG-$\tilde{A}_h^{w,F}$. (iii) NLCG-$\tilde{A}_h^{s,F}$.

Figure 7.3: $\mathcal{F}_F$, number of functional evaluations needed in every time step.

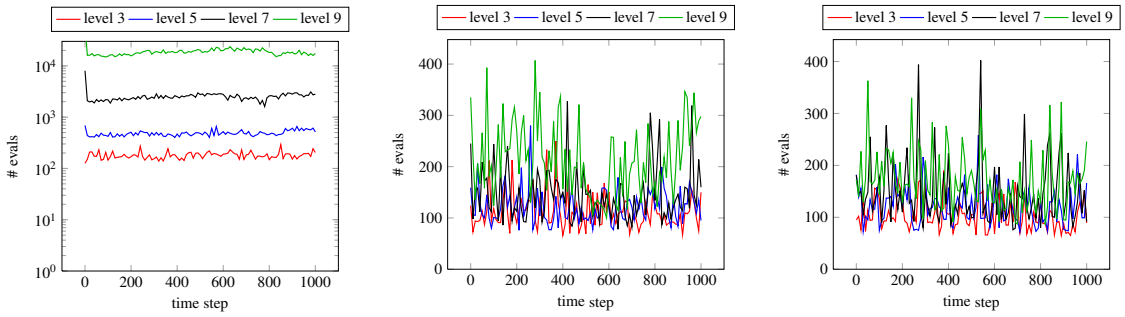
The slight degradation in the performance of the preconditioners $\tilde{A}_h^{\cdot,F}$ over time makes it worth exploring the use of a time dependent reference domain leading to the preconditioners $\tilde{A}_h^{\cdot,M}$. Here, only the multigrid V-cycle configuration from Table 7.11 (ii) will be discussed as the configuration from Table 7.11 (i) causes the solver for the preconditioner to fail to converge as mentioned above. This means that the system matrix for $\tilde{A}_h^{\cdot,M}$ needs to be reassembled in each time step.

In Table 7.15, a similar behaviour as in the case of $\mathcal{F}_F$ can be observed. The use of the preconditioners $\tilde{A}_h^{\cdot,M}$ greatly reduces the level dependence of the number of iterations and functional evaluations. For the preconditioner $\tilde{A}_h^{s,M}$, these numbers again increase approximately by a factor of two over six levels of refinement, and by a factor of 2.5 for the preconditioner $\tilde{A}_h^{w,M}$, which is significantly less than the factor of five for the preconditioner $\tilde{A}_h^{w,F}$ above.

In Figure 7.4 (ii) and Figure 7.4 (iii), the number of functional evaluations in every time step can be observed to not increase with time, which is an advantage over the preconditioners using a fixed reference domain. Results obtained by using a fixed reference domain (see Table 7.11 (i)) together with the stronger multigrid V-cycle configuration 2 from Table 7.11 (ii) show the same behaviour of increasing iteration number with time, which confirms that the above mentioned effect is really due to the moving reference domain and not the stronger linear solver for the application of the preconditioner.

	NLCG			NLCG-$\tilde{A}_h^{w,M}$			NLCG-$\tilde{A}_h^{s,M}$		
l	$\overline{\#\text{ its}}$	$\overline{\#\text{ evals}}$	$\overline{\mathcal{Q}(\Phi^*(\mathcal{T}_h))}$	$\overline{\#\text{ its}}$	$\overline{\#\text{ evals}}$	$\overline{\mathcal{Q}(\Phi^*(\mathcal{T}_h))}$	$\overline{\#\text{ its}}$	$\overline{\#\text{ evals}}$	$\overline{\mathcal{Q}(\Phi^*(\mathcal{T}_h))}$
3	98	185	$5.94e{-1}$	45	128	$5.94e{-1}$	26	105	$5.94e{-1}$
4	197	288	$5.90e{-1}$	60	150	$5.90e{-1}$	29	114	$5.90e{-1}$
5	385	482	$5.87e{-1}$	75	173	$5.87e{-1}$	33	123	$5.87e{-1}$
6	821	952	$5.83e{-1}$	95	204	$5.83e{-1}$	36	128	$5.83e{-1}$
7	1955	2252	$5.78e{-1}$	115	240	$5.78e{-1}$	41	141	$5.78e{-1}$
8	5105	5960	$5.71e{-1}$	136	280	$5.71e{-1}$	49	152	$5.71e{-1}$
9	13889	16427	$5.54e{-1}$	159	370	$5.54e{-1}$	59	175	$5.54e{-1}$

Table 7.15: $\mathcal{F}_M$, average number of iterations and functional evaluations on different refinement levels.



(i) Unpreconditioned NLCG. (ii) $\mathcal{F}_M$, NLCG-$\tilde{A}_h^{w,M}$. (iii) $\mathcal{F}_M$, NLCG-$\tilde{A}_h^{s,M}$.

Figure 7.4: $\mathcal{F}_M$, number of functional evaluations needed in every time step.

In this section we have seen that even for the class $\mathcal{G}$ of linear functionals, care has to be taken when choosing a solver. Using a moving reference domain is the default in many applications, and nonhierarchical meshes resulting from this can have a serious impact on the convergence of a geometric multigrid method. The same is then true for solving PDE with the finite element method on these meshes. A fixed reference domain still might be sufficient if the boundary deformations are small. The mesh on the reference domain being (nearly) uniform has a positive influence on the solver behaviour in the sense that the number of iterations stays the same for all time steps.

The same is also true for the preconditioners $\tilde{A}_h^{\cdot,\cdot}$. Because the preconditioner is used for solving a problem on the dual space, using a fixed reference domain has a less serious impact than using the function $\mathcal{G}_F$, but it still shows some degrading in iteration numbers as the domain moves and is deformed, because the preconditioner does not capture these effects. This can be remedied by using a moving reference domain for the preconditioner, keeping the deformations "small" in each time step.

Choosing the most efficient combination of preconditioner and solver tree appears to be highly problem dependent. As we have seen, the functionals $\mathcal{G}_F$ and $\mathcal{G}_M$ might even be sufficient if the mesh deformations are small and if we do not need to equidistribute the domain's volume over all cells. These functionals are computationally less demanding by several orders of magnitude. However, in the context of large deformations (especially compressing the mesh like in Section 3.4), using a functional of the class $\mathcal{F}$ might become unavoidable. In these cases, a moving ref-

erence domain for the preconditioner will be the better choice, as the assembly of the system matrix for the preconditioner is about as expensive as one evaluation of $\mathcal{F}$ and $\mathcal{F}'$. The results in this section suggest that it might be sufficient to solve the linear system for the application of the preconditioner with a high tolerance, or even perform a fixed number of iterations.

8

Conclusion

In this work, various aspects of PDE-based mesh optimisation were treated. Assume that we have a PDE we want to discretise with the finite element method and solve the discrete problem numerically. In many interpolation error estimates from Chapter 2 (see also [CR72]), the determinant of the local cell reference mapping's gradient is a relevant quantity, so e.g. when moving meshes are used, we first need to ensure that the cell reference mapping belongs to the space of orientation preserving mappings (Definition 3.1). Furthermore, a mesh of good quality might be necessary for discrete maximum principles. For this, mesh optimisation is an important tool.

Both PDE and non-PDE-based mesh optimisation have their own distinct set of advantages and disadvantages, where I chose the former because there are theoretical results that guarantee the existence of minimisers and to take advantage of existing tools for the numerical solution of PDEs with the finite element method.

In Chapter 3, the somewhat diffuse term mesh quality was discussed at length, with various examples illustrating different notions of mesh quality. The most important part is the orientation preserving property of a mesh deformation, which is fundamental to avoid self intersections and was directly incorporated into the definitions of the spaces of admissible deformations (Definition 3.1) and variations (Definition 3.2). The three important aspects of computing extension operators for boundary movement, *r*-adaptivity and alignment of meshes to (implicit) surfaces were motivated and discussed in Section 3.2. Especially aligning the mesh to interior surfaces (e.g. to resolve a phase boundary in two phase flow) can result in much better approximation properties for the finite element spaces discretising the original PDE (see [LMWZ10]).

Examples for mesh quality in $1d$ and $2d$ were given in Section 3.3, including examples where only the shape or the size of cells was used to define the quality of a mesh. In Section 3.4, three different PDE based methods were presented to compute an extension the movement of a boundary part of the underlying mesh. The two quadratic functionals (leading to linear problems that need to be solved) did not result in a stable mesh deformation method if a fixed reference domain was used. With a moving reference domain, the solutions stayed in the space of orientation preserving mappings (which cannot be enforced directly for quadratic functionals), but no useful lower bound for the spatial angles could be observed. The third functional was based on a stored energy functional for hyperelastic materials, directly incorporating the orientation preserving property and gave quite good results.

This was the motivation for revisiting an idea from [Rum96] to model a mesh as a hyperelastic

material, where a local optimal reference cell defines the material response function (Section 3.5). For a mapping Φ, the quantities $\|\nabla\Phi\|_F^2$, $\mathrm{Cof}(\nabla\Phi)$ and $\det(\nabla\Phi)$ describe the change in the 1, 2 and 3-dimensional volume of entities of the corresponding dimension, so it appears quite natural to use the quantities for the definition of mesh quality and mesh quality functionals. Many mesh quality functionals (see e.g. [HR11], [FK02]) can be traced back to explicitly or implicitly being defined in terms of these quantities. This allows for a fine-grained control of various quantities related to the shape, size and size distribution of cells (Section 3.5.3, Section 3.5.4 and Section 3.7) and makes the whole method robust enough that a condition for aligning the mesh to (implicit) surfaces can be incorporated directly into the mesh quality functional (Section 3.5.8, see also [BW13]). Unfortunately, the mesh topology when using hypercubes does not allow a mesh to be aligned to an arbitrary surface, as discussed in Section 3.5.8, which might be solved by special local refinement or splitting said cells into simplices. As this (locally) modifies the finite element spaces, this was not discussed further. Because of the relation to hyperelastic materials and polyconvex stored-energy functions, theoretical results from [Bal76] (see also [Cia88]) establish the existence and nonuniqueness of minimisers for this class of functionals (Section 3.5.6). The nonuniqueness of the solutions to elasticity problems is well-known ([Gur78], [Nol78]). Moreover, the notion that volumes can only be annihilated by infinite force which is critical for the orientation preserving property of deformations already rules out convex functionals (see Section 3.5.5 and [Cia88, Theorem 4.8-1], so this property cannot be directly incorporated in quadratic functionals as mentioned above.

In Chapter 4, some numerical methods for solving the discrete minimisation problems arising from the class of functionals from Section 3.5 were presented. As the theory from Section 3.5.6 is quite general, it covers the discrete case of solving the minimisation problem on finite element spaces (Section 4.1) as well. For the minimisation of the discrete mesh quality functional, methods from the field of unconstrained, nonconvex optimisation were introduced (Section 4.2), with the main focus on line search based methods. As the functionals are nonconvex and the initial guess for the iterative solver might be far away from a local minimum, Newton's method was not used (Remark 4.3, see also the examples in Section 4.4), but Quasi-Newton methods derived from the Broyden class were considered as preconditioners for line search methods. However, the focus here and in the subsequent Chapter 6 is on other preconditioners that are motivated by the continuous version of the PDE defining the mesh quality functional, rather than being more algebraic in nature like the Broyden class. The surface alignment condition from Section 3.5.8 represents a set of constraints, but no form of constraint qualification can be applied to it (Section 4.3), so most sophisticated method for the solution of the constrained minimisation problem seems to be a quadratic penalty iteration.

The numerical results with regard to the hyperelasticity-based mesh quality functionals were presented in Chapter 5. Here, the focus was on the results of the method itself, with no discussion of solver aspects, which are treated in Chapter 7. Quality heuristics for cell shapes and the cell size distribution were introduced and discussed in Section 5.2. Where possible (meaning for the computation of extension operators for boundary deformations in the sense of Section 3.2.1), the results were compared with the results obtained by using a quadratic mesh quality functional based on the $\mathbf{D}(u) : \mathbf{D}(v)$ bilinear form (see Section 3.4.2), e.g. in Section 5.3 and Section 5.4. This quadratic functional gave significantly better results than simpler, (graph-) Laplacian-based quadratic functionals (Section 3.4) and still results in a linear, symmetric, positive definite system matrix for which efficient solvers are available. Other functionals still giving rise to *linear* systems of equations (like functionals based on the biharmonic equation) were not discussed, as very often they introduce new mathematical and numerical difficulties that increase the numerical effort, which means losing one decisive advantage of using linear systems. It should be noted that this type of functional needs a reference domain, which can be a disadvantage.

In Section 5.3, the refinement of a crude approximation of the unit circle was used to study the behaviour of the quadratic and hyperelasticity-based functional with regard to recovering from a bad initial mesh. The quadratic mesh quality functional failed to do so, even though a uniform reference domain was available, as the boundary deformation was very strong. The hyperelasticity-based functional gave much better results up to a certain level of refinement, but the minimisation algorithm failed to converge with regard to the relative gradient norm criterion. Even in these cases, the quality of the resulting meshes was superior to the ones resulting from the quadratic functional. One important observation was that the iteration numbers in the nonlinear case exhibited a behaviour typical for linear systems arising from the discretisation of second order systems. This serves as a motivation for the class of preconditioners introduced in Chapter 6.

Two more cases of computing extension operators were discussed in Section 5.4.1 and Section 5.4.2. In the case of the moving nonconvex shape (Section 5.4.1), the quadratic mesh quality functional gave satisfactory results when a moving reference domain was used and failed to keep the mesh from deteriorating in the case of a fixed reference domain. The hyperelasticity-based functional gave better results still, but at higher numerical effort. For the geometry inspired by a micro gear pump in Section 5.4.2, the quadratic mesh quality functional could not be used as this geometry requires the use of unilateral boundary conditions of place (Definition 3.1), which are not easily enforced by linear projection operators. However, the results of the hyperelasticity-based functional were very good within the expected frame.

Various aspects of *r*-adaptivity were discussed in Section 5.5. In Section 5.5.1, the mesh concentration function was based on the distance to a moving circle and different choices of this concentration function were presented. While the results for simplex meshes showed no surprises, the case of hypercubes exposed some practical problems related to the convergence of the minimisation algorithm. It turned out that, depending on the initial guess, line search based solvers may fail to converge to useful local minima of the mesh quality functional, which is connected to the conditioning of the minimisation problem. Especially when *r*-adaptivity is used, this conditioning gets worse as the local optimal scales might differ greatly. The same could be observed in Section 5.5.2, where the the distance from a nonconvex shape like in Section 5.4.1 was used in the mesh concentration function and the scaling of the different terms in the mesh quality functional was examined. Even when using *r*-adaptivity, the hyperelasticity-based functional still allows some fine grained control over the weighting of cell shape and size distribution. However, strongly varying cell sizes can lead to the solver stopping at local minima resulting in poor meshes, which again indicates that care has to be taken when setting the goals for *r*-adaptivity.

The numerical results with regard to surface alignment were presented in Section 5.6. In Section 5.6.1, a bounding box mesh was aligned with a rotating ellipse and the behaviour of the quadratic penalty iteration from Section 4.3 is examined. It turns out the choice of the sequence of penalty parameters is crucial for the shape quality of the resulting surface aligned mesh. If the penalty parameter is increased too slow, many unconstrained minimisation problems need to be solved, but if it is increased too quickly, the solver may fail because the iterate from the last penalty iteration is a poor starting point for the current iteration. Moreover, cell shape and surface alignment are essentially conflicting goals. The alignment penalty term becomes dominant at some point of the iteration, so if a large deformation is needed to satisfy the constraint, cells of poor quality will occur. Perhaps the greatest advantage of using this method for aligning the mesh to surfaces is that the surface is not tracked explicitly. Instead, geometric entities (vertices, edges, faces) of an arbitrary mesh are aligned with the surface, which allows the set of aligned entities to vary (e.g. in each time step of an instationary process) and also allows for topology changes. This was demonstrated in Section 5.6.2, where the alignment was also combined with *r*-adaptivity. The topology changes are represented well but still demonstrate that the mesh must accurately resolve the surface.

Motivated by the second order operator behaviour of the iteration number in Section 5.3, a preconditioner for the solution of the nonlinear minimisation problem is introduced in Chapter 6. The system given by the matrix arising from a finite element discretisation of the operator defined by the discrete $\mathbf{D}(u) : \mathbf{D}(v)$ bilinear form is solved as a preconditioner, as the Frobenius norm term of the hyperelasticity-based mesh quality functional is very similar. This approach will only give good results as long as the terms based on $\|\operatorname{Cof}(\nabla\Phi)\|_F$ and $\det(\nabla\Phi)$ do not become dominant and needs further modification if the optimal scale distribution is nonuniform (e.g. if *r*-adaptivity is used).

The performance of the preconditioner from Chapter 6 is studied in Chapter 7, including the effect of approximately solving the preconditioner system with a low tolerance, or only applying one V-cycle of geometric multigrid and using this as an approximate inverse. For the refinement of a unit circle mesh (see Section 5.3), the iteration numbers only depend very slightly on the level of refinement for the strong preconditioner, but are notably higher if the weaker preconditioner is used. Also, the case of hypercubes appears to be more difficult. In Section 7.2, the moving nonconvex shape from Section 5.4.1 was used to examine the influence of a strongly distorting mesh on the preconditioner. It turned out that the choice of the reference domain for the quadratic functional used for the preconditioner is important in this case. If a fixed reference domain is chosen, the iteration numbers increase with time, as the preconditioner fails to represent the current state. If a moving reference domain is chosen, the PCG-MGV solver for the preconditioner system may stagnate or even diverge if the multigrid V-cycle is not "strong" enough. If the solver for the preconditioner is suitably chosen, we again obtain iteration number that only increase slightly with every level of refinement.

The present work shows that approaching mesh optimisation from the field of mathematical elasticity is very powerful, as it offers fine grained control over the notion of mesh quality, which might be different for different physical settings. It also allows to incorporate *r*-adaptivity and alignment of the mesh to (implicit) surfaces, which are very important tools for representing geometric or physical features in the discretisation of a PDE. But the computational effort is very high compared to other PDE-based mesh optimisation techniques, which might be sufficient in many situations (see Section 5.4.1 and Section 7.2), but are limited with regard to the boundary conditions that can be enforced (Section 5.4.2) and may fail altogether due to large boundary deformations (Section 3.4). Like many mesh optimisation techniques not based on PDEs, they can be improved to incorporate *r*-adaptivity, but this is not a strong direct control like prescribing an optimal cell size distribution as in the methods presented (Section 3.5.7). Moreover, since they cannot enforce the orientation preserving property directly (see Section 3.5.5), stability in the context of large deformations cannot be guaranteed. This means that in situations requiring adapted or aligned meshes, or with large deformations of the computational domain, a hyperelasticity-based mesh quality functional can be used to obtain the meshes to solve PDEs that cannot be handled by simpler methods, or dramatically increase the accuracy of the solutions (see [LMWZ10], [BP13]). The high computational effort can be remedied in some situations by using a preconditioner as introduced in Chapter 6.

Future work

It is unknown if the material response function for the family of mesh quality functionals based on the shape of reference cells (Section 3.5.3) converges to some well-defined material behaviour. In this case, there would be a continuous problem, of which the mesh optimisation is a discretisation. In this case, formulating the minimisation of the continuous functional directly in Hilbert spaces (see Remark 4.11) could be beneficial.

On the analytical side, no condition number estimates for the preconditioner from Chapter 6

were provided. Even in the case of affine transformations, this would require estimates of the form

$$\mathcal{F}'(x,A) \leq c\|A\|_F^2,$$

which seems impossible when taking the regularity property

$$\lim_{\det(A)\to 0} \mathcal{F}(x,A) = \infty$$

into account (see Remark 6.1). As the application of a preconditioner needs to have a low computational effort, but incorporating terms based on $\det(A)$ would make even the preconditioner nonlinear. As linearisations cannot capture this feature, this limitation seems hard to overcome. As mentioned in Remark 6.1, the preconditioner is just a simple approximation of the true Hessian $\mathcal{F}''$, so other approximations are possible and might yield even better properties.

In this work, hyperelasticity-based mesh optimisation was only applied in two dimensional settings. The theory from Section 3.5.6 is $3d$ and the method generalises trivially to $3d$, but some problems should be expected to arise in the implementation or the choice of the local stored energy functional. Examples of this method in $3d$ can e.g. be found in [Rum96] and [Hol15].

The minimisation process needs to start from an admissible state, which might not be available (e.g. the starting mesh) already contains intersecting cells). From this state, a mesh without intersecting cells can be generated, which is sometimes called *mesh untangling* in the literature ([HR11], [FK02]). With the hyperelasticity-based mesh quality functionals this appears easy to do by weakening the stability property of the local functional to

$$\lim_{\det(A)\to 0} \mathcal{L}(x,A) = M_k < \infty$$

for an increasing sequence $(M_k)_{k\in\mathbb{N}}$, e.g. by starting with a relatively large regularisation parameter δ_r (see Section 3.5 and (3.23)) and then decreasing it. This was not explored due to not being necessary for the examples in this work.

The computational complexity of minimising a hyperelasticity-based functional is very high but could be reduced by some simple techniques.

i) Only optimise the mesh on the coarsest level necessary to accurately represent the geometry and refine the result to obtain the meshes on the finer levels.

ii) In time dependent settings (e.g. with boundary deformations), use a mesh quality functional with low computational complexity whenever possible and optimise the mesh with the hyperelasticity-based functional only when certain quality criteria are violated.

iii) The same can be done when aligning the mesh to a moving (implicit) surface in the sense that the surface can be tracked and the mesh is only realigned with it when certain quality criteria are violated. This has been studied in [Bas16] together with discontinuous Galerkin time discretisations.

iv) Because it is based on the operator resulting from the finite element discretisation of the uniform $\mathbf{D}(u) : \mathbf{D}(v)$ bilinear form, the preconditioner from Chapter 6 cannot give good results for varying optimal scales. This can likely be remedied by using variable coefficients in the bilinear form for the preconditioner, but the exact choice might be delicate.

v) For the surface alignment, the quadratic penalty iteration can likely be made more efficient by applying a more sophisticated strategy for choosing the sequence $\mu_n \to \infty$ of penalty parameters, especially by coupling it with the different terms from the local mesh quality functional.

A

Appendix: Linear solvers

In this section I want to briefly give references for the linear solvers used in this work. If a solver is used with a preconditioner, I will write it as <SOLVER>-<PRECONDITIONER> (e.g. PCG-Jacobi) where possible.

Assume we have a linear system of equations of the form

$$A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n, \text{solve} \quad Ax = b. \tag{A.1}$$

Assume we want to solve this linear system with an iterative solver and are given an initial guess $x^{(0)}$.

The first solver to mention is *Richardson's iteration*

$$x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)}), \tag{A.2}$$

with a parameter $\omega \in \left(0, \frac{1}{\text{spr}(A)}\right)$ [Saa03, Section 13.2.1].

This can be preconditioned with a preconditioner matrix $M^{-1} \in \mathbb{R}^{n\times n}$ to obtain the iteration

$$x^{(k+1)} = x^{(k)} + \omega M^{-1}(b - Ax^{(k)}). \tag{A.3}$$

With the decomposition $A = (D - E - F)$, with the diagonal, lower triangular and upper triangular parts $D, -E, -F$ of A, choosing $M = D$ results in the Jacobi-preconditioned Richardson iteration (also called damped Jacobi iteration)

$$x^{(k+1)} = x^{(k)} + \omega D^{-1}(b - Ax^{(k)}), \tag{A.4}$$

and is simply called *Richardson-Jacobi* in this work. This is mainly used as a *smoother* in the geometric multigrid solver given below.

Another important solver is the *Preconditioned Conjugate Gradient* method ([HS52]), which is a special case of the nonlinear Conjugate Gradient method from Section 4.2.2. This will be denoted by CG (is used unpreconditioned) or PCG-<PRECONDITIONER>.

The last part is the family of *geometric multigrid methods* (see e.g. [Hac85], [Saa03, Chapter 13]), with some well-known cycles, like the V-cycle, W-cycle or F-cycle. In this work, I only use the V-cycle, which I denote with MGV. For a complete solver configuration, one needs to specify the smoothers and the coarse grid solver.

Bibliography

[ABH+15] M. S. Alnæs, J. Blechta, J. H., A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. The FEniCS Project Version 1.5. *Archive of Numerical Software*, 3(100):9–23, 2015.

[AD00] G. Acosta and R. G. Durán. Error Estimates for Q_1 Isoparametric Elements Satisfying a Weak Angle Condition. *SIAM Journal on Numerical Analysis*, 38(4):1073–1088, 2000.

[Ant83] S. S. Antman. Regular and singular problems for large elastic deformations of tubes, wedges, and cylinders. *Archive for Rational Mechanics and Analysis*, 83(1):1–52, 1983.

[Bal76] J. M. Ball. Convexity conditions and existence theorems in nonlinear elasticity. *Archive for Rational Mechanics and Analysis*, 63(4):337–403, 1976.

[Bän91] E. Bänsch. Local Mesh Refinement in 2 and 3 Dimensions. *Impact of Computing in Science and Engineering*, 3:181–191, 1991.

[Bän98] E. Bänsch. Simulation of instationary, incompressible flows. *Acta mathematica Universitatis Comenianae*, 67, no. 1:101–114, 1998.

[Bän01] E. Bänsch. Finite element discretization of the Navier-Stokes equations with a free capillay surface. *Numerische Mathematik*, 88(2):203–235, 2001.

[Bas16] S. Basting. *An interface fitted finite element method for multiphysics simulations*. PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2016.

[BBK15] E. Bänsch, S. Basting, and R. Krahl. Numerical simulation of two-phase flows with heat and mass transfer. *Discrete and Continuous Dynamical Systems*, 35(6):2325–2347, 2015.

[BCR+00] 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*, 33:375–403, 2000.

[BKST15] P. R. Brune, M. G. Knepley, B.F. Smith, and X. Tu. Composing scalable nonlinear algebraic solvers. *SIAM Review*, 57(4):535–565, 2015.

[Boc] S. Bochkanov. `ALGLIB`. www.alglib.net.

[BP13] S. Basting and R. Prignitz. An interface-fitted subspace projection method for finite element simulations of particulate flows. *Computer Methods in Applied Mechanics and Engineering*, 267:133–149, 2013.

[BPS13] 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(10):1282–1296, 2013.

[Bra07] D. Braess. *Finite Elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 3 edition, 2007.

[Bro70] C. G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

[BW13] 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*, 255:228 – 244, 2013.

[Cia78] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. Studies in Mathematics and its Applications. Elsevier Science, 1978.

[Cia88] P. G. Ciarlet. *Mathematical elasticity Vol. I: Three-dimensional elasticity*, volume 20 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam, 1988.

[CN85] P.G. Ciarlet and J. Nečas. Unilateral problems in nonlinear, three-dimensional elasticity. *Archive for Rational Mechanics and Analysis*, 87(4):319–338, 1985.

[CR72] 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(2):217 – 249, 1972.

[Dav59] W. C. Davidon. Variable metric method for minimization. Technical Report ANL-5990 (revised), Argonne National Laboratory, Lemont, Illinois, United States, November 1959.

[DHOT13] 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(2):208–222, 2013.

[DY99] Y. H. Dai and Y. Yuan. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM J. on Optimization*, 10(1):177–182, May 1999.

[Eva98] L. C. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 1998.

[FK02] 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(6):1377–1391, 2002.

[FP63] R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, 1963.

[FR64] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, 1964.

[FR14] S. Frei and T. Richter. A Locally Modified Parametric Finite Element Method for Interface Problems. *SIAM Journal on Numerical Analysis*, 52(5):2315–2334, 2014.

[GK09a] 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(4):3044–3069, 2009.

[GK09b] C. Gross and R. Krause. A recursive trust-region method for non-convex constrained minimization. In M. Bercovier, M. Gander, R. Kornhuber, and O. Widlund, editors, *18th International Conference on Domain Decomposition Methods*, pages 137–144, 2009.

[GKT08] M. Grajewski, M. Köster, and S. Turek. Mathematical and numerical analysis of a robust and efficient grid deformation method in the finite element context. *SIAM Journal on Scientific Computing*, 31(2):1539–1557, 2008.

[GKT10] M. Grajewski, M. Köster, and S. Turek. Numerical analysis and implementational aspects of a new multilevel grid deformation method. *Applied Numerical Mathematics*, 60(8):767–781, 2010. doi:10.1016/j.apnum.2010.03.017.

[GP92] R. Glowinski and O. Pironneau. Finite element methods for Navier-Stokes equations. *Annular Review of Fluid Mechanics*, 24:167–204, 1992.

[Gur78] M. E. Gurtin. On the nonlinear theory of elasticity. In G. M. Penha and L. A. J. Medeiros, editors, *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*, North-Holland Mathematics Studies, pages 237–253. Elsevier Science, 1978.

[Hac85] W. Hackbusch. *Multi-grid methods and applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer, Berlin, 1985.

[Han93] M. R. Hanisch. Multigrid Preconditioning For The Biharmonic Dirichlet Problem. *SIAM Journal on Numerical Analysis*, 30:184–214, 1993.

[Him72] D. M. Himmelblau. *Applied nonlinear programming*. McGraw-Hill Companies, 1972.

[HKT14] M. Hinze, M. Köster, and S. Turek. Space–Time Newton–Multigrid Strategies for Nonstationary Distributed and Boundary Flow Control Problems. In G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, and S. Ulbrich, editors, *Trends in PDE Constrained Optimization*, volume 165, chapter IV. Birkhäuser, 2014.

[Hol15] D. Holzner. Particulate flows with 3D mesh alignment. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, April 2015.

[HR11] W. Huang and R. D. Russel. *Adaptive moving mesh methods*, volume 174 of *Applied Mathematical Sciences*. Springer, 2011.

[HS52] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, December 1952.

[HTK+09] S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska. Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids*, 60(11):1259–1288, 2009.

[HZ05] W. W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on Optimization*, 16(1):170–192, 2005.

[HZ06] W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[Jam76] P. Jamet. Estimations d'erreur pour des éléments finis droits presque dégénérés. *RAIRO Analyse Numérique*, 10(R-1):43–60, 1976.

[JY13] M. Jamil and X. S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013. PMID: 55204.

[KOS+12] M. Köster, A. Ouazzi, F. Schieweck, S. Turek, and P. Zajac. New robust nonconforming finite elements of higher order. *Applied Numerical Mathematics*, 62(3):166 – 184, 2012.

[KT98] S. Kilian and S. Turek. An example for parallel ScaRC and its application to the incompressible Navier–Stokes equations. Preprint 98–06, SFB 359, Universität Heidelberg, 1998.

[LMWZ10] J. Li, J. M. Melenk, B. Wohlmuth, and J. Zou. Optimal a priori estimates for higher order finite elements for elliptic interface problems. *Applied Numerical Mathematics*, 60(1-2):19–37, January 2010.

[Mie05] A. Mielke. Necessary and sufficient conditions for polyconvexity of isotropic functions. *Journal of Convex Analysis*, 12(2):291–314, 2005.

[MMT12] R. Münster, O. Mierka, and S. Turek. Finite element–fictitious boundary methods (FEM–FBM) for particulate flow. *International Journal for Numerical Methods in Fluids*, 69:294–313, 2012.

[MS04] M. D. Mihajlović and D. J. Silvester. Efficient parallel solvers for the biharmonic equation. *Parallel Computing*, 30(1):35 – 55, 2004.

[Mün16] R. Münster. *Efficient Numerical Methods for the Simulation of Particulate and Liquid-Solid Flows*. PhD thesis, Technische Universität Dortmund, 2016.

[Nol78] W. Noll. A general framework for problems in the statics of finite elasticity. In G. M. Penha and L. A. J. Medeiros, editors, *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*, North-Holland Mathematics Studies, pages 363–387. Elsevier Science, 1978.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[Pet73] D. W. Peterson. A review of constraint qualifications in finite-dimensional spaces. *SIAM Review*, 15(3):639–654, 1973.

[Pol69] B. T. Polyak. The conjugate gradient method in extremal problems. *USSR Computational Mathematics and Mathematical Physics*, 9(4):94 – 112, 1969.

[PR69] 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(R1):35–43, 1969.

[Pra57] W. Prager. On ideal locking materials. *Transactions of The Society of Rheology*, 1(1):169–175, 1957.

[Ric15] T. Richter. A monolithic geometric multigrid solver for fluid-structure interactions in ALE formulation. *International Journal for Numerical Methods in Engineering*, 104(5):372–390, 2015. nme.4943.

[Ros60] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[RTH+12] M. Razzaq, S. Turek, J. Hron, H. Damanik, and A. Ouazzi. FEM Multigrid Techniques for Fluid–Structure Interaction with Application to Hemodynamics. *Applied Numerical Mathematics*, 62(9):1156 – 1170, October 2012. 10.1016/j.apnum.2010.12.010.

[Rum96] M. Rumpf. A variational approach to optimal meshes. *Numerische Mathematik*, 72:523–540, 1996.

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.

[Sac86] E. W. Sachs. Broyden's method in Hilbert space. *Mathematical Programming*, 35(1):71–82, 1986.

[SF88] G. Strang and G. J. Fix. *An analysis of the finite element method*. Wellesley-Cambridge Press, 1988.

[SSD03] P. Solin, K. Segeth, and I. Dolezel. *Higher-Order Finite Element Methods*. Studies in Advanced Mathematics. Chapman and Hall/CRC, 2003.

[Sto04] J. Stoer. *Numerische Mathematik 1*. Springer, 9th edition, 2004.

[Syn57] J. L. Synge. *The Hypercircle in Mathematical Physics: A Method for the Approximate Solution of Boundary Value Problems*. Cambridge University Press, 1957.

[TBK06] S. Turek, C. Becker, and S. Kilian. Hardware-oriented numerics and concepts for PDE software. *Future Generation Computer Systems*, 22(1–2):217 – 238, 2006.

[TGB+10] 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*, 6:2247–2265, May 2010. Special Issue Proceedings of ISC 2008. doi:10.1002/cpe.1584.

[Tur99] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*. Lecture Notes in Computational Science and Engineering. Springer-Verlag, 1999.

[Wei12] M. Weismann. The hybrid level-set front-tracking approach. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, December 2012.

[Wic11] T. Wick. Fluid-structure interactions using different mesh motion techniques. *Computers & Structures*, 89(13–14):1456 – 1467, 2011.