Mesh Optimization based on a Neo-Hookean Hyperelasticity Model

---

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

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Malte Schuh

im April 2023

Dissertation
Mesh Optimization based on a Neo-Hookean Hyperelasticity Model

Fakultät für Mathematik
Technische Universtität Dortmund

Tag der mündlichen Prüfung: 21. 06. 2023

Erstgutachter: Prof. Dr. Stefan Turek
Zweitgutachter: Prof. Dr. Christian Meyer

"*So eine Arbeit wird eigentlich nie fertig, man muss sie für fertig erklären, wenn man nach Zeit und Umständen das Mögliche getan hat.*"
Johann Wolfgang von Goethe

Abstract

For industrial applications CFD-simulations have become an important addition to experiments. To perform them the underlying geometry has to be created on a computer and embedded into a computational mesh. This mesh needs to meet certain quality criteria. This mesh needs to meet certain quality criteria, which are not commonly met by many conventional mesh-generation tool. However, automated tools are necessary to simulate experiments with a time-dependent geometry, for example a rising bubble or fluid-structure interaction.
In this thesis, we study the automatic optimization of a mesh by minimizing a neo-hookean hyperelasticity model. The aim of our mesh optimization is to either smoothen a mesh, or to adapt a mesh to a given geometry. In the first part of the thesis we propose a specific energy model from this class and investigate if a solution to the minimization of this specific energy exists or not.
To solve this minimization problem we need to develop an algorithm. After this we have to investigate if this specific energy function fulfills all requirements of this algorithm. The chosen algorithm is an adaptation of Newton's method in a function space. To globalize the convergence of Newton's method we use operator-adaption techniques in a Hilbert space. This makes the algorithm a Quasi-Newton-Method. We proceed to add other elements that are known from optimization so that the algorithm becomes even more robust. Finally we perform several numerical tests to investigate the performance of this method. In our studies we find that for a certain set of parameters the solution of the minimization problem exists. This set of parameters is limited, but the limits are reasonable for most practical use-cases. During our numerical tests we find the method to be stable and robust enough to automatically smoothen a mesh, but to adapt a given mesh to a given geometry our results are unclear: For simulations in two dimensions, the developed method seems to perform well and we get promising results with even just a type of Picard-iteration. For simulations in three dimensions, some adaptations might be necessary and more tests are required.

Acknowledgements

There are many people I would like to thank. First of all, I would like to thank my supervisor, Prof. Dr. Stefan Turek, for giving me the opportunity to research this topic. Also, whenever I had problems, he had an open ear and supported me on the way to find a solution to it. I also would like to thank Prof. Dr. Christian Meyer. Initially, he was not involved in this work, but in the whole process he became a supervisor for the optimization-related part of this work. Without his support, some parts of this work would not have been possible.
Furthermore, I would like to thank all my colleagues from the chair of Applied Mathematics and Numerics for their support and fruitful discussions. While I should list probably all of them, some of them really stood out: Jonas Dünnebacke, Maximilian Esser, Dirk Ribbrock and Peter Zajac for their development of Feat3. Also, whenever I had some special requests for changes or needed support during the implementation you were great and helped me. Naheed Begum, Robert Jendrny, Timm Treskatis and Katharina Wegener: You were great discussion-partners during the whole time and a huge emotional support during all those years. It helped a lot to just discuss my ideas with you. Also, special thanks to Timm for proof-reading my work. Mirco Arndt: You were also a huge emotional support during the whole time.
I also would like to thank my whole family for their support during this time. Without your support this would not have been possible.

Dortmund, April 2023

Malte Schuh

Contents

1

Introduction

In modern science and engineering it is common practice to use computer simulations to predict results. To do so, the physical properties of something have to be modeled. This model is then implemented in special software where it is solved. If one is not familiar with this topic this sounds like an easy step, but it is actually quite challenging. One example to think of is the simulation of a car in a wind tunnel. First of all, the physical behavior of this is modeled by partial differential equations (PDEs), the Navier-Stokes equations. Additionally, the software also needs to „know“ about the wind-tunnel and the car. To keep this picture as simple as possible, a real-world-task that can be compared to this would be to fill everything of the wind-tunnel that is not part of the car with for example cubes. In a software-package, the cubes are called „elements“ and the „net“which is defined by the edges and vertices of the cubes is called „mesh“. The challenge is that for most software-packages the cubes have to share all their vertices with their neighboring cubes.
To avoid this kind of problem, other techniques (for example fictitious boundary methods) have already been developed that could be applied. These techniques can be applied by using a mesh that has an easy structure and is generated very easily. Examples of these techniques can be found in [TWR03], [BCD09], [LV07] or [MI05]. These approaches have in common that they introduce a new type of error: How good can the geometry be represented in these type of meshes?
When the simulation is run on a computer, it is intuitive that smaller elements provide a finer resolution of the solution which might be necessary to capture some important effects. The drawback of a finer resolution is that the simulations take much more time. To account for this, it is common to create a mesh that is fine only where it needs to be and coarser everywhere else.
Both of the goals can be achived using the so-called h-adaptivity which is a powerful tool. h is usually a measurement for the size of the elements, and h-adaptivity roughly works by locally refining (and thus inserting) or merging elements. Some aspects of it are described in [Bän91]. However, this method also has some disadvantages. The most notable one is that either the software needs to be able to handle so called „hanging nodes“ (a vertex

of one cube is in the middle of an edge or a face of another cube) or is limited to a very specific pattern of inserting new elements (which avoids „hanging nodes“, but to achieve this it has to refine a larger area than desired).
In this work, we will not study one of the mentioned methods. Instead, we will focus on another method: r-adaptivity. The idea of this method is that instead of adding new vertices (which is what h-adaptivity does) the already existing vertices are moved so that a better approximation of a geometry or a high resolution mesh in regions of interest is possible. The main advantage of this method is that the connectivity of the vertices does not change. This makes it an attractive method for time-dependent simulations where the geometric setup changes over time, for example a moving object like a rising bubble: Because the connectivity of the vertices does not change, many datastructures can be reused for each timestep. R-adaptivity can also be combined with fictitious boundary methods. This can be a powerful combination: In the first step one creates a simple mesh, and subsequently it gets adapted in every timestep. By doing so, the error that the fictitious boundary method introduces due to its lack of representing the correct geometry can be reduced to a great extend.
However, this leaves the question: How do we move the vertices? In two dimensions we can directly spot a problem: If we move one vertex of a square onto the diagonal of the square, then this square becomes a triangle. Most simulation codes cannot deal with such a situation. And are there other criteria that decide if a mesh is „good“ or „bad“? In this work, we are going to discuss these questions and present an algorithm that solves this problem. The idea to some of the content of this work is the result of some discussions with Jordi Paul after he finished [Pau17]. This thesis is extending his work in some way by investigating another functional with different properties. Since we worked at the same chair and used the same software to work on similar topics (both of us had the same goal: creating a smooth mesh in the software that is developed at the chair, but Jordi Paul used a different energy that has different properties) some parts from this thesis can be more or less identical to his thesis, for example geometry definitions, definitions of finite elements, interpolation estimates, quotes on existence theorems etc. In this work, follow this approach but are going to choose a different functional with different properties. Our main goal is to extend this framework and incorporate Newton's method as a solver for the nonlinear problem. Newton's method seems promising because of its fast local convergence, but it is also challenging at the same time because this convergence is only guaranteed if we are close enough to the solution of the problem. Therefore, we are going to investigate the properties of the chosen functional, Newton's method and other components from the field of optimization to compose a robust solver for this nonlinear problem in a function space. This is a challenge on its own because most of the time, the algorithmic approaches assume a function $F : \mathbb{R}^N \to \mathbb{R}$. While this assumption is often true after the discretisation (which is necessary to solve a problem on a computer), many of these methods modify the hessian of the functional. While this seems viable at first, it becomes a challenge to store the matrix: If the original hessian is sparse, after a few steps with these modifications it becomes a dense matrix. While we could try some so-called low-memory-versions of these modifications, we instead approach the problem in a functional space and discretise the result so that we still have a sparse problem.
This work is structured as follows: In Chapter 2 we are showing theorems from the different mathematical fields and from nonlinear continuum mechanics so that we have a

common base. This base is formulated as seen from someone with a background in the field of numerics, someone with a different background might choose this different. After this we are discussing the basics of PDE-based meshoptimization and meshquality in Chapter 3. Since this is PDE-based we also have to discuss the conditions for the existence of one optimal mesh and line out an algorithm how to compute it. This outline is used to find additional requirements that are imposed by the algorithm (for example on second derivatives). After we have collected all constraints we are analyzing our chosen energy functional in Chapter 4. In Chapter 5 we then present the final implementation of the algorithm, validate it, use it to smoothen meshes and study its properties. After we know that it works in this case we modify it in Chapter 6 to adapt meshes to a given geometry. Finally, we will briefly summarize the work in Chapter 7 and give an outline of other ideas that could be tested as well.

2

Preliminaries

In this chapter we are going to establish some basic notation and recap some definitions and key theorems that we are going to use. This work was done at a numerics-oriented mathematical chair, so we focus our recap on other fields of mathematics and mechanics. However, we also present some definitions and theorems that are familiar in numerics to establish the notation.

2.1. Matrices and Tensors

For $A \in \mathbb{R}^{m\times n}$ (the vector-space of real $m \times n$ matrices) its elements will be denoted by A_{ij}. Equipped with the inner product

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

this vector-space becomes a Hilbert-space.
For square matrices $A \in \mathbb{R}^{n\times n}$ we define

- Tr as the *trace-operator* with $Tr(A) = \sum A_{ii}$
- $\det(A)$ as the determinant of A
- $GL_n := \{A \in \mathbb{R}^{n\times n} | \det(A) \neq 0\}$ the *general linear group* of invertible matrices
- $SL_n := \{A \in \mathbb{R}^{n\times n} | \det(A) > 0\}$ the *special linear group* of invertible matrices with positive determinant
- $SO_n := \{A \in \mathbb{R}^{n\times n} | A^T A = Id_n, \det(A) = 1\}$ the *special orthogonal group*
- $\mathrm{Cof}(A)$ as the cofactor matrix. Let $A^{(i,j)} \in \mathbb{R}^{n-1\times n-1}$ be the matrix obtained from A by deleting row i and column j. Then $\mathrm{Cof}(A)_{i,j} = (-1)^{i+j}\det(A^{(i,j)})$

The inner product and the trace have the following properties:

$$A : B = B : A$$
$$A : B = \mathrm{Tr}(A^T \cdot B)$$

The trace-operator also is invariant to cyclic permutations, which means that

$$\mathrm{Tr}(A \cdot B) = \mathrm{Tr}(B \cdot A)$$
$$\mathrm{Tr}(A \cdot B \cdot C) = \mathrm{Tr}(C \cdot A \cdot B) = Tr(B \cdot C \cdot A)$$

Therefore, the following identity holds as well:

$$A : B = B : A = \mathrm{Tr}(B^T \cdot A) = \mathrm{Tr}(A \cdot B^T)$$

The three *principle invariants* of a three-dimensional tensor A are

$$I_1 = \mathrm{Tr}(A) \tag{2.1.1}$$
$$I_2 = \frac{1}{2}((\mathrm{Tr}(A))^2 - \mathrm{Tr}(A^2)) = \mathrm{Tr}(\mathrm{Cof}(A)) \tag{2.1.2}$$
$$I_3 = \det(A) \tag{2.1.3}$$

2.2. Functionals and derivatives

Since we later are going to discuss minimizing an energy functional we need to remind ourselves of differentiation in Banach spaces.

Definition 2.1 (*Directional derivative*)**:** Let X, Y be real normed vector spaces and $\Omega \subset X$ an open subset, $x \in X$, $h \in X$, $\tau \in \mathbb{R}$ and $F : \Omega \mapsto Y$. If

$$F'(x)h := \lim_{\tau \searrow 0} \frac{F(x+\tau h) - F(x)}{\tau}$$

exists then $F'(x)h$ is called the *directional derivative* of F at x in direction h

Definition 2.2 (*Gâteaux*-derivative)**:** Let X, Y be real normed vector spaces and $\Omega \subset X$ an open subset, $x \in X$, $h \in X$, $\tau \in \mathbb{R}$ and $F : \Omega \mapsto Y$. If $F'(x)h$ exists and there exists $A \in \mathbb{L}(X,Y)$ with

$$F'(x)h = Ah \quad \forall h \in X$$

then F is *Gâteaux*-differentiable in x and A is the *Gâteaux*-derivative of F in x.

Note that if $F : X \to \mathbb{R}$ then $F'(x) \in \mathbb{L}(X, \mathbb{R})$ which is X^*. We will define X^* in Definition 2.8.

Definition 2.3 (*Fréchet*-derivative)**:** Let X, Y be real normed vector spaces and $\Omega \subset X$ an open subset, $x \in X$, $h \in X$, $\tau \in \mathbb{R}$ and $F : \Omega \mapsto Y$. If $F'(x)h$ exists and there exists $A \in \mathbb{L}(X,Y)$ with

$$F'(x)h = Ah \quad \forall h \in X$$
$$\frac{||F(x+h) - F(x) - Ah||}{||h||} \xrightarrow[||h|| \to 0]{} 0$$

then F is *Fréchet*-differentiable in x and A is the *Fréchet*-derivative of F in x.

When comparing the *Gâteaux*- and *Fréchet*-derivative we notice that they are defined similarly, but the *Fréchet*-derivative adds a constraint to the operator A. Therefore, we can conclude that they are calculated in a similar manner and after this we check if this constraint is fulfilled or not. There are many cases where this is not practical, and another way of checking the if the derivative is a *Fréchet*-derivative is the following theorem:

Theorem 2.4: If the *Gâteaux*-derivative $F'(x)$ exists in some neighborhood of x and is continuous at x, then F is also *Fréchet*-differentiable at x and $F'(x)$ is also the *Fréchet*-derivative.

Proof. The proof can be found in [Pat18, Th. 3.3] □

Remark 2.5 (Second derivatives)**:** If $F : \Omega \subset X \mapsto Y$ and $F'(x) \in \mathbb{L}(X,Y)$, then we can apply the same definitions on the operator $F'(x)$ to define the second derivative. The resulting operator $F''(x) \in \mathbb{L}(X,\mathbb{L}(X,Y))$ is applied by $(F''(x)h_1)h_2$ and obviously needs two directions. We can also denote it as $F''(x)[h_1,h_2]$ and we should mention that it is a bilinear form (in h_1 and h_2). Note that for *Fréchet*-differentiable functionals the second derivative is a symmetric bilinear form [Pat18, Th. 3.11]

2.3. Function spaces

Definition 2.6: Let $\Omega \subset \mathbb{R}^n$. We define the space $C_c^\infty(\Omega)$ by

$$C_c^\infty(\Omega) := \{f : f \in C^\infty(\Omega) | \operatorname{supp}(f) \text{ is compact}\}$$

Definition 2.7: Let $\Omega \subset \mathbb{R}^n$, $k \in \mathbb{N}$, $p \in [1,\infty]$. We define the space $L^p(\Omega)$ by the space of equivalence classes of functions $f : \Omega \mapsto \mathbb{R}$ that only differ on $K \subset \Omega$ with $|K| = 0$ and fulfill $||f||_{L^p(\Omega)} < \infty$ with

$$||f||_{L^p(\Omega)} = \begin{cases} (\int_\Omega |f|^p \mathrm{dx})^{\frac{1}{p}} & \text{for } p \neq \infty \\ \operatorname{ess\,sup} |f| & \text{for } p = \infty \end{cases}$$

We also need to remember the definition of the *dual space*:

Definition 2.8: Let X be a a banach space. Its dual space X^* is the space of all linear continuous functionals $x^* : X \mapsto \mathbb{R}$ with the norm

$$||x^*||_{X^*} = \sup\{|x^*(x)| \, | x \in X, ||x||_X \leq 1\}$$

We will denote the application of x^* on x also by (x,x^*) or (x^*,x) and call this the dual pairing. [Sch, p. 69]

Definition 2.9: The equation

$$(x^*, J_x x)_{X^*,X^{**}} := (x,x^*)_{X,X^*}$$

for $x \in X$ and $x^* \in X^*$ defines an isometric mapping $J_x \in \mathbb{L}(X,X^{**})$. A banach space X is called *reflexive* if the mapping J_x is surjective.

[Alt12, Def. 6.7]
To specify the derivatives in the n-dimensional case we need the so-called *multiindex*: For $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ we set

$$|\alpha| := \sum_i \alpha_i \qquad\qquad \alpha! := \prod_i \alpha_i!$$

With the multiindex we set

$$D^\alpha f := D_1^{\alpha_1} D_2^{\alpha_2} \cdots D_n^{\alpha_n} f = \frac{\partial^{|\alpha|} f}{\partial x^\alpha} = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \cdots \partial x_n^{\alpha_n}}$$

and define the set of all partial derivatives of order k with

$$D^k f := \{D^\alpha f : |\alpha| = k\}$$

Definition 2.10: Let $\Omega \subset \mathbb{R}^n$, $k \in \mathbb{N}$, $p \in [1, \infty]$ The Sobolev-Space $W^{k,p}$ is defined by

$$W^{k,p}(\Omega, \mathbb{R}) := \{u \in L^p(\Omega, \mathbb{R}) | D^\alpha u \in L^p(\Omega, \mathbb{R}) \, \forall \alpha \in \mathbb{N}^n, |\alpha| \leq k\}$$

with the norm

$$||u||_{W^{k,p}} := \sum_{|\alpha| \leq k} ||D^\alpha u||_{L^p}$$

and the seminorm

$$|u|_{W^{k,p}} := \sum_{|\alpha| = k} ||D^\alpha u||_{L^p}$$

An alternative and equivalent norm and seminorm is given by

$$||u||_{W^{k,p}}^p := \sum_{|\alpha| \leq k} ||D^\alpha u||_{L^p}^p$$
$$|u|_{W^{k,p}}^p := \sum_{|\alpha| = k} ||D^\alpha u||_{L^p}^p$$

These Sobolev-Spaces are Banach-Spaces [Sch]. For $p = 2$ we denote $H^k := W^{k,p}$. Note that H^k equipped with the dot-product

$$(f, g)_{H^k} := \sum_{|\alpha| \leq k} (D^\alpha f, D^\alpha g)_{L^2} = \sum_{|\alpha| \leq k} \int_\Omega D^\alpha f D^\alpha g$$

is a Hilbert-space. In this case the alternative norm is actually the naturally induced norm from this dot-product which is why for H^k it is the preferred choice.

Definition 2.11: Let $\Omega \subset \mathbb{R}^n$, $k \in \mathbb{N}$, $p \in [1, \infty]$ The Sobolev-Space $W_0^{k,p}$ is defined by

$$W_0^{k,p}(\Omega, \mathbb{R}) := \{u \in W^{k,p}(\Omega, \mathbb{R}) | \exists u_j \in C_c^\infty(\Omega) : ||u - u_j||_{W^{k,p}} \xrightarrow[j \to \infty]{} 0\}$$

Note that if Ω is bounded, the $W^{k,p}(\Omega, \mathbb{R})$ seminorm is a norm on $W_0^{k,p}(\Omega, \mathbb{R})$ that is equivalent to the standard norm [AF08, Cor. 6.29]

2.4. Specialisations for Hilbert-Spaces

In our case we will apply all previous definitions on a functional $F : H_0^1 \mapsto \mathbb{R}$. Since H_0^1 is a Hilbert space we can use the Riesz representation theorem and reformulate a some of these theorems a definitions.

Theorem 2.12 (Riesz representation theorem)**:** Let X be a Hilbert space. Then for every $x^* \in X^*$ there exists a unique Riesz representative $x \in X$ such that

$$x^*(z) = (x,z)_{X,X} \quad \forall z \in X$$

Furthermore, $||x||_X = ||x^*||_{X^*}$

Proof. The proof can be found in [Cla20, Th. 16.1] □

Since this is an isomorphism we can identify the dual space of a Hilbert space with the Hilbert space itself.
With this in mind let us again have a look at Definition 2.3: For $F : X \mapsto \mathbb{R}$ $F'(x) \in X^*$. But if X is a Hilbert space, we can identify X^* with X, so we can actually calculate a Riesz-representative of $F'(x)$ that is in X. We can also have another look at Remark 2.5: If $F''(x) \in \mathbb{L}(X, X^*)$, then we can identify this with $\mathbb{L}(X,X)$. Note that this view is focuses on its behavior if one argument is provided. If you focus on the way this operator behaves if both arguments are provided, this is in the form $F''(x)[h_1,h_2] : X \times X \mapsto \mathbb{R}$ while being linear in h_1 and h_2, so it is a bilinear form.

2.5. Optimization

Now let us recall some theorems and definitions from optimization. We are going to minimize, so let us just define the minimum and remind us on the conditions for its existence. For this part we assume that X is a Hilbert space.

Definition 2.13: Let $f : X \to \mathbb{R}$. $x \in X$ is called a (local) minimum if $f(x) < f(y) \quad \forall y \in \mathcal{B}_\varepsilon(x)$ for some $\varepsilon > 0$

Definition 2.14: A functional $f : X \to \mathbb{R}$ is called weakly lower semicontinuous (w.l.s.c.) if for every weakly convergent sequence $x_n \rightharpoonup x$ in X it follows that

$$f(x) \leq \liminf_{n\to\infty} f(x_n)$$

[DLR15, Def. 3.2]

Theorem 2.15: If $f : X \to \mathbb{R}$ is w.l.s.c. and radially unbounded, i.e.

$$\lim_{||x||\to\infty} f(x) = +\infty$$

then f has a global minimum.

Proof. The proof can be found in [DLR15, Th. 3.1] □

If the functional is directionally differentiable then we can formulate the following condition:

Theorem 2.16 (First order neccesary condition)**:** Let $C \subset X$ be a nonempty subset of the real normed space X and $F : C \to \mathbb{R}$.
For $x \in C$ the direction $v - x \in X$ is called admissible if there exists a sequence $\{t_n\}_{n\in\mathbb{N}}$ with $0 < t_n \to 0$ such that $x + t_n(v - x) \in C \ \forall n \in \mathbb{N}$
Suppose that $\bar{x} \in C$ is a local minimum and $v - \bar{x}$ is an admissible direction. If F is directionally differentiable at $\bar{x}$ in direction $v - \bar{x}$ then

$$F'(\bar{x})(v - \bar{x}) \geq 0$$

Proof. The proof can be found in [DLR15, Th. 3.2] □

In numerical algorithms it is difficult to work with a variational inequality. In the unconstrained case it is easy to obtain the equation $F'(\bar{x})h = 0 \forall h \in X$, but this will not help us much. However, if the optimum lies in the interior of a domain this holds true as well:

Corollary 2.17: If the conditions of Theorem 2.16 hold and $\bar{x}$ lies in the interior then

$$F'(\bar{x})h = 0 \forall h \in X$$

Proof. Let us define the directions $v(\varepsilon) = \bar{x} + \varepsilon h$ and $w(\varepsilon) = \bar{x} - \varepsilon h$.
We can find $\varepsilon > 0$ so that both directions are admissible directions for the variational inequality. Therefore

$$F'(\bar{x})(v(\varepsilon) - \bar{x}) = F'(\bar{x})(\bar{x} + \varepsilon h - \bar{x}) = F'(\bar{x})\varepsilon h \geq 0$$
$$F'(\bar{x})(w(\varepsilon) - \bar{x}) = F'(\bar{x})(\bar{x} - \varepsilon h - \bar{x}) = -F'(\bar{x})\varepsilon h \geq 0$$

Therefore, $F'(\bar{x})h = 0$ □

There are many possible ways to find the minimum, but we will focus on a *descent method*.

Definition 2.18 (Descent direction)**:** For $F : X \mapsto \mathbb{R}$ and a given iterate x_k we call δ_k a *descent direction* if

$$F(x_k + \sigma\delta_k) < F(x_k)$$

for some $\sigma > 0$ and all $\tilde{\sigma}$ with $\sigma \geq \tilde{\sigma} > 0$. σ is called the *step size*.

Lemma 2.19: If the directional derivative in $F'(x_k)\delta_k$ exists Definition 2.18 implies that δ_k is a descent direction $\Leftrightarrow F'(x_k)\delta_k < 0$.

Proof.

$$F(x_k + \sigma\delta_k) < F(x_k)$$

because $\sigma > 0$:

$$\frac{F(x_k + \sigma\delta_k) - F(x_k)}{\sigma} < 0$$

Since this should hold for some $\sigma > 0$ and all $\tilde{\sigma}$ with $\sigma \geq \tilde{\sigma} > 0$ this holds true when we take the limit for $\sigma \to 0$.
Also, if $F'(x_k)\delta_k < 0$ we know that the directional derivative exists. By definition this means that

$$F'(x_k)\delta_k = \lim_{\sigma \searrow 0} \frac{F(x_k + \sigma\delta_k) - F(x_k)}{\sigma} < 0$$

For σ sufficiently small this means that $F(x_k + \sigma\delta_k) < F(x_k)$. □

One obvious choice for δ_k is the direction of $-\nabla F(x_k)$. This leads to the so called *steepest descent* or *gradient method*. Note that in this context, $\nabla F(x_k)$ actually means its Riesz representative (compare Theorem 2.12). (see [DLR15, p. 44f]) A more general approach to the descent directions are the so called *gradient related descent directions* which should satisfy the so called *angle condition*

Definition 2.20 (Angle condition)**:** For $F : X \mapsto \mathbb{R}$ a descent direction δ_k satisfies the *angle condition* if

$$-(\nabla F(x_k), \delta_k)_X \geq \eta ||\nabla F(x_k)||_X ||\delta_k||_X \tag{2.5.1}$$

for some $\eta \in (0,1)$

This condition has a geometrical interpretation: for $F : \mathbb{R}^2 \mapsto \mathbb{R}$,

$$\cos(\theta) = \frac{(-\nabla F(x_k), \delta_k)_X}{||\nabla F(x_k)||_X ||\delta_k||_X}$$

where θ is the angle between the antigradient and the direction δ_k. [DLR15, p. 45] Therefore, with η one basically chooses „how much δ_k should point in direction of $-\nabla F(x_k)$“, so $\eta > 0$ is a very common choice. The *optimal step width* σ_{opt} would be the σ with

$$\sigma_{\text{opt}} = \arg\min_{\sigma \in \mathbb{R}} F(x_k + \sigma\delta_k)$$

but since F is generally nonlinear, this usually is too expensive to calculate. Instead, one usually uses a so called *line search* method. Generally we require the step width σ_k to fulfill the following properties:

$$F(x_k + \sigma_k\delta_k) < F(x_k) \ \forall k = 1, 2, \ldots \tag{2.5.2}$$

$$F(x_k + \sigma_k\delta_k) - F(x_k) \xrightarrow[k\to\infty]{} 0 \Rightarrow \frac{(\nabla F(x_k), \delta_k)_X}{||\delta_k||_X} \xrightarrow[k\to\infty]{} 0 \tag{2.5.3}$$

With these requirements to the step width we can write up a general descent algorithm:

Algorithm 2.1 General descent algorithm

Choose $x_0 \in X$ and set $k = 0$
repeat
  Choose a descent direction δ_k that fulfills the angle condition (2.5.1)
  Find σ_k such that the properties (2.5.2) and (2.5.3) hold
  Set $x_{k+1} = x_k + \sigma_k\delta_k$
  Set $k = k + 1$
until stopping criterion

Theorem 2.21: Let F be continuously Fréchet differentiable and bounded from below. Let $\{x_k\}$, $\{\delta_k\}$ and $\{\sigma_k\}$ be sequences generated by Algorithm 2.1 with (2.5.1), (2.5.2) and (2.5.3) holding. Then

$$\lim_{k\to\infty} \nabla F(x_k) = 0$$

and every accumulation point of $\{x_k\}$ is a stationary point of F.

Proof. See [DLR15, Theorem 4.2] □

There are many possibilities to calculate the step width. The one we are using is the so called *Armijo rule*.

Definition 2.22: Let X be a Banach space, $F : X \to \mathbb{R}$ Fréchet-differentiable, x_k be the current iterate, δ_k a descent direction. A step size σ_k satisfies the *Armijo condition* if

$$F(x_k) - F(x_k + \sigma_k\delta_k) \geq \beta\sigma_k(-\nabla F(x_k), \delta_k) \tag{2.5.4}$$

with $\beta \in (0,1)$. A common choice is $\beta = 10^{-2}$.

Using the notation $\Phi(\sigma) := F(x_k + \sigma\delta_k)$ the Armijo-Condition can be rewritten as

$$\Phi(\sigma) \leq \Phi(0) + \beta\Phi'(0)$$

[DLR15, P. 47].

Theorem 2.23: Let X be a reflexive Hilbert space, $F : X \to \mathbb{R}$ Fréchet-differentiable, x_k be the current iterate, δ_k a descent direction, $\alpha \in (0,1)$, $\beta \in (0,1)$. The Algorithm

1. Choose an initial step width σ and some value for β
2. If σ does not fulfill the Armijo-condition update σ with $\sigma = \alpha\sigma$.

to calculate a step size that fulfills the Armijo-condition converges.

Proof. All the algorithm is doing is taking the limit for $\sigma \to 0$ on (2.22). The existence of the Fréchet-derivative implies the existence of the limit

$$\lim_{\sigma\to 0} \frac{F(x_k) - F(x_k + \sigma\delta_k)}{\sigma} = -F'(x_k)\delta_k$$

Because δ_k is a descent direction we know that $-F'(x_k)\delta_k > 0$ and $-(F'(x_k), \delta_k) > 0$. $\beta \in (0,1)$, so $-F'(x_k)\delta_k > -\beta F'(x_k)\delta_k$. Therefore, there exists σ^* such that

$$F(x_k) - F(x_k + \sigma\delta_k) \geq \beta\sigma(-\nabla F(x_k), \delta_k) \ \forall \sigma \in (0, \sigma^*)$$

□

A common choice is $\alpha = 0.5$. In the rest of the work we will use both common choices for α and β. An alternative line search method would be to meet the *Wolfe-conditions*

$$F(x_k) - F(x_k + \sigma_k \delta_k) \geq \beta \sigma_k (-\nabla F(x_k), \delta_k)$$
$$(\nabla F(x_k + \sigma_k \delta_k), \delta_k) \geq \beta_2 (\nabla F(x_k), \delta_k)$$

or the *strong Wolfe-conditions* (see [DLR15, page 49])

$$F(x_k) - F(x_k + \sigma_k \delta_k) \geq \beta \sigma_k (-\nabla F(x_k), \delta_k)$$
$$|(\nabla F(x_k + \sigma_k \delta_k), \delta_k)| \geq \beta_2 |(\nabla F(x_k), \delta_k)|$$

with $0 < \beta < \beta_2 < 1$. Because the condition (2.5.3) is more difficult to show in practice we also note

Theorem 2.24: Define the level set

$$\mathcal{N}_0^\rho := \{x + d : F(x) \leq F(x_0), \quad ||d|| \leq \rho\}$$

for some $\rho > 0$ and let ∇F be uniformly continuous on it. If the iterates generated by Algorithm 2.1 with $\{\sigma_k\}$ satisfying the Armijo condition (2.5.4) are such that

$$||\delta_k|| \geq \frac{-(\nabla F(x_k), \delta_k)}{||\delta_k||}$$

then $\{\sigma_k\}$ satisfies (2.5.3).

Proof. See [DLR15, Proposition 4.1] □

This can easily be checked after the computation.

2.6. Nonlinear continuum mechanics

Since this work uses some ideas from nonlinear solid continuum mechanics we also need to establish some basic terminology here. If not stated otherwise we follow [Cia88, Ch. 1]. In the field of mechanics the body Ω „before it is deformed“ is said to be the *reference configuration*, and because the whole body with its surfaces is subject to the deformation a mapping is applied onto $\bar{\Omega}$.

Definition 2.25 (Orientation preserving)**:** Let $\gamma : \mathbb{R}^n \to \mathbb{R}^n$, $\gamma \in \mathcal{C}^1$. γ is called *orientation preserving* if $\det \nabla \gamma > 0$.

Definition 2.26 (Deformation)**:** A mapping $\gamma : \bar{\Omega} \to \mathbb{R}^d$ (where d is the spatial dimension, so usually 3) that maps the reference configuration onto a new configuration is called *deformation* if it is *smooth enough*, injective (except possibly on the boundary of Ω) and orientation preserving.

This is not a sharp definition as it misses the regularity, but this is mainly because there exist some applications that require higher regularity than others so this is depending on the application itself.
The deformation gradient is defined by the following definition:

Definition 2.27 (Deformation gradient)**:** For a given deformation γ the deformation gradient is defined by

$$\nabla\gamma_{i,j} = \partial_j \gamma_i$$

The models that are used in mechanics should not depend on the motion and position of the observer. A model that fulfills this property is called *objective*. In the case of energy functionals one can show that this is the case if the energy functional depends on the *right Cauchy-Green* strain tensor:

Definition 2.28 (Right Cauchy-Green strain tensor)**:** For a given deformation γ with the deformation-gradient $\nabla\gamma$ the right Cauchy-Green strain tensor is defined by $C := \nabla\gamma^T \nabla\gamma$.

If a configuration of a body gets deformed, its energy-state changes. A way of modeling this is to define an energy functional. This energy functional assigns the „deviation from the reference configuration“ (as this is basically what the deformation is) a number. For obvious reasons, this number should be objective which is why we are going to use an energy that depends on the right Cauchy-Green tensor.

2.7. Geometric entities and meshes[1]

Definition 2.29 (s-dimensional Simplices)**:** Let $s \leq d$ and $a_0, \ldots, a_d \in \mathbb{R}^d$ such that $\forall i \in \{1, \ldots, s\}$ $(a_i - a_0)$ are pairwise linearly independent.

i) The set

$$S := \left\{ x \in \mathbb{R}^d : x = \sum_{i=0}^{s} \lambda_i a_i \;\text{ with }\; \sum_{i=0}^{s} \lambda_i = 1 \;\text{ and } \lambda_i \geq 0 \forall i \right\} \tag{2.7.1}$$

is called a (non-degenerated) s-dimensional *simplex* in $\mathbb{R}^d$ and $a_0, \ldots, a_s$ are called its vertices.

ii) The λ_i from (2.7.1) are called *barycentric coordinates* of x with regard to S.

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

$$S' := \left\{ x \in \mathbb{R}^d : x = \sum_{i=0}^{r} \lambda_i a'_i \;\text{ with }\; \sum_{i=0}^{r} \lambda_i = 1 \;\text{ and } \lambda_i \geq 0 \forall i \right\}$$

is called r-dimensional sub-simplex of S.

iv) The s-simplex defined by $a_0 = 0$, $a_i = e_i \;\forall i = 1, \ldots, s$ is called the s-dimensional *unit simplex* and denoted by $\hat{S}$ with the points $\hat{x}$

[1] This section follows a similar section in [Pau17] because we used the same software and worked at the same chair

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

$$T(\hat{x}) = a_0 + \sum_{i=1}^{d} \hat{x}_i a_i$$

is called the *parametrization* of S over $\hat{S}$. Because it is linear it is called the $\mathbb{P}_1$-parametrization or -transformation.

Because of the linearity of the $\mathbb{P}_1$-transformation the simplices have straight edges. In contrast to the simplices the hypercubes are more difficult to describe. The reason for this is that there are no barycentric coordinates for hypercubes.

Definition 2.30 (s-dimensional Hypercube)**:** Let $s \leq d$ and $a_0, \ldots, a_{2^s-1} \in \mathbb{R}^d$

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

$s = 2$

$$\hat{Q}'_0 = [-1,1] \times \{-1\} \qquad \hat{Q}'_1 = [-1,1] \times \{1\}$$
$$\hat{Q}'_2 = \{-1\} \times [-1,1] \qquad \hat{Q}'_3 = \{1\} \times [-1,1]$$

$s = 3$

$$\hat{Q}'_0 = [-1,1] \times [-1,1] \times \{-1\} \qquad \hat{Q}'_1 = [-1,1] \times [-1,1] \times \{1\}$$
$$\hat{Q}'_2 = [-1,1] \times \{-1\} \times [-1,1] \qquad \hat{Q}'_3 = [-1,1] \times \{1\} \times [-1,1]$$
$$\hat{Q}'_4 = \{-1\} \times [-1,1] \times [-1,1] \qquad \hat{Q}'_5 = \{1\} \times [-1,1] \times [-1,1]$$

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

ii) Let us define the following functions $\Phi_i : \hat{Q}_s \to \mathbb{R}^d$, $i = 0, \ldots, 2^d - 1$ by

$d = 1$

$$\Phi_0(\hat{x}) = \frac{1}{2}(1 - \hat{x}_0) \qquad \Phi_1(\hat{x}) = \frac{1}{2}(1 + \hat{x}_0)$$

$d = 2$

$$\Phi_0(\hat{x}) = \frac{1}{4}(1-\hat{x}_0)(1-\hat{x}_1) \qquad \Phi_1(\hat{x}) = \frac{1}{4}(1-\hat{x}_0)(1+\hat{x}_1)$$
$$\Phi_2(\hat{x}) = \frac{1}{4}(1+\hat{x}_0)(1-\hat{x}_1) \qquad \Phi_3(\hat{x}) = \frac{1}{4}(1+\hat{x}_0)(1+\hat{x}_1)$$

$d = 3$

$$\Phi_0(\hat{x}) = \frac{1}{8}(1-\hat{x}_0)(1-\hat{x}_1)(1-\hat{x}_2) \quad \Phi_1(\hat{x}) = \frac{1}{8}(1+\hat{x}_0)(1-\hat{x}_1)(1-\hat{x}_2)$$
$$\Phi_2(\hat{x}) = \frac{1}{8}(1-\hat{x}_0)(1+\hat{x}_1)(1-\hat{x}_2) \quad \Phi_3(\hat{x}) = \frac{1}{8}(1+\hat{x}_0)(1+\hat{x}_1)(1-\hat{x}_2)$$
$$\Phi_4(\hat{x}) = \frac{1}{8}(1-\hat{x}_0)(1-\hat{x}_1)(1+\hat{x}_2) \quad \Phi_5(\hat{x}) = \frac{1}{8}(1+\hat{x}_0)(1-\hat{x}_1)(1+\hat{x}_2)$$
$$\Phi_6(\hat{x}) = \frac{1}{8}(1-\hat{x}_0)(1+\hat{x}_1)(1+\hat{x}_2) \quad \Phi_7(\hat{x}) = \frac{1}{8}(1+\hat{x}_0)(1+\hat{x}_1)(1+\hat{x}_2)$$

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

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

and $Q := T\left(\hat{Q}\right)$. Q is called *non-degenerated d-dimensional hypercube* if and only if

$$\forall \hat{x} \in \hat{Q} : \begin{cases} \nabla T(\hat{x}) \in GL_d & s = d \\ \nabla T(\hat{x})^T \nabla T(\hat{x}) \in GL_d & s < d \end{cases}$$

The points $a_0, \ldots, a_{2^s-1}$ are called the *vertices* of Q and T is called the parametrization of Q over $\hat{Q}$. Because it is a bilinear we call it the $\mathbb{Q}_1$-parametrization or $\mathbb{Q}_1$-transformation. For $l \in \{0, \ldots, 2s-1\}$ the sets $T(\hat{Q}'_l)$ are called (s-1)-dimensional sub-hypercubes and are not degenerated if Q is not degenerated.

It is possible to show that the $\mathbb{Q}_1$-transformation $T : \hat{Q} \to Q$ is linear if and only if Q is a parallelpiped.

Definition 2.31: Let $K \subset R^d$

i)

$$h(K) := \sup\{||x_1 - x_0|| : x_0, x_1 \in K\}$$

is called the diameter of K

ii)

$$\rho(K) := 2\sup\{r : \exists x : B_r(x) \subset K\}$$

is called the in-circle diameter of K.

iii)

$$\sigma(K) := \frac{h(K)}{\rho(K)}$$

is called the aspect-ratio of K

Later, we will also use σ as a symbol for a step-size, but from the context it should be clear which meaning σ has.

Definition 2.32 (conformal mesh)**:** 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 if and only if $\bar{\Omega} = \cup_{K \in \mathcal{T}} K$
This mesh is called *conforming* if and only if

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' of K_0: $K' \subset \partial\Omega$ 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 2.33: Let $(\mathcal{T}_i)_{i\in\mathbb{N}}$ be a family of meshes. It is called a *regular family of meshes* if and only if

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

ii) For $h(\mathcal{T}_i) := \max_{K\in\mathcal{T}_i} h(K)$ it holds that

$$\lim_{i\to\infty} h(\mathcal{T}_i) = 0$$

2.8. Finite Elements and parametric families of finite element spaces

This section follows [Pau17, Section 2.3] which in itself follows [Cia78, Chapter 2]

Definition 2.34 (Finite Element)**:** Let $K \subset \mathbb{R}^d$ be closed, $\mathring{K} \neq \emptyset$, $\partial K \in \mathcal{C}^{0,1}$. Let P be a space of real-valued functions defined over the set K and Σ be a finite set of linearly independent linear forms ϕ_i, $1 \leq i \leq N$ defined over the space P. Let further be

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

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

i) 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 $p_i \in P_i \ \ i \in \{1, \ldots, N\}$ 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 for freedom are *node functionals* or *dual basis*.

Definition 2.35 (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 2.36: Let $(K_i, P_{K_i}, \Sigma_{K_i})_{i\in\mathbb{N}}$ be a family of finite elements. It is called *regular* if and only if

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

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

Definition 2.37 (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 $T : \hat{K} \to K$ be a diffeomorphism such that $\forall i = 1,\dots,d$: $R_i \in G$ with $\exists m \in N : G \subseteq \mathbb{P}_m(\hat{S})$. Define

$$P_k := \{p : K \to \mathbb{R} : \ \exists \hat{p} \in \hat{P} : p = \hat{p} \circ T_k^{-1}\}$$
$$\Sigma_k := \{\hat{\Phi}(p \circ T_k) : \ \hat{\Phi} \in \Sigma\}$$

Then (K,P_k,Σ_k) is called

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

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

iii) *subparametrically equivalent* to $(\hat{K},\hat{P},\hat{\Sigma})$ if and only if $G \subsetneq \hat{P}$

iv) *superparametrically equivalent* to $(\hat{K},\hat{P},\hat{\Sigma})$ if and only if $G \supsetneq \hat{P}$

Definition 2.38 (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 of superparametric (with regard to the reference element) if and only if $\forall i = 1,\dots,N$: $(K_i,P_{K_i},\Sigma_{K_i})$ is affine (or isoparametric or subparametrically or superparametrically) equivalent to $(\hat{K},\hat{P},\hat{\Sigma})$. If the family is in either of the three categories we call it *parametric family*.

Definition 2.39: 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

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

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

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

and the conforming Lagrange spaces

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

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

2.9. Interpolation estimates for affine-equivalent finite elements

This section follows [Pau17, Section 2.4] which just collects a few results from [Cia78]

Lemma 2.40: Let $\hat{K}, K \subset \mathbb{R}^d$ be affine equivalent under the mapping $T : \mathbb{R}^d \to \mathbb{R}^d$, $T(x) = Bx + x_0$, $T(\hat{K}) = K$ and open. Let $m \in \mathbb{N}$, $p \in [1,\infty]$ and $v \in W^{m,p}(K)$. Define $\hat{v}(\hat{x}) = v \circ T_K(\hat{x})$. Then $\hat{v} \in W(\hat{K})^{m,p}$ and

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

Analogusly, the estimate

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

Proof. See [Cia78, Theorem 3.1.2]. □

Lemma 2.41: 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

$$W^{k+1,p} \hookrightarrow \mathcal{C}^{k_\Sigma}(\hat{K}) \tag{2.9.1}$$

$$W^{k+1,p} \hookrightarrow W^{m,q}(\hat{K}) \tag{2.9.2}$$

$$P_k(\hat{K}) \subset \hat{P} \subset W^{m,q}(\hat{K}) \tag{2.9.3}$$

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

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

holds.

Proof. See [Cia78, Theorem 3.1.5]. □

Lemma 2.42: 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 (2.9.1)-(2.9.3). Then there exists a constant $C = C(\hat{K}, \hat{P}, \hat{\Sigma})$ such that

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

Proof. See [Cia78, Theorem 3.1.6]. □

Theorem 2.43: 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 $k, l \in \mathbb{N}$ such that

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

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]. □

2.10. Interpolation estimates for finite elements with non-linear transformations

This section follows [Pau17, Section 2.5].
The estimates from Section 2.9 always assume that the transformation $T : \hat{K} \to K$ is linear. While these results can be applied to the $\mathbb{P}_1$-transformation, this is quite a limitation: To archive a better boundary approximation one can use a $\mathbb{P}_k$-transformation with $k > 1$. Another application of these transformations are the so called *isoparametric finite elements*. Also, general hypercube meshes cannot be used in combination with a linear transformation: We mentioned in Definition 2.30 that the $\mathbb{Q}_1$-transformation is linear only for parallelpipeds.

Lemma 2.44: Let $\hat{K}, K \subset \mathbb{R}^d$ be open and bounded such that $\exists T : \hat{K} \to K$ sufficiently smooth, one-to-one and with sufficiently smooth inverse $T^{-1} : K \to \hat{K}$. Let $\hat{v} : \hat{K} \to \mathbb{R}$. If for some $l \geq 0$, $p \in [1, \infty]$ the function $\hat{v} \in W^{l,p}(\hat{K})$ then $v = \hat{v} \circ T^{-1} : K \to \mathbb{R}$ and $v \in W^{l,p}(K)$ and there exist constants C such that

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

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

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

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

Proof. See [Cia78, Theorem 4.3.2]. □

Theorem 2.45: Let $\hat{\Omega} \subset \mathbb{R}^d$ be open with $\mathcal{C}^0$-boundary, $T : \hat{\Omega} \to \mathbb{R}^d$, $T(\hat{\Omega}) = \Omega$ be a $\mathcal{C}^k$-diffeomorphism for some $k \in \mathbb{N}$. Let $\hat{\Pi} \in \mathbb{L}(W^{k+1}(\hat{\Omega}), W^{m,p}(\hat{\Omega}))$ 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 $\Pi \in \mathbb{L}(W^{k+1,p}(\Omega), W^{m,p}(\Omega)$ by

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

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

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

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

The interpolation estimates from Section 2.9 and 2.10 are the basis of the discretisation error $||u - u_h||$ for many different problems and finite element spaces and corresponding norms $||\cdot||$.. Examples for these can be found in [Cia78, Chapter 3-7] or [Bra07]. The details of these are out of scope for this work because this depends on exact setting that is used to solve the underlying partial differential equation.
However, we can recall the reference mapping $T : \hat{K} \mapsto K$ from Definition 2.37 and its global version $T : \hat{\Omega} \to \mathbb{R}^d$ from eg. Theorem 2.45 where we can see the dependence of the interpolation error estimate on the term $\det(\nabla T)$. We also see in Theorem 2.43 and Theorem 2.45 that these estimates require a regular family of meshes $(\mathcal{T}_h)$. Therefore, if we want to see convergence of the discrete solution to the continuous solution in terms of

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

then we have to make sure that $(\mathcal{T}_h)_h$ is a regular and conforming family of meshes. This is surely possible with certain refinement strategies, however it becomes increasingly difficult if the geometry becomes more complex or is not static anymore (e.g. a rising bubble). A global refinement strategy which would reduce the error $||u - u_h||$ usually comes at a large computational cost, and with strategies that just locally refine the mesh we cannot guarantee the regularity of the family $(\mathcal{T}_h)$ a priori. This is where the idea of *r-adaptivity* arises: Instead of globally refining the mesh and therefore creating unnecessary many vertices and thus increasing the computational cost or locally refining the mesh and losing the regularity of the mesh-family the strategy is to move vertices from areas where no high resolution is needed to areas where high resolution is required. In this work we will not discuss how to obtain this information, but how to continue once this information is available so that it is possible to create a „good“ mesh.

3.1. Mesh quality

For optimizing the mesh we need to discuss mesh quality: What is it and how can we measure it? In one dimension this is relatively easy as meshes have only one variable (the

size of a cell), but in two or three dimensions the shape of a cell becomes an aspect of its quality as well. One approach to take this into account is to define an optimal cell, a mapping between the real and the - for this cell - optimal cell and a functional that depends on this mapping. By summing up these values for each cell of a mesh we can quantify the quality of the mesh.
With these thoughts we directly assume that our understanding of mesh quality is of the form

$$Q = \sum_{T \in \mathcal{T}_h} \mu_T \int_T W(x, \gamma) dx \tag{3.1.1}$$

where γ is the mapping from the optimal cell to the actual cell, W measures the local quality of this cell and μ_T is a possible local weight-factor.
So now, let us discuss what we do expect from the local measurement W:

1. The quality of a cell should be frame independent. In other words, it should not matter if we move or rotate the cell.

2. To normalize the functional W should be zero if we have an optimal mesh. The worse the mesh, the higher the value of the functional.

3. If a cell gets compressed very much this cell should be marked as bad and result in a high value for this cell. The same should hold true if a cell becomes very large.

These expectations are also requirements in the field of nonlinear continuum mechanics [Mos14] where they are requirements for energy functios. In the field of continuum mechanics the requirement (1) is met in the following way: Let γ be the mapping between the deformed and the undeformed body, F the energy functional and W the local measurement for the mesh quality. Then the energy functional

$$F(\gamma) = \sum_{T \in \mathcal{T}_h} \mu \int_T W(x, \gamma) dx$$

should depend only on the right Cauchy-Green-Tensor $\nabla\gamma^T \nabla\gamma$. Requirement (3) is expressed by

$$\det(\nabla\gamma) \searrow 0 \Rightarrow W \to \infty \tag{3.1.2}$$

$$||\nabla\gamma|| + ||\mathrm{Cof}(\nabla\gamma)|| + \det(\nabla\gamma) \to \infty \Rightarrow W \to \infty \tag{3.1.3}$$

Because of the geometrical interpretation of these terms these expressions are very reasonable. In three dimensions

- $||\nabla\gamma||_{\mathrm{Fro}}$ expresses the change of length of curvature under the transformation γ [Cia88, Section 1.8]

- $\mathrm{Cof}(\nabla\gamma)$ is related to the change of area under the transformation γ [Cia88, Th. 1.7-1]

- $\det(\nabla\gamma)$ is ratio the volume changes under the transformation γ [Cia88, Section 1.5]

If we now apply this to meshdeformation requirements (3.1.2) and (3.1.3) help to establish some regularity in the resulting mesh: For a 2D-Hypercube mesh it becomes unlikely that the inner angle of a quadrilateral converges to 0. The reason for this is easy to spot: If the inner angle converges to zero, one can either preserve the volume of the cell or the length of all edges, but not both at the same time. If the length of all edges stays the same, the volume must decrease. This relates to $\det(\nabla\gamma) \to 0$ and condition (3.1.2) makes this a less favorable solution. On the other hand, if the volume should be preserved and the length of the edges have to be stretched. This correlates to $||\nabla\gamma||_F \to \infty$ and then condition (3.1.3) makes this less favorable. By combining these conditions we can expect some sort of balance between both extremes that should have inner angles that are „not too bad“.

As we compared this to nonlinear continuum mechanics before, one thought comes up quite naturally: We measure the mesh quality for one cell by the amount of energy that is required to deform an optimal cell to it. The optimal mesh therefore is a mesh that requires the least amount of energy for all deformations. This thought and modeling gives us the advantage to utilize existing theorems from the field of nonlinear continuum mechanics.
From a mathematical point of view it would be nice if the energy would be convex (this would make it easier to show the existence of a minimizer), but in [Cia88, Theorem 4.8-1] it was shown that a convex energy cannot meet the required growth conditions.
Before we continue, we would like to formalize what we mean with expectation 1 (which was formulated in a very loose manner so that it does not hide the main idea): If our functional is of the form

$$F(\gamma) = \sum_{T \in \mathcal{T}_h} \mu \int_T W_T(x, \gamma) dx$$

then we would like to see *translational invariance*:

$$W_T(x, \gamma) = W_T(x, \gamma + c) \ \ \forall c \in \mathbb{R}^n$$

which implies that

$$W_T(x, \gamma) = W_T(x, \nabla\gamma)$$

Translational invariance means that the quality of a cell does not depend on its position but only on its shape. Furthermore, we expect *frame indifference*

$$\forall Q \in SO_d : W(\nabla\gamma) = W(Q\nabla\gamma)$$

which means that the quality of a cell should not depend on the position of the observer. As we mentioned, this can be guaranteed if the W is formulated in the right Cauchy-Green tensor because if $W(\nabla\gamma) = W(C)$ we just need to verify that $C(\nabla\gamma) = C(Q\nabla\gamma)$, and because $C(\nabla\gamma) = \nabla\gamma^T\nabla\gamma$ and $Q \in SO_d$ this is clearly the case. Finally, we expect *isotropy*:

$$\forall Q \in SO_d : W(\nabla\gamma) = W(\nabla\gamma Q)$$

which means that the quality of a cell should not depend on the coordinate system of the reference cell. Here it helps again to formulate W in the right Cauchy-Green-Tensor because $C(\nabla\gamma Q) = Q^T\nabla\gamma^T\nabla\gamma Q$. Since $Q \in SO_d$ we know that this is $C(\nabla\gamma Q) = Q^{-1}\nabla\gamma^T\nabla\gamma Q$,

so this is a similarity transformation of $\nabla\gamma$. If the model is formulated using the invariants of C then this is fulfilled by definition of the invariants.
Because for every matrix $A \in \mathbb{R}^{d\times d}$ there exists a left polar decomposition $A = Q(A^TA)^{\frac{1}{2}}$ with $Q \in SL_d$ and a right polar decomposition $A = (A^TA)^{\frac{1}{2}}Q$ with $Q \in SL_d$ we can use these together with the frame indifference and isotropy to deduce with help of the Rivlin-Erikson Representation Theorem [Cia88, Theorem 3.6-1] that

$$W(x,\gamma) = W(||\nabla\gamma||_F^2, ||\operatorname{Cof}(\nabla\gamma)||_F^2, \det(\nabla\gamma))$$

3.2. Deformations and variations

Now we need to discuss the kind of deformations we are going to allow. Mainly, we do not want to allow any deformation that leads to deterorientation, for example self-intersections. For this, we are going to follow [Pau17, Section 3.1]. Note that the term variations is taken from [Rum96].

Definition 3.1 (The space of admissible deformations)**:** Let $\Omega \subset R^d$, $d = 2,3$ be a bounded domain with

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

i) The set

$$\tilde{\mathcal{D}}_{op} := \left\{\Phi : \Omega \to \mathbb{R}^d : \forall x \in \Omega : \nabla\Phi(x) \in SL_d\right\} \tag{3.2.1}$$

is called the *space of orientation preserving deformations*.

ii) Let Ω_h be 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

$$\tilde{\mathcal{D}}_{op,h} := \tilde{\mathcal{D}}_{op,h}(\mathcal{T}_h) := \tilde{\mathcal{D}}_{op} \cap V_h, \quad V_h = \begin{cases} \mathbb{P}_1 & \hat{K} = \hat{S} \\ \mathbb{Q}_1 & \hat{K} = \hat{Q} \end{cases} \tag{3.2.2}$$

iii) Our main focus in this work are deformations that preserve the boundary $\partial\Omega$. Therefore, we also define the subspace of *boundary preserving deformations*

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

To these spaces a variety of boundary conditions can be applied and constraints can be imposed. Some 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 $\hat{T} : \bar{\Omega} \times SL_d \to \mathbb{R}^{d\times d}$ is a response function for the first Piola-Kichhoff stress and ν is the outer unit normal field, then a traction boundary condition is

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

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 3.8

With this definition of admissible deformations it is possible to formulate everything, but there can be situations where this getting tricky: If $\partial\Omega = \cup_i \Gamma_i$ with Γ_i being a smooth surface segment, the definition of deformations does not prevent vertices of a mesh $\mathcal{T}_h$ on a polygonal approximation Ω_h of Ω from switching between Γ_i and Γ_j with $i \neq j$. This in turn could then worsen the approximation of Ω with Ω_h. As [Pau17] and [Rum96] state this would need combinatorial treatment which does not fit well into a variational setting. Therefore, let us introduce *variations*:

Definition 3.2 (The space of admissible variations)**:** Let $\Omega \subset \mathbb{R}^d$ and $\partial\Omega = \uplus_{m=0}^{d-1} \partial\Omega^m$ where $\partial\Omega^0$ is a set of singular points and $\partial\Omega^m$ are sets of relatively open, smooth m-dimensional manifolds.

i) Then

$$\mathcal{D} := \left\{\Phi \in \tilde{\mathcal{D}}_{bp} : \forall m = 0, \ldots, d-1 : \forall \Gamma \in \partial\Omega^m : \forall x \in \Gamma : \Phi(x) \in \Gamma\right\} \tag{3.2.3}$$

is called the space of admissible *variations*.

ii) If Ω_h is a polygonal approximation of Ω and $\mathcal{T}_h$ a regular conforming mesh on Ω_h and $\forall m = 0, \ldots, d-1$: $\partial\Omega_h^m$ are the sets of polygonal approximations to $\partial\Omega^m$ define the space of *discrete variations* as

$$\mathcal{D}_h = \mathcal{D}_h(\mathcal{T}_h) := \left\{\Phi \in \tilde{\mathcal{D}}_{bp,h} : \forall m = 0, \ldots, d-1 : \forall \Gamma \in \partial\Omega_h^m : \forall x \in \Gamma : \Phi(x) \in \Gamma\right\} \tag{3.2.4}$$

3.3. Reference cells [1]

When introducing the mesh quality we just had mentioned some optimal cell - or reference cell - and not discussed it and its properties.
Firstly, the optimal cell should not depend on any input mesh because this would mean

[1] Jordi Paul used the same software, so the reference cells are identical to those in [Pau17] and therefore, we follow [Pau17] here

that we need always a good input mesh and then mesh-optimization seems somehow pointless. Therefore, it should be independent of the computational domain for the partial differential equation that should be solved on the mesh. One important choice of reference cells are those where all edges and all interior angles are identical. This ensures that one problems cannot arise: Imagine if in two dimensions the reference cell for a quadrilateral would not be a square but rather a rectangle. If now the cell in the mesh is a rectangle as well, it could happen that we map the „short“ edge of the cell in the real mesh onto the „long“ edge of the optimal cell. This could then indicate a bad quality of this cell, when everything that might have been necessary was to just rotate the reference cell.

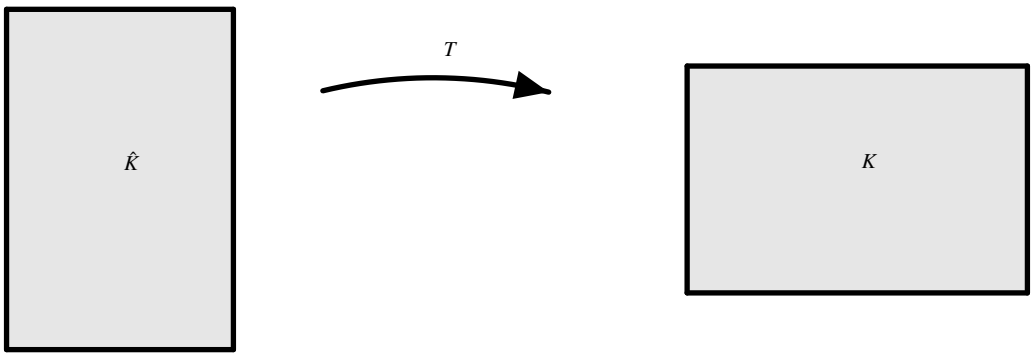


Figure 3.1: Reference cell should be rotated

With this in mind we can define the optimal cell (up to a scaling by volume): For hypercubes, this is easy as the well known unitcube fulfills the requirements:

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

For simplices, we set

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

with

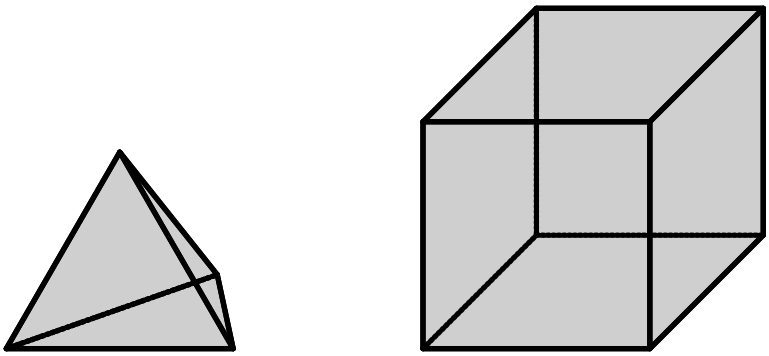
$$\begin{cases} a_0 = (0,0)^T, a_1 = (1,0)^T, a_2 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right)^T & d = 2 \\ a_0 = (0,0,0)^T, a_1 = (1,0,0)^T, a_2 = \left(\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right)^T, a_3 = \left(\frac{1}{2}, \frac{\sqrt{3}}{6}, \frac{\sqrt{6}}{3}\right)^T & d = 3 \end{cases}$$

These choices for $\hat{S}_n$ have the benefit that no combinatorial testing for the evaluation of the local integrand is required because they are invariant with respect to their local numbering. For simplices, one can get all local numberings by taking the even permutations If this property cannot be archived, one has to calculate the energy for all combinations that are possible and meaningful and then take the smallest value of all of them.

For more details about this see [Pau17, Section 3.5.3]

(a) Reference triangle $\hat{S}_2$ (b) Reference square $\hat{Q}_2$

Figure 3.2: Reference elements in 2D



(a) Reference tetrahedron $\hat{S}_3$ (b) Reference hypercube $\hat{Q}_3$

Figure 3.3: Reference elements in 3D

3.4. Existence of minimizer

Let us now discuss the properties we need to show the existence of a minimizer. This part is going to follow [Pau17, Section 3.5.6] [2] and [Cia88, Section 7], which itself is based on [Bal76] where the real pioneering work was done. Since convexity of the energy was ruled out, the concept of *polyconvexity* was introduced by [Bal76]. For our purpose, the following definition taken from [Cia88, Section 4.9] is sufficent

Definition 3.3 (Polyconvexity)**:** A stored energy function $f : \bar{\Omega} \times SL_3 \to \mathbb{R}$ is called *polyconvex* if for each $x \in \bar{\Omega}$ there exists a convex function

$$\hat{f}(x,.) : \mathbb{R}^{3\times 3} \times \mathbb{R}^{3\times 3} \times (0,\infty] \to \mathbb{R}$$

such that

$$f(x,G) = \hat{f}(x,G,\mathrm{Cof}(G),\det(G)) \quad \forall G \in SL_3$$

The next theorems are adaptions of theorems to our situation: The general definition of polyconvexity is found in Definition 3.3 and then required in the existence theorems in [Bal76] and [Cia88], but the energy we are going to choose does not depend on $\mathrm{Cof}\nabla\gamma$ (see (4.1.1)). Therefore, this requirement has to be modified.

[2] As already mentioned, discussions between Jordi Paul and us happend, and in this work we wanted to try a different energy-functional, so in some way this work is also based on the experiences from [Pau17] and a continuation of it

Theorem 3.4 (Existence of minimisers for pure displacement problems)**:** Let $\Omega \subset \mathbb{R}^3$ 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 that $W : \Omega \times SL_3 \to \mathbb{R}$ with the following properties

i) Polyconvexity: For almost all $x \in \Omega$ there exists a convex function $\hat{W}(x,.,.) : \mathbb{R}^{3\times3} \times (0,+\infty) \to \mathbb{R}$ such that $\hat{W}(x,F,\det(F)) = W(x,F) \forall F \in \mathbb{R}^{3\times3}$

ii) Stability

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

iii) Coerciveness: There exist α, β, p, q, r such that

$$\alpha > 0, p \geq 2, q \geq \frac{p}{p-1}, r > 1 :$$
$$W(x,F) \geq \alpha(||F||^p + ||\mathrm{Cof}F||^q + (\det(F))^r) + \beta$$

Let $\Phi_0 \in L^1(\Gamma_0, \mathbb{R}^3)$ such that

$$\begin{aligned}\mathcal{D}_{\Phi_0} := \big\{\Phi \in W^{1,p}(\Omega) : &\mathrm{Cof}\nabla\Phi \in L^q(\Omega), \det(\nabla\Phi) \in L^r(\Omega),\\ &\forall x \in \Omega a.e. : \det(\nabla\Phi)(x) > 0,\\ &\forall x \in \Gamma_0 d\sigma a.e. : \Phi(x) = \Phi_0(x)\big\}\end{aligned}$$

is not empty. Let $f \in L^q(\Omega)$ and $g \in L^s(\Gamma_1)$ with $\frac{1}{p} + \frac{1}{s} = 1$ such that the linear form

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

is continuous and define

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

Under the assumption that $\exists\Phi \in \mathcal{D}_{\Phi_0} : \mathcal{F}(\Phi) < +\infty$ there exists at least one function

$$\Phi^* \in \mathcal{D}_{\Phi_0} : \mathcal{F}(\Phi^*) = \inf_{\Phi\in\mathcal{D}_{\Phi_0}} \mathcal{F}(\Phi)$$

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

Remark 3.5: If we consider a 2d-problem and we choose $p = q = 2$ then the coercivity condition in 3.4 can be modified to

$$W(x,F) \geq \alpha(||F||^p + (\det(F))^r) + \beta$$

because in this case $||F||$ and $||\mathrm{Cof}F||$ are equivalent.

Remark 3.6: We should note the following things:

i) The condition $\mathcal{D}_{\Phi_0} \neq \emptyset$ is a condition on Φ_0 in general as the traces of $W^{1,p}(\Omega)$-functions might not have enough regularity [Cia88, Theorem 6.1-7]. In case of deformations, the boundary deformation still has to admit *some* orientation preserving deformation of Ω, meaning self-intersections of the boundary should not occur.

ii) The resulting problem is a *pure displacement problem*.

iii) The minimizer is not unique in general. This can be observed.

Theorem 3.7 (existence of minimizers 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 $W : \Omega \times SL_3 \to \mathbb{R}$ with the same properties as in Theorem 3.4 (polyconvexity, regularity, coerciveness). Let $\Lambda : SL_3 \to \mathbb{R}$ be a polyconvex function such that

$$\exists \hat{\Lambda} : SL_3 \times SL_3 \times (0, +\infty) \to \mathbb{R} : \quad \forall F \in SL_3 : \Lambda(F) = \hat{\Lambda}(F, \det(F))$$

Let further $\Gamma \subset \mathbb{R}^3$ be closed and $\Phi_0 \in L^1(\Gamma_0, \mathbb{R}^3)$ such that

$$\begin{aligned} \emptyset \neq \mathcal{D}_{\Phi_0,\Gamma} := \{\Phi \in W^{1,p}(\Omega) : & \operatorname{Cof}\nabla\Phi \in L^q(\Omega), \det(\nabla\Phi) \in L^r(\Omega), \\ & \forall x \in \Omega a.e. : \quad \det(\nabla\Phi)(x) > 0, \\ & \forall x \in \Omega a.e. : \quad \Lambda(\nabla\Phi(x)) \leq 0, \\ & \forall x \in \Gamma_0 d\sigma a.e. : \Phi(x) = \Phi_0(x), \\ & \forall x \in \Gamma_2 d\sigma a.e. : \Phi(x) \in \Gamma_2\} \end{aligned}$$

Let $f \in L^q(\Omega)$ and $g \in L^s(\Gamma_1)$ with $\frac{1}{p} + \frac{1}{s} = 1$ such that the linear form

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

is continuous and define

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

Under the assumption that $\exists \Phi \in \mathcal{D}_{\Phi_0,\Gamma} : \mathcal{F}(\Phi) < +\infty$ there exists at least one function

$$\Phi^* \in \mathcal{D}_{\Phi_0,\Gamma} : \mathcal{F}(\Phi^*) = \inf_{\Phi \in \mathcal{D}_{\Phi_0,\Gamma}} \mathcal{F}(\Phi)$$

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

Remark 3.8: We should note

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

ii) The additional boundary condition in Theorem 3.7 is a *unilateral boundary condition of place* as defined in Definition 3.1. It is particular useful for the application for mesh deformations and the underlying mathematical property of the definition of the space of variations (see Definition 3.2)

iii) The function Λ in Theorem 3.7 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.-Vernant stress: Let $\alpha \in \mathbb{R}_+$ and define

$$E := (\nabla\Phi)^T \nabla\Phi - I_3, \quad E_d = E - \frac{1}{3}\mathrm{Tr}(E)I_3$$
$$\Lambda : SL_3 \to \mathbb{R} \quad \Lambda(F) := ||E_d||_F^2 - \alpha$$

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 method.

Theorem 3.9 (Existence of minimizers 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 $W : \Omega \times SL_3 \to \mathbb{R}$ with the same properties as in Theorem 3.4 (polyconvexity, regularity, coerciveness) with $p > 3$.
Let further $\Phi_0 \in L^1(\Gamma_0, \mathbb{R}^3)$ such that

$$\begin{aligned}\emptyset \neq \mathcal{D}_{\Phi_0} := \{\Phi \in W^{1,p}(\Omega) : &\mathrm{Cof}\nabla\Phi \in L^q(\Omega), \det(\nabla\Phi) \in L^r(\Omega),\\ &\forall x \in \Omega a.e.: \quad \det(\nabla\Phi)(x) > 0,\\ &\forall x \in \Gamma_0 d\sigma a.e. : \Phi(x) = \Phi_0(x),\\ &\int_\Omega \det(\nabla\Phi)dx \leq \mathrm{vol}(\Phi(\Omega))\}\end{aligned}$$

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

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

is continuous and define

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

Under the assumption that $\exists\Phi \in \mathcal{D}_{\Phi_0} : \mathcal{F}(\Phi) < +\infty$ there exists at least one function

$$\Phi^* \in \mathcal{D}_{\Phi_0} : \mathcal{F}(\Phi^*) = \inf_{\Phi\in\mathcal{D}_{\Phi_0}} \mathcal{F}(\Phi)$$

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

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

Remark 3.10: We should note that

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 3.4 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 3.7) fulfilling an appropriate condition on the volume.

Remark 3.11: The coercivity condition reads different than one might expect. Its purpose is to ensure there is a lower bound to the functional when minimizing. This is in some way similar to *radially unbound* (Theorem 2.15).

3.5. Optimization algorithm

We have discussed the existence of a minimizer, so now we are discussing how we are going to calculate it and develop an algorithm. The minimizer itself is a function and we are minimizing a functional, so we are looking into *Optimization in Banach spaces*. Each algorithm has its own requirements to the functional, so after this section, we know which aspects of the functional we need to investigate. Therefore, we are not discussing requirements or regularity of γ here. As we have indicated earlier we are going to focus on descent methods, in particular Newton's method. The other options that we are not going to discuss here include

- Nonlinear steepest descent method: This method sets the search-direction as $\delta_k = -F'(\gamma_k)$
- Nonlinear conjugate gradient: $\delta_{k+1} = \beta_k\delta_k - F'(\gamma_k)$ or preconditioned nonlinear conjugate gradient: $\delta_{k+1} = \beta_k\delta_k - B^{-1}F'(\gamma_k)$ with $\delta_0 = -F'(\gamma_k)$

For the nonlinear conjugate gradient method the choice of β_k is not as unique as for the linear conjugate gradient method. In the nonlinear case the different choices have different properties. Some choices are

- The Hestenes-Stiefel-Update [HS52]

$$\beta_{k+1}^{HS} = \frac{\left(B^{-1}F'(\gamma_{k+1}), F'(\gamma_{k+1}) - F'(\gamma_k)\right)}{\left(F'(\gamma_{k+1}) - F'(\gamma_k), \delta_k\right)}$$

- The Fletcher-Reeves-Update [Fle64]

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

- The Polak-Ribiere-Polyak-Update [PR69] [Pol69]

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

  Because this does not guarantee a descent direction this is usually modified:

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

- The Dai-Yuan-Update [DY99]

$$\beta_{k+1}^{DY} = \frac{\left(B^{-1}F'(\gamma_{k+1}), F'(\gamma_k)\right)}{(F'(\gamma_{k+1}) - F'(\gamma_k), \delta_k)}$$

- The Hager-Zhang-Update [HZ05]

$$\beta_{k+1}^{HZ} = \left(\frac{B^{-1}F'(\gamma_{k+1})}{(F'(\gamma_{k+1}) - F'(\gamma_k), \delta_k)}, F'(\gamma_{k+1}) - F'(\gamma_k) - \frac{2||F'(\gamma_{k+1} - F'(\gamma_k)||^2}{(F'(\gamma_{k+1} - F'(\gamma_k), \delta_k)}\delta_k\right)$$

As we showed in Corollary 2.17 we are looking for a solution γ^* such that $F'(\gamma^*)h = 0\ \forall h$ (assuming that the optimum lies in the interior). One possibility to solve this nonlinear equation is the Newton-Scheme. Normally, the Newton-Scheme is presented to solve the equation $G(x) = 0$, but if we apply this general notation from a Banach-Space (see [Deu11, Section 1.2] for example) on the equation $F'(\gamma^*)h = 0$ we can derive the following algorithm:

Algorithm 3.1 Newton-Scheme for optimization

given: Initial guess γ_0
while do $||F'(\gamma_k)|| \neq 0$
  Solve $F''[\delta_k, \phi](\gamma_k) = -F'(\gamma_k)\phi\ \forall\phi$
  Update: $\gamma_{k+1} = \gamma_k + \delta_k$
end while

Therefore, we now know that the second derivative should exist. Let us now investigate if this method qualifies as a descent-method as this would allow us to apply Theorem 2.21. Recall Lemma 2.19 which states that a direction is a descent direction if

$$F'(\gamma_k)\delta_k < 0$$

In this case, we have

$$F'(\gamma_k)\phi = -F''[\delta_k, \phi](\gamma_k)\ \forall\phi \tag{3.5.1}$$

so clearly if we can pick $\phi = \delta_k$ then

$$F'(\gamma_k)\delta_k = -F''[\delta_k, \delta_k](\gamma_k)$$

Therefore, if $-F''[\delta_k, \delta_k](\gamma_k) < 0$ then this qualifies as a descent method. This is the case if $F''[h_1, h_2](\gamma_k)$ is a positive definite operator. The benefit of this requirement is that it helps to ensure that equation (3.5.1) has a solution. If the operator is not positive definite, we need an alternative. Therefore, we directly are going to resort to a Quasi-Newton method. The main idea is that instead of $F''[h_1, h_2](\gamma_k)$ we use an approximation $H[h_1, h_2](\gamma_k)$. One can show that if this approximation H_k fulfills the Dennis-Moré-Condition

$$\frac{||(H_k - F''(\gamma^*))(\gamma_{k+1} - \gamma_k)||}{||(\gamma_{k+1} - \gamma_k)||} \xrightarrow[k\to\infty]{} 0 \tag{3.5.2}$$

we still can expect to observe superlinear convergence for $\{\gamma_k\}$ [BMdY11].
Since we are going to use an approximation it might be good to test if the search-direction δ_k is still a gradient-based search-direction and fulfills the angle-condition (Definition 2.20)

$$-(\nabla F(\gamma_k), \delta_k)_X \geq \eta ||\nabla F(\gamma_k)||_X ||\delta_k||_X$$

If it does not fulfill the angle-condition we are going to use $-\nabla F(\gamma_k)$ as search-direction - or rather its Riesz-representative, which we have to calculate. After we know the update direction we are going to use the Algorithm from Theorem 2.23 to generate a step size that fulfills the Armijo-Condition (recall (2.5.4))

$$F(x_k) - F(x_k + \sigma_k \delta_k) \geq \beta \sigma_k (-\nabla F(x_k), \delta_k)$$

After we have determined both update-direction and step-size we have to update:

$$\gamma_{k+1} = \gamma_k + \sigma_k \delta_k$$

This update will not be performed directly, but instead it will be done by moving the coordinates of the mesh. This is possible because γ is mapping the optimal cell to the current cell, so the effect of a change in γ is a change of the mesh.

4

Choice and analysis of the energy functional

4.1. Choice of the Energy functional

In [Pau17] or [Rum96] the idea was to use a hyperelasticity-functional as a mesh-quality functional. Here, we are going to follow the same approach. We will use a so called *Neo-Hooke* model.

Definition 4.1 (Neo-Hooke-Model)**:** A model is called a *Neo-Hookean-Model* if it only depends on the first and third invariant (see (2.1.1)-(2.1.3)) of the right Cauchy-Green-Tensor. The dependency on the first invariant must be linear.

We are going to use the following energy functional:

$$F(\gamma) = \int \frac{1}{2}\lambda[\log(\det(\nabla\gamma))]^2 + \frac{1}{2}\mu[\mathrm{Tr}(\nabla\gamma^T\nabla\gamma) - d] - \mu\log(\det(\nabla\gamma))\,\mathrm{dx} \tag{4.1.1}$$

where λ and μ are the known Lamé constants. For our purpose we can assume that both are positive.
The obvious benefit of this kind of functional is that it is cheaper to compute or evaluate than a model which depends on all three invariants or the cofactor matrix (which appears in the original definition of polyconvexity). This is important to us because mesh-optimization is almost always done as a preprocessing for a simulation, so it should not take too much computational time. On a side note we would like to mention that it is well-known in continuum mechanics that a model which should reflect large deformations cannot be linear.

4.2. Analysis of the energy function

Objectivity

Let us quickly check that we can reformulate it depending on the right Cauchy-Green tensor of the deformation γ:

$$F(C) = \int \frac{1}{2}\lambda[\log(\sqrt{\det(C)})]^2 + \frac{1}{2}\mu[\mathrm{Tr}(C) - d] - \mu\log(\sqrt{\det(C)})\,\mathrm{dx} \tag{4.2.1}$$

By doing so, we directly showed that this energy functional is objective and that it is indeed a Neo-Hooke model.

Growth conditions

Recall the growth-conditions (3.1.2) and (3.1.3):

$$\det(\nabla\gamma) \searrow 0 \Rightarrow W \to \infty$$
$$||\nabla\gamma|| + ||\,\mathrm{Cof}(\nabla\gamma)|| + \det(\nabla\gamma) \to \infty \Rightarrow W \to \infty$$

Let us now verify that our model fulfills them:

Proof. If $\det(\nabla\gamma) \searrow 0$ then $\log(\det(\nabla\gamma)) \to -\infty$ and thus $[\log(\det(\nabla\gamma))]^2 \to +\infty$. Therefore:

$$\underbrace{\frac{1}{2}\lambda[\log(\det(\nabla\gamma))]^2}_{\to\infty} + \frac{1}{2}\mu\underbrace{[\mathrm{Tr}(\nabla\gamma^T\nabla\gamma) - d]}_{\substack{=\mathrm{Tr}(C)-d \\ =||\nabla\gamma||_F^2-d>-d}} \underbrace{-\mu\log(\det(\nabla\gamma))}_{\substack{\to-\infty \\ \to\infty}} \xrightarrow[\det(\nabla\gamma)\to 0]{} \infty$$

Therefore (3.1.2) holds.

For the other condition let us split the investigation by splitting the left hand side into its three different terms. This is a valid idea because we know (because γ is orientation preserving) that all three terms are positive. If $\det(\nabla\gamma) \to \infty$, then

$$\log(\det(\nabla\gamma)) \to +\infty$$
$$[\log(\det(\nabla\gamma))]^2 \to +\infty$$

Since this is of the type $x^2 - x$ for $x \to \infty$ we know that for $\det(\nabla\gamma) \to \infty$

$$\frac{1}{2}\lambda[\log(\det(\nabla\gamma))]^2 - \mu\log(\det(\nabla\gamma)) \to \infty$$

Since

$$\frac{1}{2}\mu[\mathrm{Tr}(\nabla\gamma^T\nabla\gamma) - d] = \frac{1}{2}\mu[||\nabla\gamma||^2 - d] > -\frac{1}{2}\mu d$$

so if $\det(\nabla\gamma) \to \infty$ then $W \to \infty$.
Also, if $||\nabla\gamma|| \to \infty$ we know that

$$\exists C \in \mathbb{R} : \frac{1}{2}\lambda[\log(\det(\nabla\gamma))]^2 - \mu\log(\det(\nabla\gamma)) > C$$

and because

$$\frac{1}{2}\mu[\mathrm{Tr}(\nabla\gamma^T\nabla\gamma)-d]=\frac{1}{2}\mu[||\nabla\gamma||^2-d]$$

the growth condition (3.1.3) is also fulfilled. What remains is: Is there a possibility that $||\mathrm{Cof}\nabla\gamma||\to\infty$ but $||\nabla\gamma||<C_1$ and $\det(\nabla\gamma)<C_2$?
To answer this let us reformulate the second growth condition: Let $\tilde{f}:SL_n\to\mathbb{R}$,

$$\tilde{f}(A)=||A||_F+||\mathrm{Cof}(A)||_F+\det(A)$$

If $\tilde{f}(\nabla\gamma)\to\infty\Rightarrow W\to\infty$.
For $n=2$ we can calculate that

$$A=\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix}\Rightarrow\mathrm{Cof}(A)=\begin{pmatrix}a_{22}&-a_{21}\\-a_{12}&a_{11}\end{pmatrix}$$

so obviously, $||A||_F=||\mathrm{Cof}(A)||_F$ and we do not need any further investigation.
For $n=3$ we can calculate that for

$$A=\begin{pmatrix}a_{11}&a_{12}&a_{13}\\a_{21}&a_{22}&a_{23}\\a_{31}&a_{32}&a_{33}\end{pmatrix}\Rightarrow\mathrm{Cof}(A)=\begin{pmatrix}a_{22}a_{33}-a_{23}a_{32}&a_{23}a_{31}-a_{21}a_{33}&a_{21}a_{32}-a_{22}a_{31}\\a_{13}a_{32}-a_{12}a_{33}&a_{11}a_{33}-a_{13}a_{31}&a_{12}a_{31}-a_{11}a_{32}\\a_{12}a_{23}-a_{13}a_{22}&a_{13}a_{21}-a_{11}a_{23}&a_{11}a_{22}-a_{12}a_{21}\end{pmatrix}$$

From this we can deduce that if $||\mathrm{Cof}(A)||_F\to\infty$ then for one of its entries (which have the form of) $a_{ij}a_{kl}-a_{mn}a_{op}$

$$|a_{ij}a_{kl}-a_{mn}a_{op}|\to\infty$$

which can only be true if

$$|a_{ij}a_{kl}|\to\infty\text{ or}\qquad\qquad|a_{mn}a_{op}|\to\infty$$

which can only happen if one of the factors $\to\infty$. However, if this is the case then $||A||_F\to\infty$. Therefore, we can conclude that the energy fulfills (3.1.3) as well. □

Therefore, we now know that this energy functional meets the requirements that we have taken over from the field of nonlinear continuum mechanics. We should note that the „Stability“-requirement from Theorem 3.4 is identical to the growth-condition (3.1.2).

Polyconvexity

To investigate if the stored energy function

$$f(G)=\frac{1}{2}\lambda[\log(\det(G))]^2+\frac{1}{2}\mu[\mathrm{Tr}(G^TG)-d]-\mu\log(\det(G))$$

is polyconvex we split it into

$$\begin{aligned}f_1(G)&=\frac{1}{2}\mu[\mathrm{Tr}(G^TG)-d]=\frac{1}{2}\mu[||G||^2_{\mathrm{Fro}}-d]\\f_2(\delta)&=\frac{1}{2}\lambda[\log(\delta)]^2-\mu\log(\delta)\\f(\gamma)&=f_1(\nabla\gamma)+f_2(\det(\nabla\gamma))\end{aligned}$$

It is obvious that f_1 is convex. However, for f_2 we can calculate that

$$f_2''(\delta) = \frac{\lambda + \mu - 2\lambda \log(\delta)}{\delta^2}$$

One can calculate that $f_2''(\delta) < 0$ if $\delta > \exp\left(\frac{\mu}{\lambda} + 1\right)$, so clearly f_2 is not a globally convex function. However, we can note that $f_2(\delta)$ is convex for all $\delta < \exp\left(\frac{\mu}{\lambda} + 1\right)$. We are just going to keep this in mind and move on for now, but we will come back to this later.

Coerciveness

For a 2d problem we need to show that

$$f(\gamma) \geq \alpha(||\nabla\gamma||_F^p + \det(\nabla\gamma)^r) + \beta$$

with $\alpha > 0$, $p \geq 2$,$r > 1$. Recall the functional

$$\begin{aligned} f(\gamma) &= \frac{1}{2}\lambda[\log(\det(\nabla\gamma))]^2 + \frac{1}{2}\mu[\mathrm{Tr}(\nabla\gamma^T\nabla\gamma) - d] - \mu\log(\det(\nabla\gamma)) \\ &= \frac{1}{2}\lambda[\log(\det(\nabla\gamma))]^2 - \mu\log(\det(\nabla\gamma)) + \frac{1}{2}\mu[||\nabla\gamma||_F^2 - d] \end{aligned}$$

Just by comparison we see that the term $||\nabla\gamma||_F^2$ is a big advantage because this means we can pick $p = 2$ which allows us to solve the problem in the Hilbert space H^1. However, the terms depending on the determinant are problematic as they do not fulfill the required growth condition ($\log(x)^2 - \log(x) > x$ holds true only on a short interval, and we would need $\log(x)^2 - \log(x) > x^r$ with $r > 1$.) Hence, we have to modify the stored energy function.
For 3d, we would have to show that

$$f(\gamma) \geq \alpha(||\nabla\gamma||_F^p + ||\mathrm{Cof}\nabla\gamma||_F^q + \det(\nabla\gamma)^r) + \beta$$

Since $f(\gamma)$ does not depend on $\mathrm{Cof}\nabla\gamma$ this cannot be shown directly, but with another modification (similar to the one we do for the determinant) it would be possible.

4.3. Modification of the stored energy function

As we have seen in the previous section our energy is neither polyconvex nor coercive. In both cases the term that depends on the determinant of $\nabla\gamma$ is the one that is violating the requirement. Therefore, we are now modifying our stored energy function:

$$\begin{aligned} \tilde{f}(\gamma) &= \frac{1}{2}\lambda[\log(\det(\nabla\gamma))]^2 + \frac{1}{2}\mu[\mathrm{Tr}(\nabla\gamma^T\nabla\gamma) - d] - \mu\log(\det(\nabla\gamma)) \\ &\quad + \kappa\left(\left(\det(\nabla\gamma) - \exp\left(\frac{\mu}{\lambda} + 1\right)\right)_+\right)^2 \end{aligned}$$

with some $\kappa \in \mathbb{R}$. We are going to determine constraints for κ during the analysis. Before we continue here let us discuss the reasoning behind this modification: In practice, we

do not want to actually have to deal with the additional term, so we want it to be zero in all „real life situations“. In the field of nonlinear continuum mechanics,μ and λ are fixed material parameters that cannot be chosen free, they are chosen by the material that has to be simulated. In our case, however, they are free parameters. We know (as stated shortly after (3.1.3)) that $\det(\nabla\gamma)$ is the ratio of the volume change under the transformation γ. Since in our case the γ maps from the optimal cell to the cell on the mesh this means that the term will only get activated if one cell has a much volume that is $\exp\left(\frac{\mu}{\lambda}+1\right)$ times bigger than the optimal cell, so we should always be able to pick μ and λ in such a way that this does not happen. For polyconvexity we now have to investigate

$$f_1(G) = \frac{1}{2}\mu[\mathrm{Tr}(G^T G) - d] = \frac{1}{2}\mu[||G||^2_{\mathrm{Fro}} - d]$$

and

$$f_2(\delta) = \begin{cases} \frac{1}{2}\lambda[\log(\delta)]^2 - \mu\log(\delta) & \delta \leq \exp\left(\frac{\mu}{\lambda}+1\right) \\ \frac{1}{2}\lambda[\log(\delta)]^2 - \mu\log(\delta) + \kappa\left(\delta - \exp\left(\frac{\mu}{\lambda}+1\right)\right)^2 & \delta > \exp\left(\frac{\mu}{\lambda}+1\right) \end{cases}$$

It is obvious that f_1 is convex, but f_2 needs further investigation. We have dealt with the first case before and know that f_2 is convex on part. For $\delta > \exp\left(\frac{\mu}{\lambda}+1\right)$ we calculate that

$$f_2''(\delta) = \frac{\lambda + \mu - 2\lambda\log(\delta) + 2\delta^2\kappa}{\delta^2}$$

Clearly, one can find κ that this is always positive (for example $\kappa = \lambda$), so f_2 is convex on this part as well. Because $f_2 \in \mathcal{C}^1$ f_2 is globally convex and our stored energy function is polyconvex.
For coerciveness we have to find

$$\begin{aligned} \alpha > 0, p \geq 2, r > 1 : \\ f(x,F) \geq \alpha(||F||_F^p + (\det(F))^r) + \beta \end{aligned}$$

With this modification we now clearly chose $p = 2$ and $r = 2$. With the same split of f we can estimate that

$$\begin{aligned} f_1(G) &= \frac{1}{2}\mu[\mathrm{Tr}(G^T G) - d] \\ &= \frac{1}{2}\mu[||G||^2_{\mathrm{F}} - d] \end{aligned}$$

so $\alpha \leq \frac{\mu}{2}$ works fine, and $\beta = \frac{-2d}{\mu}$ is a possible choice. For f_2 it becomes less obvious:
Firstly, we need to note that f_2 is continuous on $(0,\infty)$ and that we do not have to consider the case $\det(\nabla\gamma) \to 0$ because we just showed that for this case $f \to \infty$, so surely the estimate holds.

For $\det(\nabla\gamma) > \exp\left(\frac{\mu}{\lambda}+1\right)$ we know

$$\begin{aligned}
f_2(\delta) &= \frac{1}{2}\lambda[\log(\delta)]^2 - \mu\log(\delta) + \kappa\left(\delta - \exp\left(\frac{\mu}{\lambda}+1\right)\right)^2 \\
&= \frac{1}{2}\lambda\left(\log(\delta) - \frac{\mu}{\lambda}\right)^2 - \frac{\mu^2}{2\lambda} + \kappa\left(\delta - \exp\left(\frac{\mu}{\lambda}+1\right)\right)^2 \\
&\geq -\frac{\mu^2}{2\lambda} + \kappa\left(\delta - \exp\left(\frac{\mu}{\lambda}+1\right)\right)^2 \\
&= -\frac{\mu^2}{2\lambda} + \varepsilon\kappa\delta^2 + \kappa(1-\varepsilon)\left(\delta - \frac{1}{1-\varepsilon}\exp\left(\frac{\mu}{\lambda}+1\right)\right)^2 \\
&+ \kappa(1-\varepsilon)\left(\left(\frac{-1}{1-\varepsilon}\exp\left(\frac{\mu}{\lambda}+1\right)\right)^2 + \frac{1}{1-\varepsilon}\left(\exp\left(\frac{\mu}{\lambda}+1\right)\right)^2\right) \\
&\geq -\frac{\mu^2}{2\lambda} + \varepsilon\kappa\delta^2 + \kappa(1-\varepsilon)\left(\left(\frac{-1}{1-\varepsilon}\exp\left(\frac{\mu}{\lambda}+1\right)\right)^2 + \frac{1}{1-\varepsilon}\left(\exp\left(\frac{\mu}{\lambda}+1\right)\right)^2\right)
\end{aligned}$$

with some $\varepsilon \in (0,1)$. Therefore, we have shown the coerciveness if $\det(\nabla\gamma) \in \left(\exp\left(\frac{\mu}{\lambda}+1\right), +\infty\right)$. For $\det(\nabla\gamma) \in \left(0, \exp\left(\frac{\mu}{\lambda}+1\right)\right)$ we know that $f_2 \to \infty$ for $\det(\nabla\gamma) \to 0$. We also know that f_2 is a continuous function and therefore has a minimum on this interval. Therefore, it is possible to find some $\tilde{\beta}$ such that

$$f_2(\delta) \geq -\frac{\mu^2}{2\lambda} + \varepsilon\kappa\delta^2 + \kappa(1-\varepsilon)\left(\left(\frac{-1}{1-\varepsilon}\exp\left(\frac{\mu}{\lambda}+1\right)\right)^2 + \frac{1}{1-\varepsilon}\left(\exp\left(\frac{\mu}{\lambda}+1\right)\right)^2\right) + \tilde{\beta}$$

with the ε from above. This means that our function is coercive.

4.4. Analysis of the derivatives

From Section 3.5 we know that we need the first and second derivative of the functional. Formally we derive

$$F'(\gamma)\varphi = \int [\lambda\log(\det(\nabla\gamma))(\nabla\gamma)^{-T} - \mu(\nabla\gamma)^{-T} + \mu\nabla\gamma] : \nabla\varphi \, dx \tag{4.4.1}$$

$$\begin{aligned}
F''(\gamma)[\varphi,\psi] = \int &\lambda[(\nabla\gamma)^{-T} : \nabla\psi][(\nabla\gamma)^{-T} : \nabla\varphi] \\
&+ \lambda\log(\det(\nabla\gamma))(-(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}) : \nabla\varphi \\
&- \mu(-(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}) : \nabla\varphi \\
&+ \mu\nabla\psi : \nabla\varphi dx
\end{aligned} \tag{4.4.2}$$

as candidates for the derivatives. For the presented algorithm we have to solve the partial differential equation

$$F''(\gamma)[\varphi,\delta] = -F'(\gamma)\varphi \, \forall\varphi \in \mathcal{V}$$

in some space $\mathcal{V}$. From Theorem 3.4 we know that this is going to be some subspace of $H^1(\Omega)$, and because we restrict ourselves to the case that some part of the boundary does

not move we can restrict us to a subspace of $H_0^1(\Omega)$. We also know that the candidates for the derivatives are not Fréchet-derivatives on the whole space $H_0^1(\Omega)$, so during our analysis we are going to find further restrictions. For our analysis, we are going to equip $H_0^1(\Omega)$ it with the dot-product

$$(u,v) = \int_\Omega \nabla u : \nabla v \mathrm{dx}$$

and its induced norm $||u||^2 = (u,u)$. If we use a more general variant of the Riesz representation Theorem 2.12, for example from [Sch, Theorem 6.4] we need to check that the right hand side $F'(\gamma)\varphi$ and the bilinear form $F''(\gamma)[\varphi,\psi]$ are continuous and that $F''(\gamma)[\varphi,\psi]$ is symmetric and coercive.

Continuity of the derivatives

For this part we need to keep in mind that because of the existence Theorems 3.4-3.9 we can assume that $\gamma \in W^{1,p}(\Omega)$, $\det(\gamma) \in L^r(\Omega)$ and we chose $p = 2$ and $r = 2$. Because γ is in fact a transformation we can also expect γ to be a diffeomorphism. It is not unrealistic to assume that $\det(\nabla\gamma) \in L^\infty(\Omega)$ because as we noted earlier, $\det(\nabla\gamma)$ reflects the change of volume under the transformation γ (shortly after (3.1.3)). Also, $||\nabla\gamma||_F \in L^\infty(\Omega)$ and $||(\nabla\gamma)^{-1}||_F \in L^\infty(\Omega)$ is a save assumption because $||\nabla\gamma||_F$ reflects the change of the length of curvature (as we mentioned at the same place). Both last assumptions can only be made because we expect Ω to be bounded in our applications: There, we will be mapping a bounded domain onto a bounded domain. Later we will also need that $\overrightarrow{((\nabla\gamma)^{-T})}(x) \otimes \overrightarrow{((\nabla\gamma)^{-T})}(x) \in L^\infty(\Omega)$. The notation will become clear soon.
For $F'(\gamma)\varphi$ we can calculate that

$$\begin{aligned}
F'(\gamma)\varphi &= \int [\lambda \log(\det(\nabla\gamma))(\nabla\gamma)^{-T} - \mu(\nabla\gamma)^{-T} + \mu\nabla\gamma] : \nabla\varphi \,\mathrm{dx} \\
&\leq \underbrace{\sup_{x\in\Omega}(\lambda\log(\det(\nabla\gamma(x))) - \mu)}_{:=C} \int (\nabla\gamma)^{-T} : \nabla\varphi \,\mathrm{dx} + \mu(\nabla\gamma, \nabla\varphi)_{L^2} \\
&= C((\nabla\gamma)^{-T}, \nabla\varphi)_{L^2} + \mu(\nabla\gamma, \nabla\varphi)_{L^2} \\
&\leq C||(\nabla\gamma)^{-T}||_{L^2}||\nabla\varphi||_{L^2} + \mu||\nabla\gamma||_{L^2}||\nabla\varphi||_{L^2} \\
&= (C||(\nabla\gamma)^{-T}||_{L^2} + \mu||\nabla\gamma||_{L^2})||\varphi||_{H^1}
\end{aligned}$$

where we used the Cauchy-Schwarz-inequality and then our choice of norm for the chosen space. For $F''(\gamma)[\varphi,\psi]$ we set

$$\begin{aligned}
B_1(\gamma)[\varphi,\psi] &= \int \lambda[(\nabla\gamma)^{-T} : \nabla\psi][(\nabla\gamma)^{-T} : \nabla\varphi]\mathrm{dx} \\
B_2(\gamma)[\varphi,\psi] &= \int (\lambda\log(\det(\nabla\gamma)) - \mu)(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}) : \nabla\varphi \mathrm{dx} \\
B_3(\gamma)[\varphi,\psi] &= \int \mu\nabla\psi : \nabla\varphi \mathrm{dx}
\end{aligned}$$

such that

$$F''(\gamma)[\varphi,\psi] = B_1(\gamma)[\varphi,\psi] + B_2(\gamma)[\varphi,\psi] + B_3(\gamma)[\varphi,\psi]$$

and calculate for B_1:

$$\begin{aligned} B_1(\gamma)[\varphi,\psi] &= \int \lambda[(\nabla\gamma)^{-T} : \nabla\psi][(\nabla\gamma)^{-T} : \nabla\varphi]\mathrm{dx} \\ &= \lambda \int \sum_{i,j} ((\nabla\gamma)^{-T})_i (\nabla\psi)_i ((\nabla\gamma)^{-T})_j (\nabla\varphi)_j \end{aligned}$$

where we rearranged the values from those matrices from $\mathbb{R}^{d\times d}$ into vectors. In the next few lines we are going to denote this like this:

$$\begin{aligned} B_1(\gamma)[\varphi,\psi] &= \lambda \int \sum_{i,j} \overrightarrow{((\nabla\gamma)^{-T})}_i \overrightarrow{(\nabla\psi)}_i \overrightarrow{((\nabla\gamma)^{-T})}_j \overrightarrow{(\nabla\varphi)}_j \\ &= \lambda \int \left(\overrightarrow{\nabla\varphi}, \left(\overrightarrow{((\nabla\gamma)^{-T})} \otimes \overrightarrow{((\nabla\gamma)^{-T})} \right) \overrightarrow{\nabla\psi} \right)_2 \\ &\leq \lambda \int \left(\overrightarrow{\nabla\varphi}, c\overrightarrow{\nabla\psi} \right)_2 \end{aligned}$$

where

$$c = \sup_{x\in\Omega} \left\{ |\Lambda(x)| : \Lambda(x) \text{ eigenvalue of } \overrightarrow{((\nabla\gamma)^{-T})}(x) \otimes \overrightarrow{((\nabla\gamma)^{-T})}(x) \right\}$$

This in only possible if $\overrightarrow{((\nabla\gamma)^{-T})}(x) \otimes \overrightarrow{((\nabla\gamma)^{-T})}(x) \in L^\infty(\Omega)$. We can reformulate this estimate as

$$B_1(\gamma)[\varphi,\psi] \leq c\lambda\,(\nabla\varphi,\nabla\psi)_{L^2} \leq \lambda c ||\varphi||_{H^1} ||\psi||_{H^1}$$

where in the end we used again our choice of norm for the space H_0^1.
For B_2 we can calculate

$$\begin{aligned} B_2(\gamma)[\varphi,\psi] &= \int (\lambda \log(\det(\nabla\gamma)) - \mu)(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}) : \nabla\varphi\mathrm{dx} \\ &\leq \underbrace{\sup_{x\in\Omega}(\lambda \log(\det(\nabla\gamma(x))) - \mu)}_{:=C} \int (\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}) : \nabla\varphi\mathrm{dx} \\ &= C(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}, \nabla\varphi)_{L^2} \\ &\leq C||(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}||_{L^2} ||\nabla\varphi||_{L^2} \end{aligned}$$

$$\begin{aligned} ||(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}||_{L^2}^2 &= \int ||(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}||_F^2 \\ &\leq \int ||(\nabla\gamma)^{-T}||_F^2 ||(\nabla\psi)^T||_F^2 ||(\nabla\gamma)^{-T}||_F^2 \\ &\leq \sup_{x\in\Omega} ||(\nabla\gamma)^{-T}||_F^4 \int ||\nabla\psi||_F^2 \\ &= \sup_{x\in\Omega} ||(\nabla\gamma)^{-T}||_F^4 ||\nabla\psi||_{L^2}^2 \end{aligned}$$

Therefore, in total we have that

$$B_2(\gamma)[\varphi,\psi] \leq \tilde{C}_2||\nabla\psi||_{L^2}||\nabla\varphi||_{L^2} = \tilde{C}_2||\nabla\psi||_{H^1}||\nabla\varphi||_{H^1}$$

Note that the last equality only holds because of our choice of space and norm.
B_3 is obvious:

$$\begin{aligned} B_3(\gamma)[\varphi,\psi] &= \int \mu\nabla\psi : \nabla\varphi \mathrm{dx} \\ &= \mu(\psi,\varphi)_{H^1} \\ &\leq \mu||\psi||_{H^1}||\varphi||_{H^1} \end{aligned}$$

as this is just the application of the Cauchy-Schwarz inequality on the chosen dot-product. By showing the continuity of both derivatives, we have shown that both derivatives (4.4.1) and (4.4.2) are Fréchet-derivatives if $\det(\nabla\gamma) \in L^\infty(\Omega)$, $||\nabla\gamma||_F \in L^\infty(\Omega)$, $||(\nabla\gamma)^{-1}||_F \in L^\infty(\Omega)$ and $\overrightarrow{((\nabla\gamma)^{-T})}(x) \otimes \overrightarrow{((\nabla\gamma)^{-T})}(x) \in L^\infty(\Omega)$ (see Theorem 2.4).

Symmetry of the second derivative

It is obvious that B_1 and B_3 are symmetric. To investigate B_2 we hide the scalar value of $(\lambda\log(\det(\nabla\gamma)) - \mu)$ in b:

$$\begin{aligned} B_2(\gamma)[\varphi,\psi] &= \int b((\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T}) : \nabla\varphi \mathrm{dx} \\ &= \int b\,\mathrm{Tr}\left((\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T})(\nabla\varphi)^T\right) \mathrm{dx} \\ &= \int b\,\mathrm{Tr}\left((\nabla\varphi)^T(\nabla\gamma)^{-T}(\nabla\psi)^T(\nabla\gamma)^{-T})\right) \mathrm{dx} \\ &= \int b\,\mathrm{Tr}\left((\nabla\gamma)^{-T})(\nabla\varphi)^T(\nabla\gamma)^{-T}(\nabla\psi)^T\right) \mathrm{dx} \\ &= \int b((\nabla\gamma)^{-T}(\nabla\varphi)^T(\nabla\gamma)^{-T}) : \nabla\psi \mathrm{dx} \\ &= B_2(\gamma)[\psi,\varphi] \end{aligned}$$

Therefore, $F''(\gamma)[\varphi,\psi]$ is symmetric.

Coerciveness of the second derivative

We need to show that

$$||u||^2 \leq CF''(\gamma)[u,u]$$

for some $C > 0$. Because $B_3(\gamma)[u,u] = b||u||^2$ and $B_1(\gamma)[u,u] \geq 0$ we know that

$$||u||^2 \leq B_1(\gamma)[u,u] + B_3(\gamma)[u,u] \tag{4.4.3}$$

but we cannot draw any conclusion for B_2. However, because we come from a minimization problem we know that if our iterate γ_k is close to γ_* then the second derivative must be coercive, but until we are close to it we know nothing about it.

Since we anticipated this problem earlier we presented the Quasi-Newton methods where approximations H_k are used instead of the second derivative. From here on, there are many different possibilities with different approaches to approximate the second derivative, for example the BFGS or DFP-method [NW06, Section 6.1], rank-1 updates [NW06, Section 6.2] or the Broyden-class [NW06, Section 6.3]. These methods work by taking an initial approximation of the Hessian and then update it from iteration to iteration. These methods have the disadvantage that they can produce dense matrices H_k which cannot be stored for large problems. There do exist some „low-memory"-variants for some of these algorithms (for example [Yam08] proposed a method), but we are not going to use them. Instead, we are going to adapt a technique that has been carried out at this chair a few times already, for example in [Man17]: In [Man17] the overall idea was to take a parameter $\alpha \in (0,1]$ such that for $\alpha \to 0$ the nonlinear solver was more like a fixed point iteration and for $\alpha = 1$ the nonlinear solver became a Newton solver.
We are adapting this by setting

$$H_k(\gamma,\alpha)[\varphi,\psi] = B_1(\gamma)[\varphi,\psi] + \alpha B_2(\gamma)[\varphi,\psi] + B_3(\gamma)[\varphi,\psi]$$

with $\alpha \in (0,1]$. This is an approximation of the Hessian on the continuous level which is then discretized. The advantage is that this preserves the sparsity of H_k. Of course, we have to choose α such that H_k is still coercive, but as we see in (4.4.3) this is possible because we can choose α arbitrary small. This setting guarantees that the partial differential equation in the Newton-Scheme is always solvable and that the resulting searchdirection is a descent direction: We are solving the equation

$$H_k(\gamma,\alpha)[\varphi,\delta] = -F'(\gamma)\varphi \ \forall\varphi$$

where δ is the searchdirection. By choosing $\varphi = \delta$ we have

$$-H_k(\gamma,\alpha)[\delta,\delta] = F'(\gamma)\delta$$

and because we just showed that H_k is coercive and thus a positive definite bilinear form we have that

$$F'(\gamma)\delta < 0$$

which is just what we stated in Lemma 2.19.
Another advantage is that as long as $\alpha \to 1$ if the nonlinear solver converges then this fulfills the Dennis-Moré-Condition (3.5.2) even in a stronger sense: The condition only requires for the values to converge, but in our case the whole operator converges.
Clearly, it is a benefit of this energy function that the approximation of $F''(\gamma)[\varphi,\psi]$ for the Quasi-Newton-Scheme arises almost „naturally".

5

Numerical results I

Now we have analyzed the aspects of the functional and know that this is a basis to mesh adaptations. Therefore, we are going to implement it. Remember the outline of the Algorithm 3.1. In this chapter, we are going to adapt it to everything we learned during the analysis and then verify the results. To do this, we are going to restrict ourselves to a simple task: we are going to assume that a mesh generator created something that we would like to smoothen.

5.1. Implementing the algorithm

Let us now briefly put together all components that we discussed. We have the initial mesh from the mesh generator as an initial guess for the optimal mesh, and the according situation is remembered as γ_0. We also choose an initial value α_0 for the approximation of the second derivative, a parameter η for the angle condition, our common parameters $\alpha_{\text{Armijo}} = 0.5$ and $\beta = 10^{-2}$ for the Armijo-Scheme and of course some μ and λ. In each iteration we have to solve

$$H_k(\gamma, \alpha)[\varphi, \delta] = -F'(\gamma)\varphi \ \forall \varphi \in H_0^1(\Omega) \tag{5.1.1}$$

$$\int \nabla s : \nabla \varphi \mathrm{dx} = F'(\gamma)\varphi \ \forall \varphi \in H_0^1(\Omega) \tag{5.1.2}$$

to obtain a searchdirection $\tilde{\delta}$ (or rather $\tilde{\delta}_k$ as it belongs to this step) and the Riesz-representative s of the gradient $F'(\gamma)$. After we have done that we check if our searchdirection $\tilde{\delta}_k$ fulfills the angle-condition

$$-(s, \tilde{\delta}_k)_X \geq \eta ||s||_X ||\tilde{\delta}_k||_X$$

where the dot-product and the norm are discrete versions of our chosen H_0^1-norm. If this not the case then we discard the „Quasi-Newton-direction“ $\tilde{\delta}_k$ and use $-s$ as searchdirection. This check helps us in an unexpected way as well: If our solution of δ_k is calculated with an iterative solver, we might try to use a relatively „loose“ stopping criterion for it. If

it was „too loose“, the check of the angle-condition still prevents us from using it. Let us denote the chosen search-direction as δ_k. We now use the Armijo-Scheme from Theorem 2.23 to calculate a stepsize σ_k that fulfills the Armijo-Condition (2.5.4):

$$F(\gamma_k) - F(\gamma_k + \sigma_k \delta_k) \geq 10^{-2} \sigma_k (-\nabla F(\gamma_k), \delta_k)$$

where we directly use the common value $\beta = 10^{-2}$. Note that we earlier calculated $s = \nabla F(\gamma_k)$, the dot-product $(s, \tilde{\delta}_k)$ and $||s||$ for the angle condition, so all we have to do here is to update the coordinates of the mesh and evaluate the functional. Since $\gamma_k + \sigma_k \delta_k$ is the next iterate, the cost of one Armijo-step is approximately one evaluation of the functional which is an integral over the domain. We should also probably note that we do need to modify-the Armijo-condition slightly and require that $F(\gamma_k + \sigma_k \delta_k)$ is in fact a real number. This might seem odd at first, but since we have no initial constraint on the stepsize σ_k it could happen that the mesh contains degenerated elements if σ_k is too large. These degenerated elements can cause *NaN* or ∞ as functional value. Some programming languages treat these values in ways that the condition would be fulfilled, but we do not want any degenerated cells. After we updated the mesh we also update α_k for the approximation of the second derivative. In summary we have the following algorithm:

Algorithm to smoothen a mesh

given: Initial guess γ_0 (in form of an initial mesh), α_0, η, $\sigma_k^{(0)}$, k_{min}
while $||F'(\gamma_k)|| \neq 0$ **do**
  if $k > k_{min}$ **then**
    Solve $H_k(\gamma_k, \alpha_k)[\delta_k, \varphi] = -F'(\gamma_k)\varphi \ \forall \varphi$
  end if
  Solve $\int \nabla s : \nabla \varphi dx = F'(\gamma)\varphi \ \forall \varphi$
  if $-(s, \delta_k) < \eta ||s|| ||\delta_k||$ or $k \leq k_{min}$ **then**
    $\delta_k \leftarrow s$
  end if
  Set $\sigma_k = \sigma_k^{(0)}$
  while $F(\gamma_k) - F(\gamma_k + \sigma_k \delta_k) < 10^{-2} \sigma_k (-\nabla F(\gamma_k), \delta_k)$ **do**
    $\sigma_k \leftarrow 0.5 \sigma_k$
  end while
  Update: $\gamma_{k+1} = \gamma_k + \delta_k$
  Calculate α_{k+1}
end while

The parameter k_{min} has the purpose of omitting to solve the H_k-equation in the first iterations. This is mostly because of our experience that during the first few iterations there are hardly benefits when trying out the Quasi-Newton-direction, but it is (numerically) expensive to assemble the linear system and then solve it. So far, we have not explained how we are going to calculate α_{k+1} and only stated at the end of our analysis that every heuristic which guarantees $\alpha_k \to 1$ for $||F'(\gamma_k)|| \to 0$ works. Our choice is an adaptation of a strategy used in [Man17] and [Jened]

$$\alpha_{k+1} = \min \left\{ 1, \alpha_k \cdot g \left(||F'(\gamma_k)|| / ||F'(\gamma_{k-1})|| \right) \right\}$$

with

$$g(x) = 0.15 + \frac{6.51657}{3.66657 + \exp(1.38629x)}$$

which is shown in Figure 5.1 and going through the three points $(0, 1.5464)$, $(1,1)$, $(2, 0.4814)$

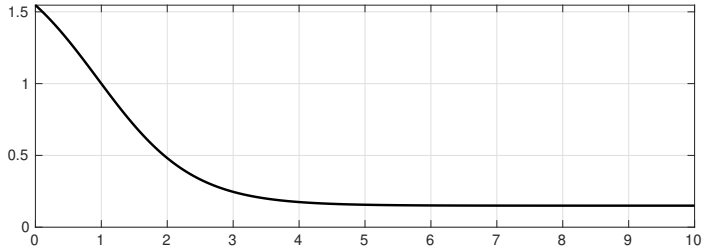


Figure 5.1: The function for calculating the new α

In some edge cases (for example if the linear solver for δ_k would not converge) we manually set $\alpha_{k+1} = 0.1\alpha_k$.
We also need to discuss how we are setting up the reference cells. Since we discussed that the reference cell has a specific shape, we set up a list of d-dimensional volumes where we store the desired volume for each cell. We then compose the mapping γ from the transformation τ that is already included in the finite element software (going from the „reference element for the basis functions“ $\tilde{K}$ to the actual element K) and a mapping from a scaled version of the optimal cell to the cell $\tilde{K}$ so that the implementation is as displayed in the Figures 5.2 and 5.3. All results shown in this work are a done on the servers at the Technische Universität Dortmund using the software FEAT3 [RGR$^+$21] which is part of the FEAT and FeatFlow software family ([Tur99], [TGB$^+$10], [KOS$^+$12], [HKT14]).

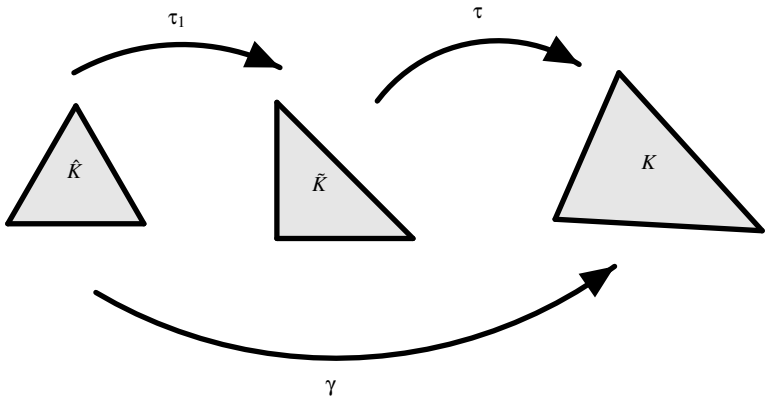


Figure 5.2: Composing the mapping γ for triangles

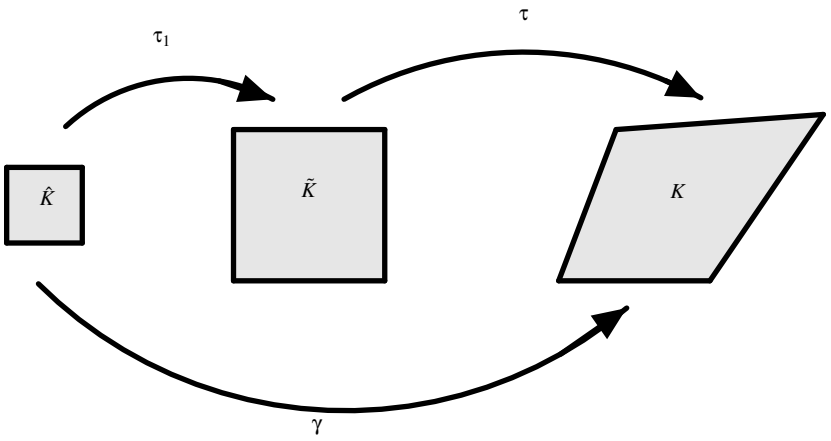


Figure 5.3: Composing the mapping γ for quads

The main difference between these cases is that for the quadrilaterals we just need to scale them, but for the triangles we do also need to change the shape because in Feat the transformation τ is defined for a rectangular triangle, but as we discussed in Section 3.3 the mapping γ must start on an equilateral triangle. This composition is a valid approach because we know that our functional is invariant with respect to translations and rotations of the reference cell. Therefore, all we have to implement is the mapping τ_1 in such a way that it depends on the prescribed volume for the specific cell K.

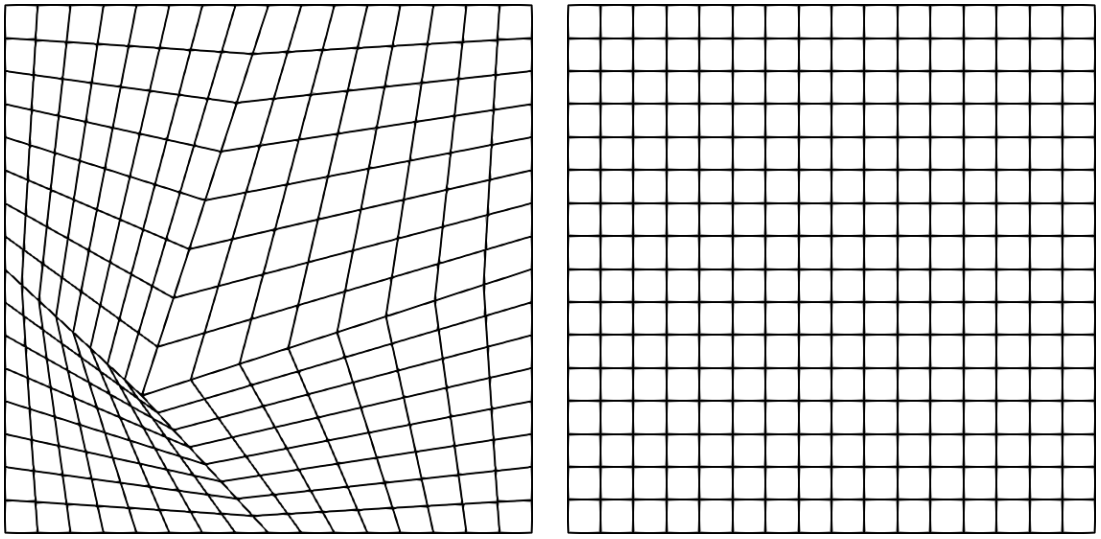
5.2. First tests - smoothing a mesh

Let us start with a testcase where we are able to predict the final mesh: We take the square $[0,1]^2$, refine it once, then move the midpoint to $(0.26, 0.26)$ and use this as a starting mesh. We then prescribe that all quadrilaterals shall have the same volume. By intuition we know that as long as the vertices on the boundary allow it we can deform all quadrilaterals to optimal cells, and because of our refinement method the positions of the vertices on the boundary allow this. For all tests, we are going to use $\mathbb{P}_1$ or $\mathbb{Q}_1$ finite elements.

Validation on a coarse mesh

If we refine the described mesh three more times then we start with the mesh in Figure 5.4a. As a start we choose $\lambda = 1$, $\mu = 10$, $\alpha_0 = 0.3$ and $\eta = 0.6$ To solve the linear problems we choose Umfpack [Dav04] so that we do not introduce any other error here. The final mesh in Figure 5.4b looks as expected, but for a better evaluation we need to have a look at the quantities presented in Table 5.1. Note that η_k is the value calculated for the angle-condition.

As expected, we see that the algorithm converges. The convergence is in three different aspects: First of all, we notice that the Newton-defect converges to zero. This is mathematically expected because we explained in Corollary 2.17 that the necessary condition



(a) Initial mesh (b) Final mesh

Figure 5.4: First test: Smoothing a unitsquare

k	σ_k	Armojo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α_k
1	1	0	1.16E+01	2.11E+00	2.18E-01	0.77	0.30
2	1	0	3.12E+00	4.50E-01	1.74E-02	0.96	0.27
3	1	0	7.85E-01	1.40E-01	2.29E-03	1.00	0.38
4	1	0	3.17E-01	3.94E-02	1.48E-04	1.00	0.55
5	1	0	8.36E-02	8.06E-03	2.64E-06	1.00	0.74
6	1	0	1.15E-02	8.78E-04	2.13E-13	1.00	1.00
7	1	0	3.76E-06	1.84E-07	9.56E-17	0.97	1.00
8	1	0	6.06E-13	2.10E-14	3.16E-31	0.97	1.00

Table 5.1: Validation for the first testcase

for a minimum is $F'(\gamma^*) = 0$ and the Newton-defect is $F'(\gamma_k)\varphi$ with some testfunction φ. Second, we notice a convergence in $F(\gamma_k)$. This is also expected because $F(\gamma^*)$ is the value of the functional with the optimal mesh which is what we are calculating. We can also notice that $||\delta|| \to 0$. This might seem obvious, but nevertheless we want to point out the reason for this: If we are calculating an optimal mesh, at some point we do not need to move the coordinates „a lot“. The movement of the coordinates is $\sigma_k\delta_k$, so it seems obvious that $\delta_k \to 0$. This information is really useful: If we switch to iterative solvers for both linear problems, we now know that $\vec{0}$ as a starting vector is a good choice that becomes better in each Newton-Iteration. Unfortunately, it is not possible to investigate $||\gamma_k - \gamma^*||$ because we do not explicitly compose γ, and by our choice of function spaces we would have to investigate $||\nabla\gamma_k - \nabla\gamma^*||_{L^2}$. Even if we apply our knowledge that γ^* is the identity, we do not see chances to find a mesh to compute this. The reason for this is that by our composition of γ (see Figures 5.2 and 5.3) γ depends on the transformation τ. (τ is the transformation from the finite element reference element to the actual mesh.)

While this has many advantages (for example that we could apply higher order transformations which help representing curved boundaries), the downside is that γ depends on the mesh which should be deformed. However, to compute $||\nabla\gamma_k - \nabla\gamma^*||_{L^2}$ we would need to integrate over an optimal mesh because this is how γ is defined. To overcome this problem and still get an idea of the convergence rate, we take a look at the euclidean norm $||\vec{\gamma}_k - \vec{\gamma}^*||_2$ where $\vec{\gamma}_k$ are the coordinates of the mesh in iteration k. This is a valid replacement since the coordinates of the mesh are the image of γ. We had to generate these numbers in the postprocessing, which is why we trust them only for the first 5 iterations.

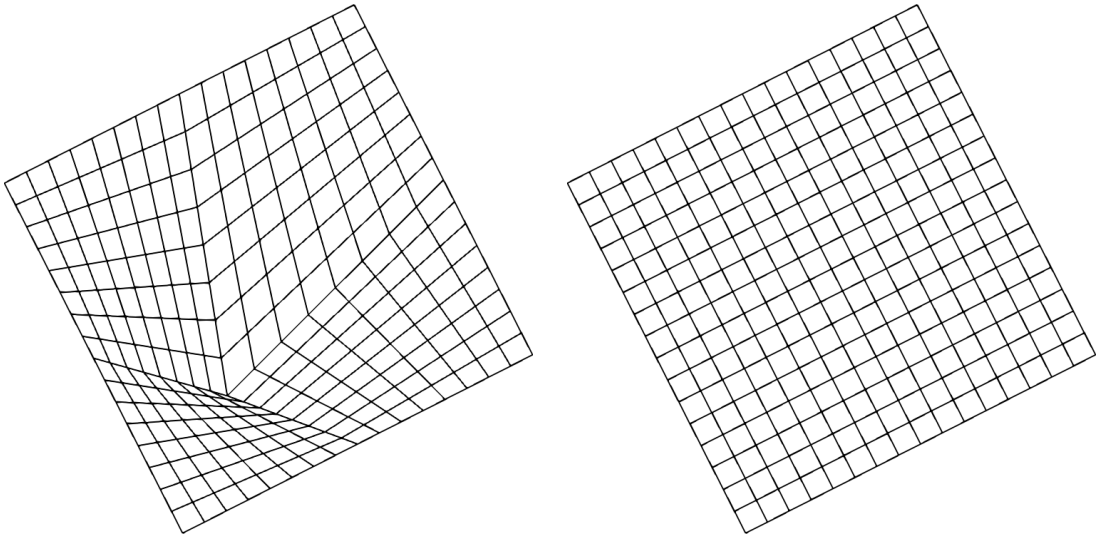
k	$\|\|\vec{\gamma}_k - \vec{\gamma}^*\|\|_2$
0	3.48E+02
1	7.03E+01
2	1.88E+01
3	4.36E+00
4	5.76E-01
5	1.63E-05

Table 5.2: Convergence of the coordinates

In Table 5.2 we see that the coordinates converge faster than linear, so this is showing us that the approach seems to work.

Rotated unitsquare

In our motivation and analysis we showed that the values are independent of the orientation of the mesh. While this might seem obvious, we still choose to test the impact of the orientation of the mesh by repeating the previous test and rotating the mesh by 27°. In Figure 5.5b we see that it works as well, and if we compare the date from Table 5.3 with Table 5.1 we see that overall it gets the same results, the only noticeable difference is that it took one more Armijo-step. The reason for this is that while the functional value does not depend on the rotation of the mesh, the chosen dot-product for our problem does. Since we use the dot-product on several occasions during our algorithm (for example to calculate the Riesz representative of $F'(\gamma_k)$) we have to expect differences in the convergence behavior, but the final mesh should be identical if we first rotate the mesh and then smoothen it or the other way around.



(a) Initial mesh (b) Final mesh

Figure 5.5: Test: Rotated unitsquare

k	σ_k	Armojo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α_k
1	1	0	1.16E+01	2.11E+00	2.18E-01	0.77	0.30
2	1	0	3.12E+00	4.50E-01	1.74E-02	0.96	0.27
3	1	0	7.85E-01	1.40E-01	2.29E-03	1.00	0.38
4	1	0	3.17E-01	3.94E-02	1.48E-04	1.00	0.55
5	1	0	8.36E-02	8.06E-03	2.64E-06	1.00	0.74
6	1	0	1.15E-02	8.78E-04	4.43E-11	1.00	1.00
7	1	0	3.76E-06	1.84E-07	4.41E-11	0.97	1.00
8	0.5	1	6.07E-13	2.12E-14	4.41E-11	0.97	1.00

Table 5.3: Rotated unitsquare

Iterative linear solver

Before we continue we have to investigate the runtime of our solver depending on the level.

Before we discuss the results from Table 5.4 in details we need to clarify the content of some columns:

- The total runtime is the time it took to smoothen the mesh. This includes solving the equation with H_k and the representative of the antigradient, both using Umfpack.
- Init. Umfpack is the time it took to initialise all Umfpack-solvers for the H_k-equation. This is the part where „the heavy work “ is done and the LU-decomposition is calculated

Level	total runtime	Init. Umfpack	runtime Umfpack	total Umfpack	Newton-Iterations	normalized runtime Umfpack	ratio
4	0.43	0.12	0.01	0.13	7	0.02	
5	1.86	0.70	0.05	0.75	7	0.11	5.96
6	10.10	4.24	0.28	4.52	7	0.65	6.02
7	75.72	38.00	1.52	39.52	8	4.94	7.65
8	755.06	447.01	7.04	454.05	8	56.76	11.49
9	14797.54	9446.64	53.10	9499.73	12	791.64	13.95

Table 5.4: Runtime with Umfpack

- runtime Umfpack is the time it took to apply this LU-decomposition
- total Umfpack the sum of Init. Umfpack and runtime Umfpack
- Newton-Iterations is the Number of Newton-Iterations that were necessary to smoothen the mesh
- normalized runtime Umfpack is $\frac{\text{total Umfpack}}{\text{Newton-Iterations}}$
- ratio is the ratio how the total runtime increases per level

All times are given in seconds. Without going too much into details we can directly observe that the runtime becomes really long, and in the end it increases roughly with a factor of 14. When calculating the numerical cost by hand we can estimate that it is $\mathcal{O}\left(n^2\right)$ because the matrix is a banded matrix. The reason we did not calculate it for level 10 is because it did not finish in time, but we can estimate that it would take $12656s \approx 3.5h$ per Newton-Iteration just for the H_k-equation plus additional time for the calculation of the Riesz-representative of the Antigradient. We can clearly see that this duration is not acceptable. Therefore, let us now repeat the previous test and replace Umfpack with a PCG(Jacobi)-solver ([MV15, Section 5.8]):

Level	total runtime	Init. PCG	runtime PCG	total PCG	Newton-Iterations	normalized runtime PCG	ratio
4	0.24	0.00	0.02	0.02	7	0.00	
5	0.75	0.00	0.12	0.12	7	0.02	5.85
6	3.45	0.00	0.90	0.90	7	0.13	7.72
7	26.95	0.02	11.01	11.02	8	1.38	10.68
8	188.38	0.07	93.69	93.76	8	11.72	8.51
9	3833.50	0.45	2598.96	2599.41	13	199.95	17.06
10	34454.64	2.33	24456.46	24458.78	15	1630.59	8.15

Table 5.5: Runtime with PCG

In Table 5.5 we see a huge improvement: We were able to calculate on level 10. We also see that PCG works different than Umfpack: For PCG the initialization is mostly the creation of some vectors and most of the time is spent in the actual „solve-routine“ which is the opposite of Umfpack. We also see a jump in the ratio at level 9, but we assume that this is because until then a big portion of the data would fit into the CPU-cache. Let us now therefore redo the test from Table 5.1 and replace every linear solver with a PCG. The results of this are shown in Table 5.6.

k	σ_k	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter s	defect s
1	1	1.16E+01	2.11E+00	2.18E-01	0.77	0.30	1	7.40E-04	23	1.04E-03
2	1	3.12E+00	4.50E-01	1.74E-02	0.96	0.27	1	1.81E-04	23	2.90E-04
3	1	7.85E-01	1.40E-01	2.29E-03	1.00	0.38	1	3.92E-05	23	7.53E-05
4	1	3.17E-01	3.94E-02	1.48E-04	1.00	0.55	1	1.86E-05	24	1.67E-05
5	1	8.35E-02	8.06E-03	2.64E-06	1.00	0.74	1	4.46E-06	24	4.50E-06
6	1	1.15E-02	8.77E-04	2.30E-13	1.00	1.00	1	7.74E-07	24	6.25E-07
7	1	3.72E-06	2.20E-07	1.18E-16	0.97	1.00	1	1.72E-10	23	2.32E-10
8	1	1.72E-10	1.69E-11	1.35E-17	0.96	1.00	1	8.40E-15	24	9.93E-15

Table 5.6: Validation for the first testcase with PCG

We do not show figures of the mesh because they would look identical to Figure 5.4. What stands out most is that one PCG-Iteration to solve the equation for δ is actually enough to achieve a somehow similar result. While the residuals in Table 5.1 are smaller, this is somehow expected and within tolerances for practical usecases. If one would need a better solution, we can estimate that probably one iteration for δ would be good again. We did not explicitly set the solver to 1 iteration, but rather we set it to stop if it either gained 4 digits or the residual is better then 10^{-3} times the Newton-defect. These numerical tests show that we need a fast linear solver or else the algorithm will be impractical. Our test then showed that a PCG-solver speeds up the process significantly, especially since it was not necessary to do many PCG-steps. This might change for other geometries, but for the moment this is sufficient. Overall, what we can estimate now is that solving the equation for δ is not going to be „too expensive“ which means we might not need to implement a multigrid-solver for this. On the other hand, the equation for s might become more expensive to solve, but this is - at least in our case - solving a Poisson-equation. We are leaving this open to discussions because there are better experts at solving a Poisson-equation.

5.3. Effects μ and λ

Let us now repeat the previous test, but instead of refining it 3 times we refine it 6 times. We then test different quantities of μ and λ. We also then just sum up the total number of Armijo steps and total number of iterations for δ.

μ	λ	Newton-Iterations	Iterations δ	Armijo-Steps	δ rejected
0.01	1	22	22	6	19
0.1	1	13	13	8	6
1	1	6	6	5	1
10	1	7	7	3	0
100	1	16	16	1	0

Table 5.7: Influence of μ and λ

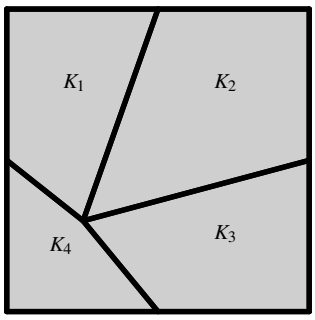


Figure 5.6: Example that is not covered from the theory

In Table 5.7 we see two different things: First of all we remind ourselves that our analysis covers only cases where $\det(\nabla\gamma) < \exp\left(\frac{\mu}{\lambda}+1\right)$, so clearly we could not expect any good results for $\mu = 0.01$ or $\mu = 0.1$. It does not necessarily have to diverge because imagine a situation as presented in Figure 5.6: For K_1, K_2 and K_3 the cells shrink, so the determinant is smaller than 1. Therefore, for these cells we have coverage from our analysis. However, for K_4 this might not be the case, but because they share vertices this still might work well even if our choice of μ and λ would not allow us to draw this conclusion.
We also notice that while $\mu = 100$ is well covered and within the theory the general optimization problem seems to be „harder“ to solve here because we need more Newton-Iterations. Therefore, we conclude from this test that we should use the smallest pair of μ and λ that fulfills the condition $\det(\nabla\gamma) < \exp\left(\frac{\mu}{\lambda}+1\right)$ for all cells. Since we do not know this a priori and we would like to have a small safety-margin we choose $\mu = 10$ and $\lambda = 1$. The reason is that we gained only a little bit by choosing $\mu = 1$ which would allow $\det(\nabla\gamma) < 7.389$, and a volume-change of 7 is no complete unreasonable assumption.

5.4. Finer mesh

We now refine the mesh up to level 9 and repeat the test. We get the iteration number from Table 5.8.

We can clearly see that in the beginning, it seems as if the number of Newton-iterations seems to be independent of the level, but at level 9 this rule does not apply any longer. We also observe that the number of Armijo-Steps increase. Therefore, we check the details of level 9 in Table 5.9

Level	Newton-Iterations	Armijo-Steps	Iterations δ	δ rejected	Iterations s
3	8	0	8	0	188
4	7	0	7	0	327
5	7	0	7	0	645
6	7	3	7	0	1265
7	8	3	8	0	2634
8	8	9	8	1	4841
9	13	33	13	1	17218

Table 5.8: Iterations on various levels

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	6	1.96E+00	1.35E+02	2.57E+00	0.62	0.30	1	1.64E-04	1466	FALSE
2	11	1.71E+00	2.16E+03	2.49E+00	0.49	0.44	1	1.49E-04	1473	TRUE
3	4	1.28E+00	1.22E+02	2.17E+00	0.65	0.47	1	9.81E-05	1479	FALSE
4	4	1.05E+00	1.09E+02	1.90E+00	0.78	0.55	1	8.47E-05	1481	FALSE
5	0	9.21E-01	1.00E+02	1.44E-01	0.81	0.60	1	7.41E-05	1482	FALSE
6	2	4.94E-01	1.11E+01	8.19E-02	0.91	0.65	1	3.79E-05	1476	FALSE
7	5	3.08E-01	8.15E+00	7.71E-02	0.87	0.83	1	2.38E-05	1488	FALSE
8	1	2.77E-01	7.67E+00	2.25E-02	0.92	1.00	1	2.02E-05	1493	FALSE
9	0	1.51E-01	4.19E+00	1.98E-04	0.94	1.00	1	1.07E-05	1490	FALSE
10	0	4.04E-02	2.31E-01	8.78E-07	0.94	1.00	1	2.85E-06	1302	FALSE
11	0	8.30E-03	3.69E-03	1.79E-09	0.95	1.00	1	5.86E-07	732	FALSE
12	0	3.78E-04	1.12E-04	1.33E-14	0.95	1.00	1	2.66E-08	788	FALSE
13	0	9.10E-07	2.04E-06	1.22E-16	0.95	1.00	1	6.44E-11	1068	FALSE

Table 5.9: Iterations on levels 9

Summing this up, the total computational cost adds up to

- Assembling and solving 13 Poisson-equations
- Assembling H_k 13 times
- 13 PCG-Steps to solve the H_k-equation
- 33 Armijo-Steps

We observe that most Armijo-Steps were necessary during the first few Newton-Iterations, and one time we obtained a direction δ that got discarded anyways. This is good and bad in itself: It is good because we see that both the Armijo-scheme and the check of the angle-condition are important to keep this method as robust as possible, but at the same time it is bad because we put a lot of computational work in assembling and solving the linear system for δ and evaluating the Armijo-condition, so we do want to try to avoid to do this very often. This test shows us that we should work on reducing the computational

cost in the first few iterations. Therefore, we now investigate what happens if we do not use δ in the first steps but directly switch to use s in the first 2-4 steps.

5.5. Initial antigradient steps

We are now repeating the previous test with one small modification: During the first two Newton-Iterations we do not solve for δ and directly choose s as searchdirection. The result can be found in Table 5.10

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	11	1.96E+00	2.22E+03	2.58E+00	1.00	0.30	-	-	1466	TRUE
2	7	1.32E+00	2.16E+03	1.69E+00	1.00	0.30	-	-	1479	TRUE
3	0	4.55E-01	1.16E+02	8.07E-02	0.97	0.36	1	3.27E-05	1446	FALSE
4	0	1.49E-01	2.24E+01	4.22E-03	0.99	0.50	1	1.06E-05	1466	FALSE
5	0	4.09E-02	4.33E+00	9.85E-05	0.99	0.69	1	2.88E-06	1481	FALSE
6	0	6.77E-03	5.00E-01	4.13E-09	0.99	0.98	1	4.77E-07	1482	FALSE
7	0	5.13E-05	2.77E-03	1.43E-16	0.99	1.00	1	3.61E-09	1484	FALSE
8	0	3.92E-09	1.57E-07	5.30E-17	0.95	1.00	1	2.75E-13	1546	FALSE

Table 5.10: Iterations on levels 9 - two initial antigradient steps

In total the computational cost for this modification is

- Assembling and solving 8 Poisson-equations
- Assembling H_k 6 times
- 6 PCG-Steps
- 18 Armijo-Steps

Comparing this to Table 5.9 we notice that we saved

- Assembling and solving the Poisson-equation 5 times
- Assembling H_k 7 times
- 7 PCG-Steps to solve the H_k-equation
- 15 Armijo-Steps

and never rejected δ. This is clearly a huge improvement. If we push this idea and start with three antigradient steps we obtain the results from Table 5.11.

so in total we have

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	11	1.96E+00	2.22E+03	2.58E+00	1.00	0.30	-	-	1466	TRUE
2	7	1.32E+00	2.16E+03	1.69E+00	1.00	0.30	-	-	1479	TRUE
3	5	4.55E-01	1.65E+03	3.78E-01	1.00	0.30	-	-	1446	TRUE
4	0	1.79E-01	5.69E+01	1.69E-02	0.99	0.42	1	1.28E-05	1487	FALSE
5	0	4.94E-02	1.10E+01	6.36E-04	0.99	0.56	1	3.50E-06	1515	FALSE
6	0	1.29E-02	1.72E+00	5.19E-06	0.99	0.80	1	9.13E-07	1514	FALSE
7	0	1.31E-03	1.25E-01	9.92E-13	0.99	1.00	1	9.29E-08	1510	FALSE
8	0	8.62E-07	3.62E-05	1.17E-16	0.96	1.00	1	6.08E-11	1516	FALSE

Table 5.11: Iterations on levels 9 - three initial antigradient steps

- Assembling and solving 8 Poisson-equations
- Assembling H_k 5 times
- 5 PCG-Steps
- 23 Armijo-Steps

This is almost comparable to two initial antigradient steps: 1 assembly of H_k and 5 PCG-Steps vs. 5 Armijo-Steps, but the Newton-defect is slightly worse. One might get the idea that this continues, but with 4 initial antigradient steps we get the results from Table 5.12

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	11	1.96E+00	2.22E+03	2.58E+00	1.00	0.30	-	-	1466	TRUE
2	7	1.32E+00	2.16E+03	1.69E+00	1.00	0.30	-	-	1479	TRUE
3	5	4.55E-01	1.65E+03	3.78E-01	1.00	0.30	-	-	1446	TRUE
4	3	1.79E-01	7.04E+02	2.96E-01	1.00	0.30	-	-	1487	TRUE
5	0	2.05E-01	4.98E+01	1.97E-02	0.99	0.41	1	1.46E-05	1507	FALSE
6	0	7.68E-02	1.18E+01	2.32E-03	1.00	0.37	1	5.41E-06	1513	FALSE
7	0	2.99E-02	3.24E+00	1.71E-04	1.00	0.51	1	2.11E-06	1511	FALSE
8	0	9.21E-03	6.83E-01	4.43E-06	1.00	0.69	1	6.46E-07	1518	FALSE
9	0	1.57E-03	8.50E-02	8.50E-10	1.00	0.97	1	1.10E-07	1518	FALSE
10	0	2.29E-05	1.03E-03	1.06E-16	1.00	1.00	1	1.61E-09	1526	FALSE
11	0	1.64E-09	7.26E-08	4.16E-17	0.94	1.00	1	1.16E-13	1550	FALSE

Table 5.12: Iterations on levels 9 - four initial antigradient steps

We notice one important thing here: Even though is now getting more expensive it is still cheaper than no initial antigradient step at all:

- Assembling and solving 11 Poisson-equations
- Assembling H_k 7 times

- 7 PCG-Steps
- 26 Armijo-Steps

We also see that one time, the heuristic for increasing α had to decrease α again, but this just worked and did not cause any other problems. These experiments show us that using the antigradient for the first few steps might save some computational cost and improve convergence. If we do too many initial antigradient steps we eventually might end up with a gradient method, so we should not push this too far, but it is a viable option for the first 2-4 steps.

5.6. Different testcases in 2D

So far we only did tests on a unitsquare with a really extreme deformation. Now we want to test this on other meshes as well.

Unitsquare - randomly distorted

We start with a unitsquare that is randomly distorted as in figure 5.7: We still know what

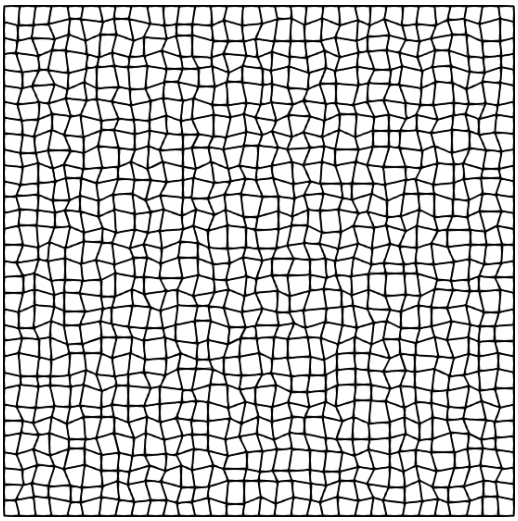


Figure 5.7: Randomly distorted unitsquare

the outcome should look like, but it might be a bit harder to solve because the deformation is not unidirectional.

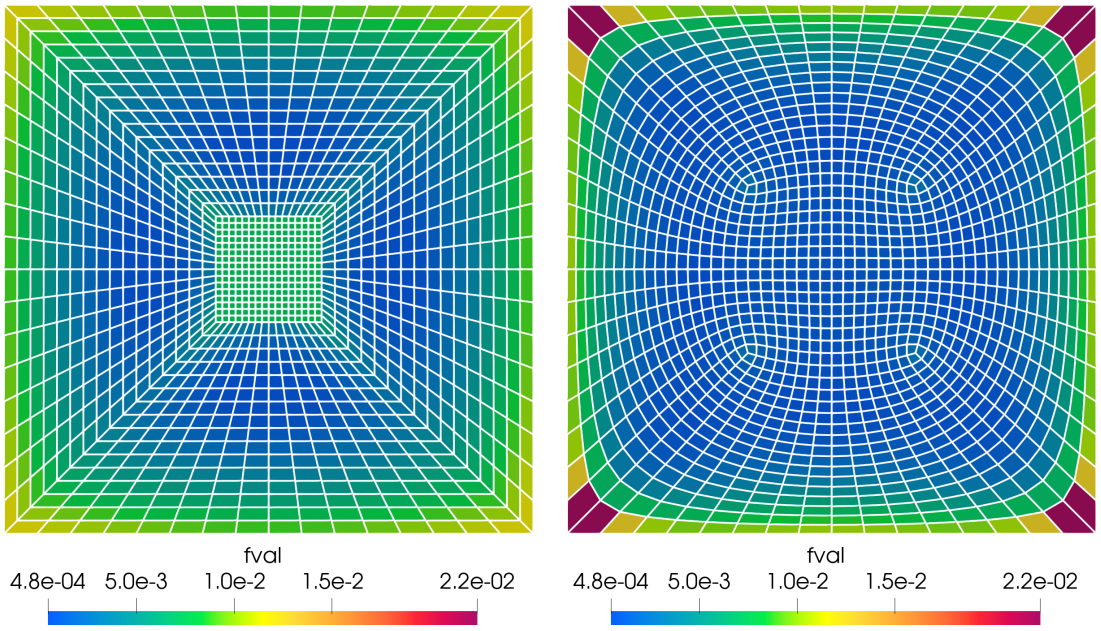
In Table 5.13 we see that also in this case it works out nicely. What we did not show is the final result because this would just be a perfect regularly refined unitsquare.

Transition element

To refine a mesh locally from one quadrilateral to five quadrilaterals one can use a so called „transition element“. We now test the behavior of this algorithm when we try to

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	4	1.07E+01	3.85E+00	1.06E-01	1.00	0.30	-	-	36	TRUE
2	4	3.29E+00	1.30E+00	8.29E-03	1.00	0.30	-	-	39	TRUE
3	0	9.06E-01	2.66E-02	7.01E-04	0.98	0.42	1	6.51E-05	38	FALSE
4	0	2.92E-01	6.68E-03	3.75E-05	1.00	0.60	1	1.75E-05	38	FALSE
5	0	6.77E-02	1.36E-03	2.61E-07	1.00	0.84	1	4.02E-06	38	FALSE
6	0	5.66E-03	1.05E-04	2.29E-14	1.00	1.00	1	3.81E-07	30	FALSE
7	0	4.89E-07	2.53E-07	1.20E-16	0.97	1.00	1	3.24E-11	48	FALSE
8	0	3.24E-11	1.06E-11	1.46E-17	0.96	1.00	1	2.05E-15	49	FALSE

Table 5.13: Iterations randomly distorted unitsquare



(a) Initial mesh (b) Final mesh

Figure 5.8: Smoothing the transition element

smoothen it. In Figure 5.8 we evaluated the functional for each cell and colored the cell according to this value. The lower the value is, the better this cell is considered. Visually we can see in Figure 5.8 that the result looks good, only the four corners become worse, but if we see this element as part of a bigger mesh then these corners could move as well and it might end up better. We also notice that the elements in the center are much smaller than those close to the boundary. This is reflected by the functional value that increases when close to the boundary. In Table 5.14 we see a relatively normal convergence behavior as we expected from our previous experiments, but now we see one major difference: In the last iteration we had to do two Armijo-Steps again. This is now new to us, earlier they were necessary only during the first iterations and towards the end no damping was required. While there is no proof for this theory, it seems plausible to us that the main reason is that the four elements in the middle (which would be at the corners of the original inner quadrilateral, compare Figure 5.8) can easily degenerate and this is

the reason for smaller stepsizes. This test shows us that there are limits to smoothening a mesh, sometimes the connectivity or the geometry do not allow to create cells that have the same size.

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	4	6.27E+00	6.10E+01	5.2130	1.00	0.30	-	-	37	TRUE
2	4	5.00E+00	5.44E+01	4.6063	1.00	0.30	-	-	37	TRUE
3	0	5.92E+00	1.83E+00	3.4803	0.94	0.34	1	3.98E-04	37	FALSE
4	0	3.89E+00	9.09E-01	3.2009	0.95	0.30	1	2.52E-04	36	FALSE
5	0	1.11E+00	5.88E-01	3.1241	1.00	0.36	1	6.30E-05	38	FALSE
6	0	7.36E-01	3.09E-01	3.0803	1.00	0.51	1	3.20E-05	37	FALSE
7	0	2.66E-01	1.16E-01	3.0737	1.00	0.62	1	1.53E-05	38	FALSE
8	0	8.37E-02	2.89E-02	3.0730	1.00	0.85	1	5.31E-06	38	FALSE
9	0	8.09E-03	2.60E-03	3.0730	1.00	1.00	1	5.19E-07	38	FALSE
10	0	6.21E-06	1.20E-06	3.0730	0.95	1.00	1	3.37E-10	38	FALSE
11	2	3.36E-10	4.96E-11	3.0730	0.90	1.00	1	2.64E-14	54	FALSE

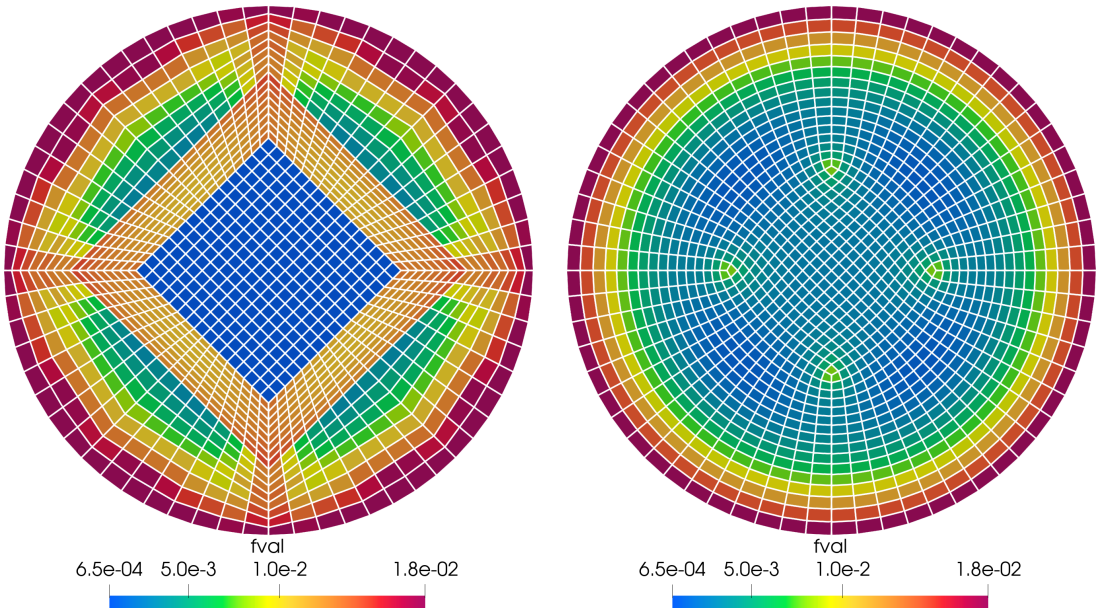
Table 5.14: Iterations transition element

This example also shows that we really should not „deactivate“ the Armijo-Scheme after a few iterations. We also see that (because the final mesh cannot contain just optimal cells) we can have a final value of $F(\gamma)$ that is significantly greater than zero.

Unitcircle

Now let us try to smoothen a unitcircle: We directly can expect the outermost layer of elements to have a bigger size than we prescribe them, and the inner ones have to compensate for that and thus have to become smaller. The reason for this is that the number of elements directly prescribes the „boundary-edge“, and then one layer to the interior the same number of elements is used for a smaller diameter. The initial and final mesh are shown in Figure 5.9. First of all we notice that the quality becomes much more uniform, the only cells in the interior that are „not so good“ are those that stem from the corners of the initial quadrilateral. We also notice that the outer cells are mostly flagged as bad is because they became too large, the shape of them looks quite reasonable. Therefore, this is as predicted. Let us also look into the details of the iterations in Table 5.15.

We can observe the same behavior as for the transition element, but the number of Armijo-Steps increases even more. When looking at Table 5.15 and from the previous test Table 5.14 and looking at the norm of the update δ we start wondering if the last iterations are actually necessary. Therefore, let us now show the iterations visually. In Figure 5.10 $k = 0$ would be the initial mesh. Visually, we can barely see a difference between $k = 4$ and the final mesh. This is a really important result because it allows us to reduce the computational cost even more - at least for practical applications. We could now try to



(a) Initial mesh (b) Final mesh

Figure 5.9: Smoothing a unitcircle

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	4	1.13E+01	4.43E+01	9.8954	1.00	0.30	-	-	37	TRUE
2	4	1.12E+01	3.53E+01	9.1382	1.00	0.30	-	-	37	TRUE
3	1	9.95E+00	1.76E+00	7.0451	0.96	0.30	1	5.36E-04	37	FALSE
4	0	2.17E+00	3.18E-01	6.9882	0.98	0.32	1	1.21E-04	36	FALSE
5	0	1.63E+00	1.87E-01	6.9567	0.99	0.47	1	8.64E-05	37	FALSE
6	0	6.30E-01	8.18E-02	6.9504	1.00	0.54	1	3.49E-05	37	FALSE
7	0	2.68E-01	2.72E-02	6.9493	1.00	0.73	1	1.64E-05	36	FALSE
8	0	5.12E-02	4.45E-03	6.9492	0.99	0.98	1	3.42E-06	36	FALSE
9	0	7.02E-04	5.14E-05	6.9492	0.99	1.00	1	4.07E-08	36	FALSE
10	19	4.78E-08	6.86E-09	6.9492	0.91	1.00	1	2.78E-12	48	FALSE

Table 5.15: Iterations unitcircle

work out a stopping criterion that would stop the algorithm as soon as nothing „visually“ changes, but from our experience this leads to other problems which we will discuss later. We just should keep this in mind for Chapter 6. We also have just shown that it is possible to smoothen a mesh that does have a different shape than the optimal cell.

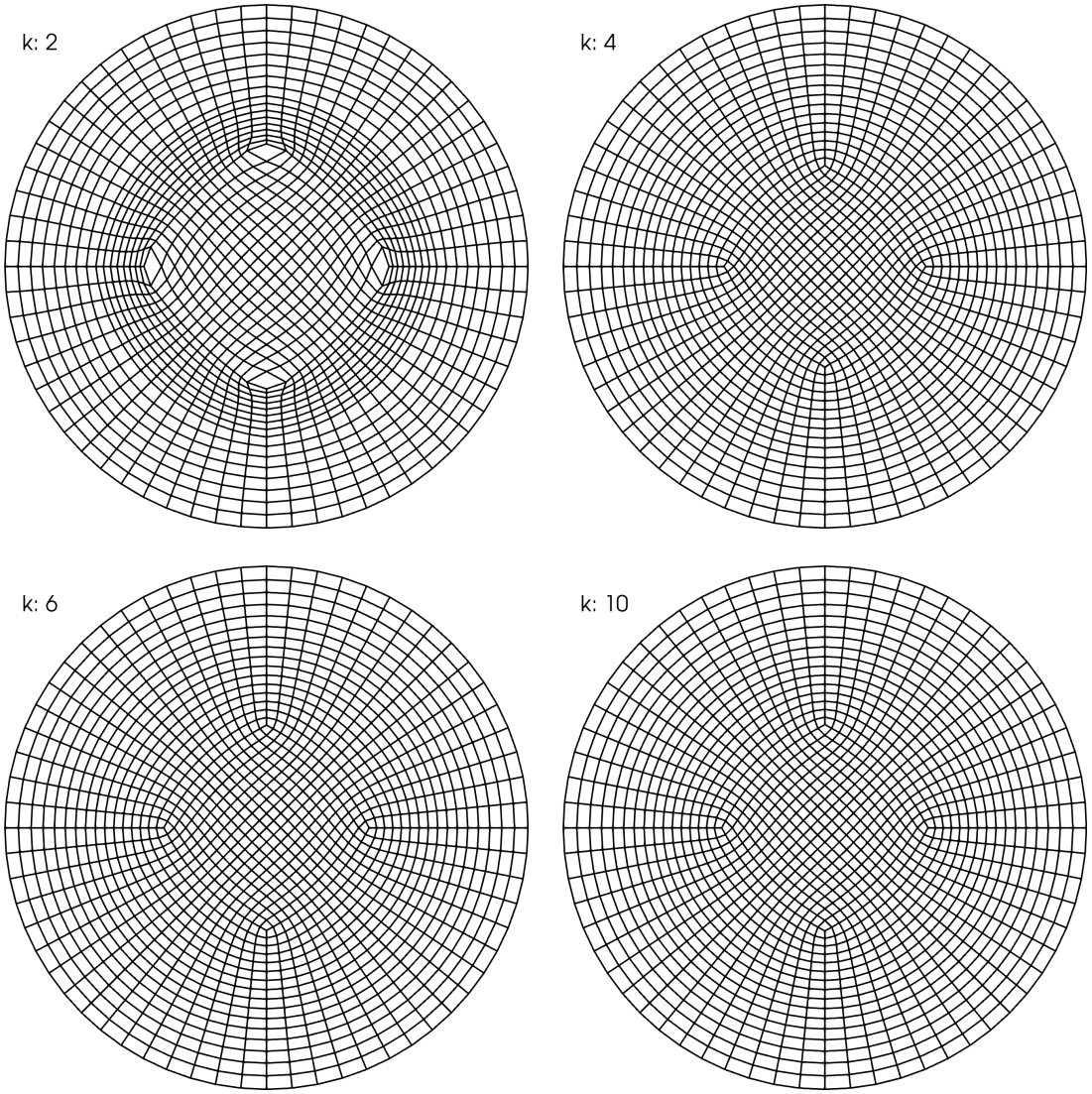
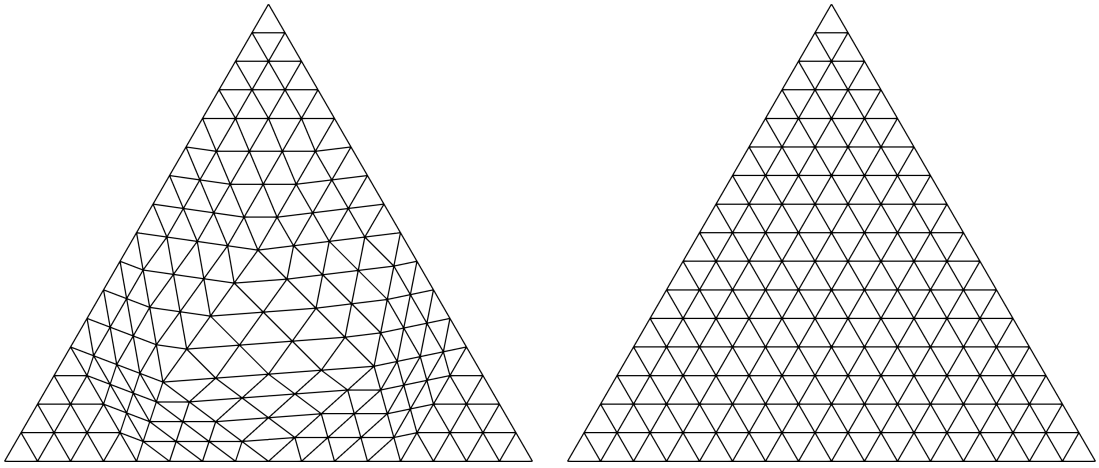


Figure 5.10: Smoothing a unitcircle - mesh per iteration

5.7. Triangle meshes

So far we only showed results for quadrilaterals because they are the focus of our work. Nevertheless our analysis showed that it should also work for triangles. Therefore, we construct a testcase similar to the one for quadrilaterals: We take a mesh that in itself is an equilateral triangle, refine it and after the refinement we move some of the inner vertices so that the initial mesh looks like in Figure 5.11a.

The final mesh (Figure 5.11b) suggests that that the code works for triangles as well, whereas the results in Table 5.16 seem to be a bit undecided because our functional value is not as close to 0 as it was for the quadrilateral meshes. Looking more into details, we see that the functional value on each cell is $7.6 \cdot 10^{-4}$. This might be because either the Newton-Iteration stopped too early or because our initial mesh was not an equilateral



(a) Initial mesh (b) Final mesh

Figure 5.11: Validation for triangles

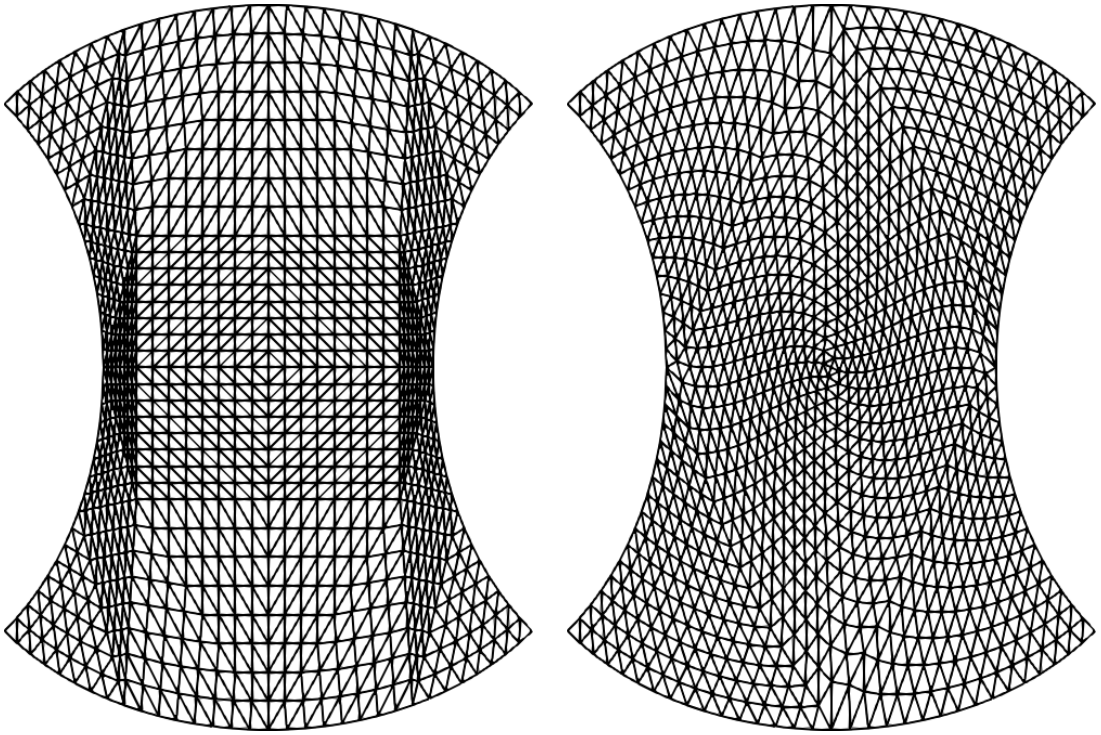
k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	0	4.35E+00	4.84E-01	2.48E-01	0.98	0.30	1	2.28E-04	22	FALSE
2	0	1.63E+00	1.32E-01	2.00E-01	0.99	0.40	1	6.81E-05	21	FALSE
3	0	5.87E-01	4.24E-02	1.94E-01	1.00	0.55	1	4.28E-05	21	FALSE
4	0	1.66E-01	1.04E-02	1.93E-01	1.00	0.75	1	1.01E-05	21	FALSE
5	0	2.27E-02	1.21E-03	1.93E-01	1.00	1.00	1	1.30E-06	21	FALSE
6	0	1.95E-05	9.46E-07	1.93E-01	0.96	1.00	1	8.42E-10	21	FALSE
7	34	8.42E-10	1.16E-10	1.93E-01	0.96	1.00	1	3.84E-14	21	FALSE

Table 5.16: Iterations validation triangle

triangle: Because of the coordinates of the top corner, all y-coordinates are fractions of $\sqrt{3}$. In the process of creating the mesh, we had to perform a few export- and import-operations where we certainly have lost some digits. Therefore, all coordinates on the boundary (except those at $y = 0$) are off a little bit, and our boundary-conditions fixed their positions. This leads to each cell being „slightly non-optimal“ which causes a value of $7.6 \cdot 10^{-4}$ on it. Keeping this in mind and recalling (3.1.1) (where we state that our assumed form is to sum up the quality of the cells) we consider this result close enough for passing the test. This means that our algorithm works for triangles as well.

Nontrivial triangle mesh

Now we show a mesh that is now a bit of a challenge for triangles because of the way the refinement works: Initially, we have just a set of cells and a very rough polygonal approximation of the correct boundary. When we refine the mesh, we also want to have a better approximation of the boundary. Therefore, after each step, the parametrization of the boundary is applied to move the newly created boundary nodes to the real boundary. When following this algorithm, something like in Figure 5.12a can happen, but as we see in Figure 5.12b our algorithm manages to smoothen it.



(a) Initial mesh (b) Final mesh

Figure 5.12: Smoothing a triangle mesh

Let us look at the iterations in Table 5.17.

From the sheer number of iterations we already see that this was one of the harder meshes that we tested, especially in the beginning the residual does not drop as fast as it did on other meshes. However, at some point we saw the fast convergence of the Newton-Scheme that we can expect. We also see that even though the initial mesh was mirror-symmetric, the final mesh is not, it seems to be point-symmetric. This seems to be counter-intuitive, but in the analysis we never showed the uniqueness of the solution, only the existence (compare Theorem 3.4, 3.7 and3.9). Here we might have a case where several local minimums exist because it seems that if we would mirror the deformations from the left half and apply this version to the right half of the initial mesh (and for the other half accordingly) we might have the same functional value. This could be a reason why it was one of the harder meshes. This test has therefore shown us that the algorithm works and creates a mesh of good quality even if we have no unique solution to the problem.

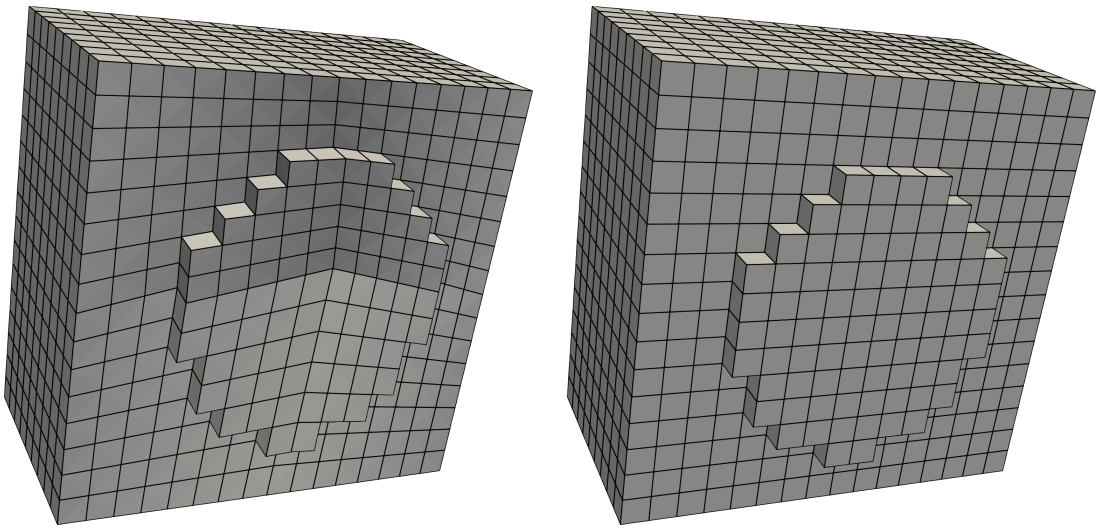
k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	5	1.11E+01	5.47E+01	3.9848	1.00	0.30	-	-	46	TRUE
2	8	2.38E+01	2.42E+01	3.7645	1.00	0.30	-	-	43	TRUE
3	0	8.61E+00	1.07E+00	3.4523	0.80	0.13	1	6.28E-04	44	FALSE
4	0	8.07E+00	7.35E-01	3.2915	0.85	0.18	1	5.70E-04	46	FALSE
5	0	7.95E+00	6.76E-01	3.2838	0.78	0.19	1	5.65E-04	44	FALSE
6	0	4.32E+00	6.68E-01	3.2737	0.97	0.19	1	2.89E-04	46	FALSE
7	0	5.65E+00	6.20E-01	3.2197	0.92	0.24	1	3.62E-04	44	FALSE
8	0	3.67E+00	5.78E-01	3.2111	0.98	0.19	1	2.27E-04	47	FALSE
9	1	3.82E+00	5.59E-01	2.9995	0.97	0.24	1	2.34E-04	45	FALSE
10	0	3.43E-01	2.92E-02	2.9994	0.98	0.23	1	2.04E-05	43	FALSE
11	0	3.68E-01	2.19E-02	2.9992	0.99	0.35	1	2.30E-05	42	FALSE
12	0	3.20E-01	1.54E-02	2.9990	0.99	0.33	1	2.18E-05	42	FALSE
13	0	2.71E-01	1.10E-02	2.9989	0.99	0.36	1	1.72E-05	41	FALSE
14	0	2.34E-01	7.36E-03	2.9988	0.99	0.39	1	1.52E-05	41	FALSE
15	0	1.72E-01	4.56E-03	2.9988	0.99	0.43	1	1.21E-05	41	FALSE
16	0	1.23E-01	2.56E-03	2.9987	0.99	0.50	1	8.08E-06	40	FALSE
17	0	6.85E-02	1.17E-03	2.9987	0.99	0.58	1	4.60E-06	39	FALSE
18	0	2.98E-02	4.01E-04	2.9987	0.99	0.74	1	2.05E-06	39	FALSE
19	0	6.89E-03	7.09E-05	2.9987	0.99	0.99	1	4.46E-07	38	FALSE
20	0	7.21E-05	7.87E-07	2.9987	0.99	1.00	1	5.12E-09	38	FALSE
21	1	5.13E-09	1.78E-09	2.9987	0.96	1.00	1	3.31E-13	58	FALSE

Table 5.17: Iterations for this triangle mesh

5.8. 3D Testcase

When we presented all components and the reference cells we also showed them for three dimensions. Therefore, we briefly want to show that this is also possible. We take the same settings as in previous tests, but we will not do any initial antigradient steps. As testcase, we create something similar to the first tests we did here: We take $[0,2]^3$, refine it once, move the midpoint to $(1.2, 1.2, 1.2)$ and then continue to refine it (see Figure 5.13a). We are going to just focus on this one testcase with hypercubes even though many other would be possible as well. We will also just show the tables for this case: For level 3 the details are shown in Table 5.18, and a summary for levels 3-6 is shown in Table 5.19. We see a very similar behavior to the equivalent 2d case.

This test shows us that we can also smoothen 3D meshes using this algorithm. It also suggests that solving for δ is not significantly harder than in 2D.



(a) Initial mesh (b) Final mesh

Figure 5.13: Selected cells of the 3D-mesh

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	0	1.77E+00	5.14E+00	1.56E-01	0.99	0.30	1	8.35E-05	21	FALSE
2	0	6.31E-01	1.10E+00	1.40E-02	0.99	0.44	1	2.87E-05	20	FALSE
3	0	2.19E-01	2.52E-01	6.40E-04	1.00	0.61	1	1.05E-05	21	FALSE
4	0	5.02E-02	4.25E-02	4.08E-06	1.00	0.84	1	2.63E-06	21	FALSE
5	0	4.12E-03	2.93E-03	5.66E-14	1.00	1.00	1	1.75E-07	21	FALSE
6	0	5.98E-07	2.72E-07	4.51E-15	0.95	1.00	1	3.30E-11	20	FALSE

Table 5.18: Iterations for 3D hypercube on level 3

Level	Newton-Iterations	Armijo-Steps	Iterations δ	δ rejected	Iterations s
3	6	0	6	0	124
4	6	0	6	0	249
5	6	0	6	0	496
6	6	0	6	0	994

Table 5.19: Iterations on various levels

6

Numerical results II - r-Adaptivity

So far, all we did was smoothing a mesh that came out of a meshgenerator. As we mentioned in Chapter 3, we smoothened the mesh to reduce the error $||u - u_h||$. This itself is already a powerful tool, but there are other applications as well. One other application is when we do time-dependent simulations and the geometry changes over time, for example a rising bubble. Another application would be that the geometry is complicated to mesh so that a fictitious boundary method is used: These methods become really powerful tools if the approximation error of the boundary-approximation becomes small. For both applications it can be an advantage to move the points inside a mesh so that an (internal) object is resolved better. How does this now translate to our method? Actually, rezoning - or r-adaptivity - is nothing else but choosing different scales for the reference cell. If we assume that there is a function $v(x)$ then we have basically two options:

i) We reformulate the model so that the cell sizes depend on the final position. This would mean that $\gamma\restriction_K$ would be something along the lines of

$$\gamma\restriction_K = \gamma(x, v(x^*))$$

This is possible and was examined in [Pau17], even though this introduces new terms to the derivatives.

ii) We can do something like a Picard-Iteration: Initially, we set the cell sizes to $v(\gamma_0)$. We then solve for this given cell sizes and call the result $\gamma_{\text{final},i}$. Then we set $\gamma_{0,i+1} = \gamma_{\text{final},i}$, update the desired cell sizes to $v(\gamma_{0,i+1})$ and repeat the process for i until it converges.
This was also investigated in [Pau17], but there the results were not satisfying.

Even though the results in [Pau17] were not satisfying we are going to investigate the second approach. There are three reasons for this: First, it does not require to calculate new derivatives and the implementation of new terms which mean we can use the results from Chapter 5 as a validation for our code. Second, we minimize a different functional than [Pau17], so it is generally difficult to draw conclusions from this. Third, we also use

a different solver to solve the minimization problem (a Quasi-Newton scheme, whereas [Pau17] mainly relied on a NLCG). The second and third reason mean that we have a completely different problem here.
Before we start to test this method we need to give some thought into the terms „solve“ and „converges“. From a pure mathematical point of view, we could very well do the Newton-Iteration until the Newton-defect is smaller than 10^{-12} before we update the cell sizes, but to define the convergence for the Picard-Iteration in a way that does not conflict with the convergence of the Newton-Iteration becomes a real challenge: If we rely on just $F(\gamma)$, $||F'(\gamma)||$ or $||\delta||$ we could very well just monitor that our Newton-Scheme converged but gained no information if the mesh is now adapted to the intended situation.
After our tests in Chapter 5, especially as presented in Table 5.15 and Figure 5.10, we think it is not necessary to solve the Newton-Iteration to a very small residual before updating the desired cell volumes. Since we learned that during each iteration usually $||\delta||$ decreases, we are going to incorporate this in our criterion because it could be interpreted as „with these desired cell volumes we have now an almost optimal mesh“, so we do not spend time to „over-optimize“ it. Therefore, our strategy is going to be: Every time $||\delta||$ is below a certain threshold or every X steps we update the desired volumes. We stop the Picard-Iteration if our Newton-defect is below a certain threshold and we updated the desired volumes of the cells at least Y times. If the threshold for $||\delta||$ is not too small we found this to be a practical approach. However, we also must recall the Armijo-Condition 2.5.4

$$F(x_k) - F(x_k + \sigma_k \delta_k) \geq \beta \sigma_k (-\nabla F(x_k), \delta_k)$$

and one shortcut that came in handy when implementing: After the update of the volumes, we are technically solving a new problem. Therefore, we cannot use $F(x_k)$ as $F(x_k + \sigma_k \delta_k)$ from the previous step but also have to update this.
We also have to manage our expectations here: Smoothing a mesh so that every cell is optimal and adapting a mesh to a geometry are goals that are usually opposing each other: Just think of a 2d-quadrilateral mesh with a circle as interior boundary. The smooth optimal mesh would consist only of square cells, but the approximation of the circle would be quite bad.

6.1. Adapting a square to a circle

With these things out of the way let us now take $[-1,1]^2$ and adapt it to an interior circle with the center $(0,0)$ and the radius $r = 0.7$. All we do is set the desired cell volume to

$$v = 10^{-8} + \sqrt{\sqrt{c_x^2 + c_y^2} - 0.7}$$

where c_x and c_y are the x- and y-coordinates of the geometric center of the cell. This means that the cells should get smaller when they are closer to the circle. The 10^{-8} is just a small regularization parameter so that no cell has the volume zero. Note that because these cell sizes might not sum up the 4 (the volume of our whole domain) we treat these as relative cell sizes. If we apply the explained algorithm to a unitsquare on level 5 we get the results from Table 6.1. The final result is in Figure 6.1. The total cost

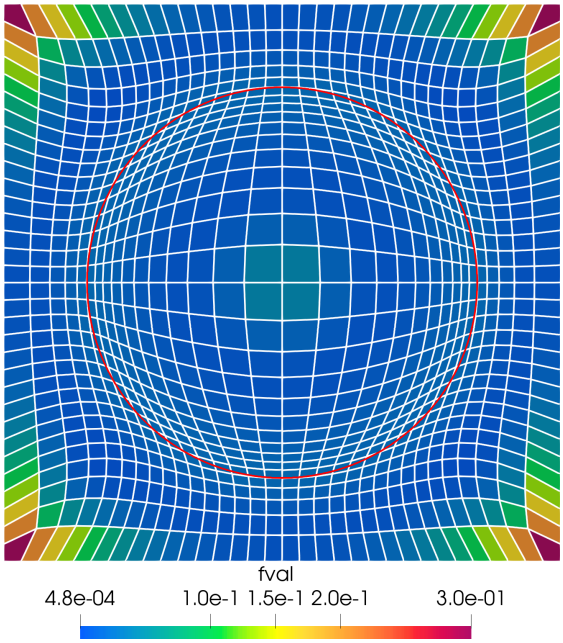
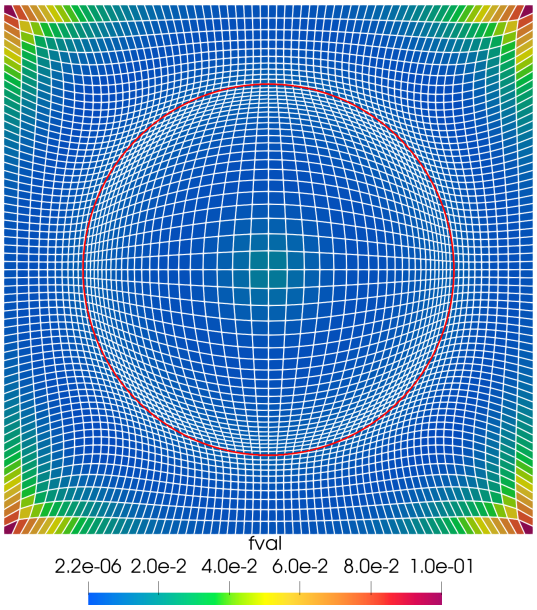


Figure 6.1: Adapting a square to a circle on level 5

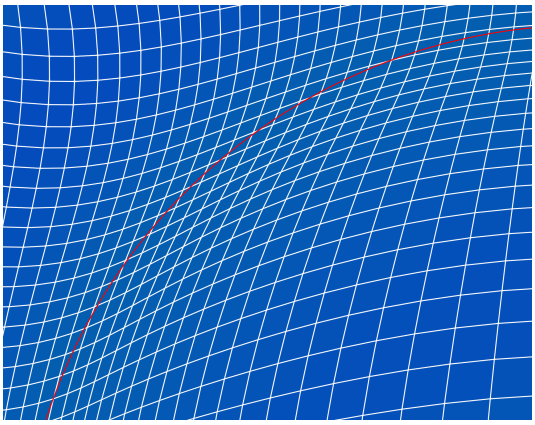
was 35 Newton-Iterations, 84 Armijo-Steps and 407 steps with a PCG-solver to solve the δ-equation. Even though we had to do 1044 PCG-Steps to solve the equation for s, this is a Poisson-equation and there are better ways to solve it, but this is not the focus of this work so we do not count these. The cell-volumes got updated after the Newton-Iterations 10,11,18,19,26,27 and 34. After the update, we treat this as a new problem and therefore start with 2 steps in the antigradient direction.

In Figure 6.1 again a lower value means that the cell is closer to the optimal cell. While in two instances we had to do many Armijo-Steps, overall this looks almost normal and as expected. The reason for the huge number of Armijo-Steps in the Newton-Steps 27 and 35 might be that a larger stepsize would have caused a cell to degenerate. We also notice here that even though we are optimizing the mesh an thus minimizing the functional, the overall value of the functional increases with each update of the desired cell-volumes, only to decrease from then on. This shows that our initial assumption of mesh-quality opposing adaptation to a geometry was right.
Geometrically, we do see a problem here: Because of our chosen boundary condition, the four cells in the corners of the mesh have only one vertex that can be moved, so these cells cannot adapt too much. Our assumption here is that if we would allow the vertices to slide on the boundary these cells would have a better quality. This also influences the quality of the mesh in the surrounding cells.
Together, these tests show us that our approach to r-adaptivity with just choosing different cell volumes is indeed valid and works. Also, the algorithmic idea with a Picard-Iteration and the stopping- and update-criterion are effective.
If we repeat this on level 6 the overall adaption to the circle becomes better (see Figure 6.2). Even though this is somehow expected (we have more elements that we can deform)

we still want to show this because we are not quite happy with the overall result on level 5, but it becomes difficult to see everything on the overview for level 6 (see Figure 6.2a).



(a) Adapting a square to a circle on level 6



(b) Adapting a square to a circle on level 6 - zoom

Figure 6.2: Adapting a square to a circle on level 6

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	4	9.03E+00	5.80E+01	16.15	1.00	0.30	0	-	30	TRUE
2	4	6.25E+00	2.07E+01	14.48	1.00	0.30	0	-	31	TRUE
3	0	3.57E+00	6.07E-01	14.08	0.99	0.36	21	4.47E-03	31	FALSE
4	0	2.09E+00	2.72E-01	13.96	0.99	0.45	26	3.42E-03	31	FALSE
5	0	9.82E-01	1.01E-01	13.94	0.99	0.56	25	1.86E-03	31	FALSE
6	0	3.73E-01	2.67E-02	13.94	0.99	0.74	24	7.69E-04	30	FALSE
7	0	7.31E-02	3.70E-03	13.94	0.98	1.00	19	3.51E-04	30	FALSE
8	0	3.46E-04	9.98E-05	13.94	0.95	1.00	11	5.49E-05	29	FALSE
9	0	5.49E-05	1.50E-05	13.94	0.96	1.00	24	3.03E-07	31	FALSE
10	0	3.03E-07	3.03E-08	13.94	0.94	1.00	6	4.51E-08	29	FALSE
11	0	4.51E-08	8.94E-08	23.90	1.00	1.00	0	-	30	TRUE
12	4	5.47E+00	2.00E+01	22.40	1.00	1.00	0	-	30	TRUE
13	4	3.57E+00	4.89E+00	22.24	1.00	1.00	0	-	30	TRUE
14	0	3.53E+00	1.13E-01	22.11	0.96	1.00	26	3.05E-03	29	FALSE
15	0	1.75E-01	7.05E-03	22.11	0.87	1.00	13	2.50E-03	29	FALSE
16	0	2.55E-03	5.29E-04	22.11	0.97	1.00	14	1.21E-04	31	FALSE
17	0	1.21E-04	2.13E-05	22.11	0.94	1.00	18	2.29E-06	29	FALSE
18	0	2.29E-06	3.37E-07	22.11	0.91	1.00	12	1.05E-07	30	FALSE
19	0	1.05E-07	3.79E-07	22.40	1.00	1.00	0	-	33	TRUE
20	4	2.36E+00	8.13E+00	22.10	1.00	1.00	0	-	31	TRUE
21	4	1.37E+00	2.13E+00	22.08	1.00	1.00	0	-	30	TRUE
22	0	1.69E+00	3.41E-02	22.06	0.96	1.00	19	1.20E-03	29	FALSE
23	0	4.50E-02	1.07E-03	22.06	0.85	0.86	8	1.49E-03	28	FALSE
24	0	5.30E-03	3.05E-04	22.06	0.80	1.00	18	4.15E-05	29	FALSE
25	0	4.14E-05	1.39E-05	22.06	0.96	1.00	15	3.55E-06	31	FALSE
26	0	3.55E-06	6.49E-07	22.06	0.93	1.00	23	3.70E-08	30	FALSE
27	34	3.70E-08	4.38E-08	21.93	1.00	1.00	0	-	29	TRUE
28	3	8.13E-01	2.38E+00	21.91	1.00	1.00	0	-	31	TRUE
29	4	1.05E+00	1.93E+00	21.88	1.00	1.00	0	-	31	TRUE
30	0	1.19E+00	3.50E-02	21.87	0.94	0.83	23	8.86E-04	27	FALSE
31	0	1.87E-01	3.00E-03	21.87	0.97	0.76	13	9.85E-04	27	FALSE
32	0	3.89E-02	5.15E-04	21.87	0.96	1.00	15	1.84E-04	26	FALSE
33	0	1.88E-04	4.26E-05	21.87	0.94	1.00	11	3.41E-05	31	FALSE
34	0	3.41E-05	9.18E-06	21.87	0.96	1.00	23	1.63E-07	29	FALSE
35	19	1.63E-07	3.11E-07	22.25	1.00	1.00	0	-	31	TRUE

Table 6.1: Iterations adapting a square to a circle

6.2. Adapting a square to a circle - small cells

In the previous section we showed that we can adapt a mesh to a circle by setting the relative cell-volumes to

$$v = 10^{-8} + \sqrt{\sqrt{c_x^2 + c_y^2} - 0.7}$$

which is effectively

$$v = 10^{-8} + d$$

where d is the distance to the surface. If one wants to get a finer resolution close to the surface one could get the idea to set the relative cell-volumes to

$$v = 10^{-8} + d^{\alpha}$$

with $\alpha > 1$. On the other hand we have a geometrical problem: If we do this on level 6, we only have 4225 vertices and 4096 cells, so the desired cell-volumes might require extreme cell deformations. This was one of the problems [Gra08] could observe: Under those circumstances, the cells began to deform in the way that they became very stretched rhombi with interior angles coming close to π like in Figure 6.3.

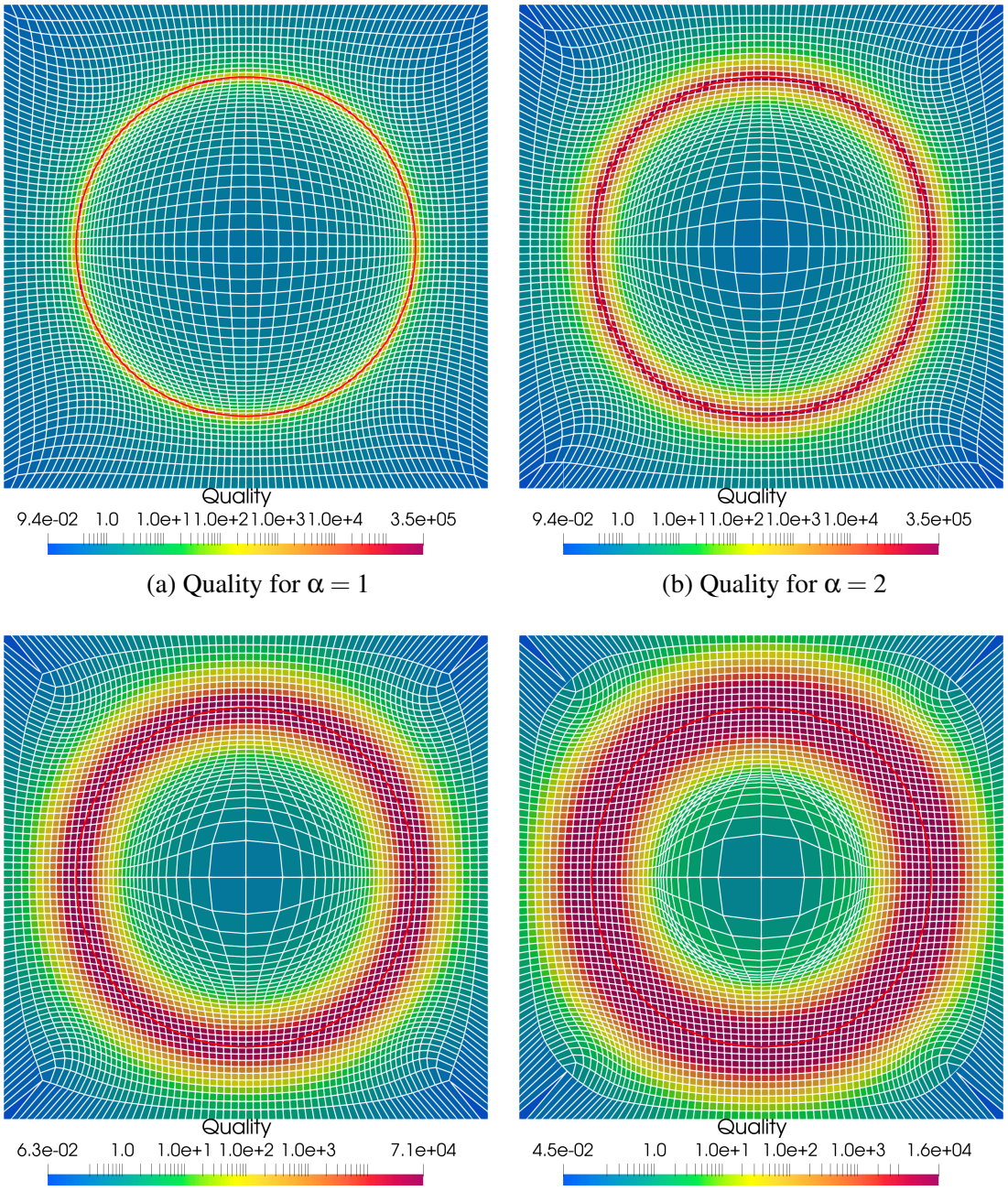
Figure 6.3: Deformed rhombus with interior angle close to π

This way they met the requirement of the desired volume but were extremely deformed so that it was difficult to run simulations on the resulting mesh. However, these extreme deformations should be prevented by the growth-conditions of the functional ((3.1.2) and (3.1.3)). Therefore, let us now investigate for $\alpha = 1,2,3,4$ the quality of the mesh. Here, we define the quality slightly different: we set the quality as the ratio

$$Q = \frac{\text{actual cell volume}}{\text{desired cell volume}}$$

In this case is necessary for a new definition of quality because we are minimizing the functional value, so the functional values in each cell will be relatively small. This can actually be seen in Figure 6.5. In ideal situations the define cell quality should always be close to one. However, we cannot really expect this when we are close to circle because there d would be very small, but we expect the quality to be somehow close to one in most parts of the mesh. In Figure 6.4 we see that with increasing α the quality decreases: The part of the mesh where this ratio is extremely high is increasing, but the overall geometrical quality of the mesh stays good - except the corners, but we get to that. We hereby see that the growth conditions actually help to get a geometrical good mesh and the form of the functional „prefers“ a mesh that is geometrically good over a mesh where all cell-volumes are exactly as prescribed. This is really good and can be used as a check

when this method is applied to real simulations: If there are many cells where this ratio is so bad, then the mesh might simply not have enough cells for the desired resolution, so if a user needs the resolution that results from $\alpha = 3$ or $\alpha = 4$ then the user needs to retry on a finer level. It is important to note that this information can be obtained by just looking at the quality of one mesh. In contrast to this we would not be able to tell this by checking only the functional value in each cell: Only when comparing the results for different values of α we notice that the lower bound of the scale shifts by an order of magnitude per increase in α, so this is some indication, but if we would have the result from just one α we would not be able to judge based on that.
We do not consider the corners too much because we see this mesh more as a prototype of a real-world situation where the mesh might extend even further. Because of the geometrical situation we are in the boundary conditions allow only one vertex to be moved, so obviously this will generate bad results for this one particular cell.



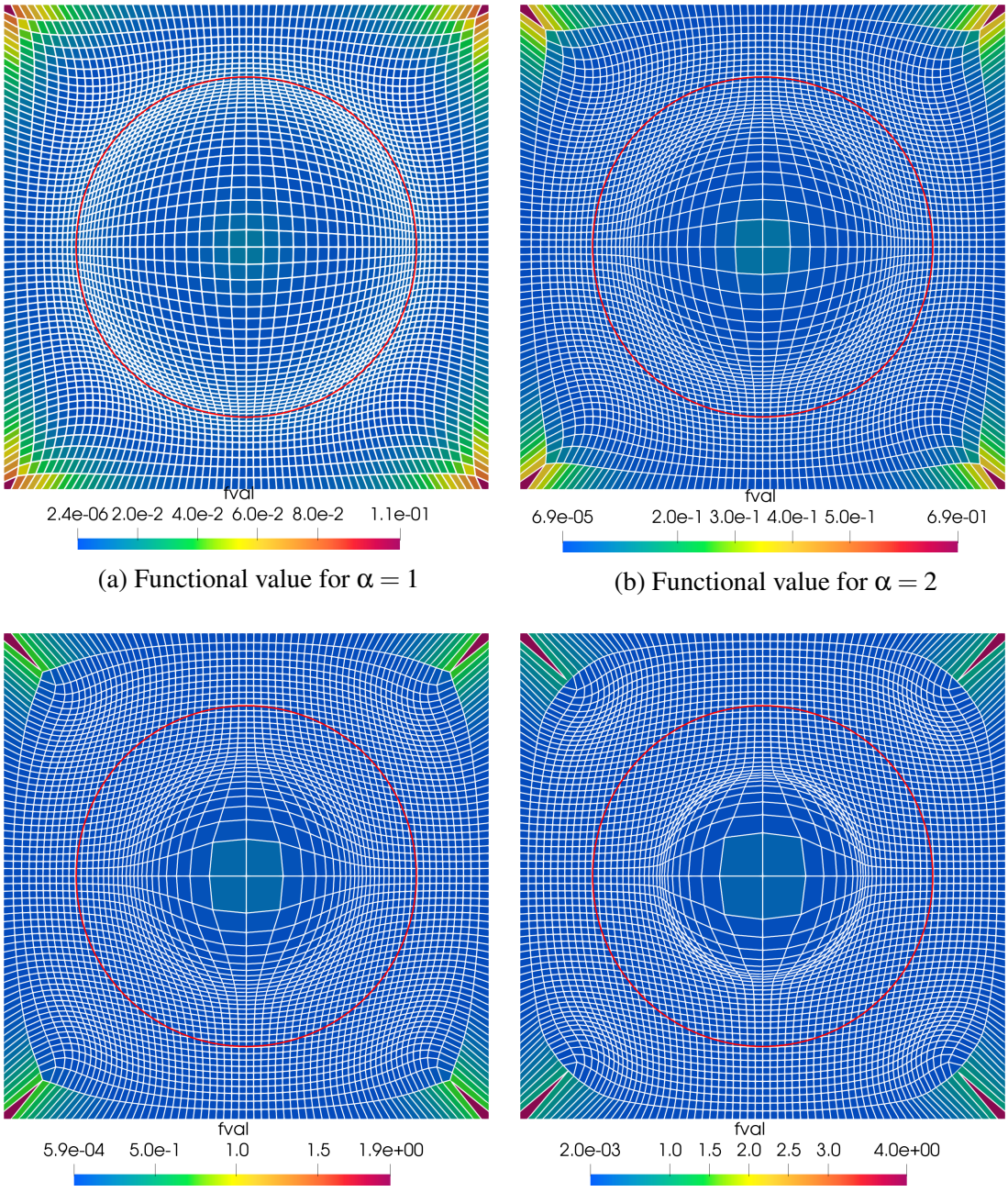
(a) Quality for $\alpha = 1$

(b) Quality for $\alpha = 2$

(c) Quality for $\alpha = 3$

(d) Quality for $\alpha = 4$

Figure 6.4: Adapting a square to a circle on level 6 - Quality



(a) Functional value for $\alpha = 1$

(b) Functional value for $\alpha = 2$

(c) Functional value for $\alpha = 3$

(d) Functional value for $\alpha = 4$

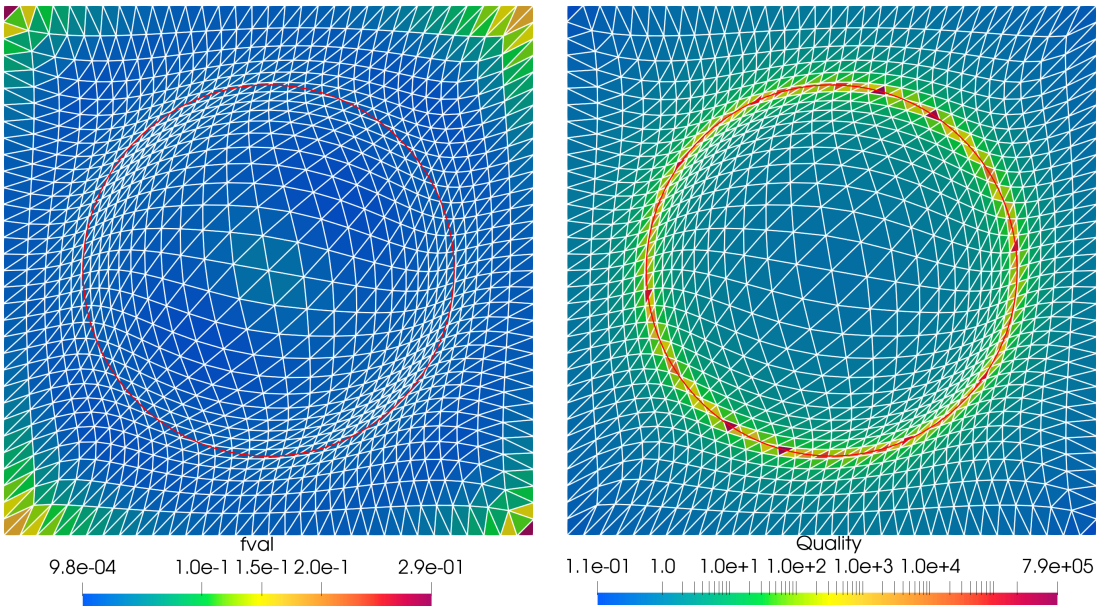
Figure 6.5: Adapting a square to a circle on level 6 - Functional values

6.3. Adapting a square to a circle - triangle mesh

Let us now repeat the rest from Section 6.1 on a mesh with triangles on level 5.
As before, we judge the quality of a cell not only by the functional value but also by

$$Q = \frac{\text{actual cell volume}}{\text{desired cell volume}}$$

In Figure 6.6 we see that visually, the result is very good. We also see some cells where



(a) Functional value

(b) Quality

Figure 6.6: Adapting a square to a circle on level 4 on triangle mesh

it might have been necessary to add 10^{-8} to the relative cell volume because their center is so close to the circle that else the desired cell volume might have been 0. While these results look very promising, it took us 87 Newton-Iterations to the final result. Therefore, we just give a brief summary of all numerical operations:

- Newton-Iterations: 87
- Total number Armijo-Steps: 237
- Total number of PCG-Steps for δ: 2187
- Total number of PCG-Steps for s: 3426
- δ rejected: 7 times

Before we interpret these results we should mention that triangle are generally more flexible with respect to degeneration. This extra flexibility adds „more freedom to move the vertices“ and therefore, with fewer cells than a mesh with quadrilaterals, a higher quality can be achieved. Therefore, we cannot really compare it to a mesh with quadrilaterals.

Most likely, one could reduce the number of Newton-Iterations significantly by updating the desired cell-volumes earlier as we did, and we took the settings from our experience with the quadrilateral meshes. Because of the better quality of the adaptation compared to the quadrilaterals we are undecided: It definitely takes longer to adapt a mesh that contains triangles, but this is only because it allows more adaptation. If we need a quadrilateral mesh on level X because we need the good approximation to the geometry, we might have to compare it to the cost of the mesh with triangles on level $X-1$. If however we need level X for other reasons, we would have to compare the cost on the same level. Since our focus was mostly on 2d quadrilateral meshes we conclude that we do not really have enough to compare it to, and more tests would be necessary.

6.4. 3D Testcase

We now try something similar to Section 6.1, but in three dimensions. This means that we start with a refined mesh of $[-1,1]^3$ and try to adapt it to a sphere with the radius of 0.7. To measure the quality, we use the same metric as in two dimensions and define describe the quality of a cell not only by the functional value, but also by:

$$Q = \frac{\text{actual cell volume}}{\text{desired cell volume}}$$

Even without going into details we can clearly see in both Figures 6.7 and 6.8 that the method does not seem to work as intended in this case. In Figure 6.9 we see that it is not just the quality that has a bad measurement, but also that the approximation of the sphere is not good. There is no real indication in Table 6.2 that something went wrong here. On a first glance one might think that the final Newton-defect is too large, but the results from iteration 63 do not look any different, we just intentionally let the algorithm run a bit longer. We also do not believe that there were not enough elements to fulfill this geometrically (something that we investigated in 2D in Figure 6.4)because we tried this on level 7, so there were 2097152 elements available. Therefore, we conclude that more research needs to be done in the 3D-Case, our focus was mainly 2d.

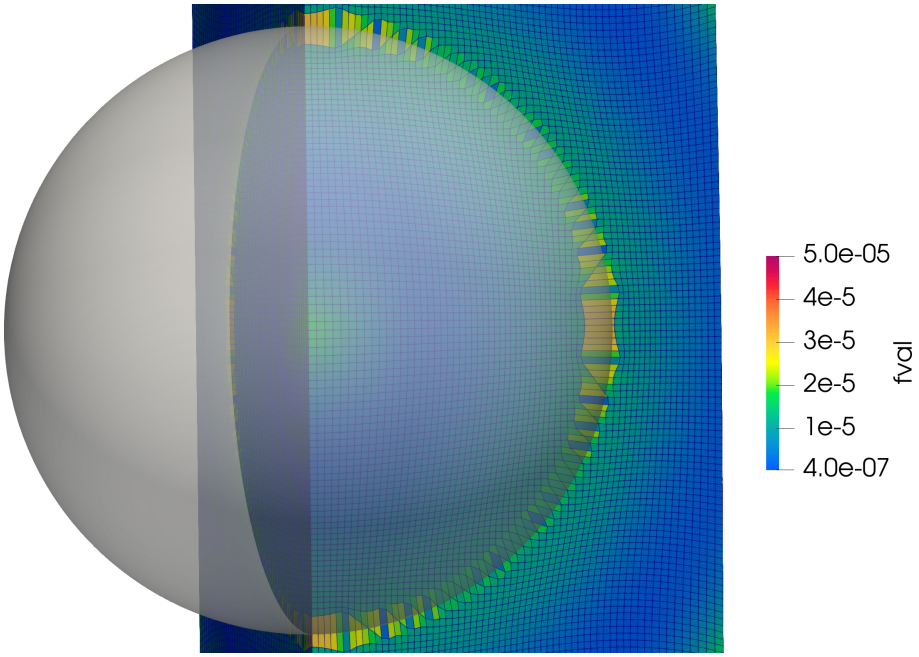


Figure 6.7: Adapting a cube to a sphere - functional value

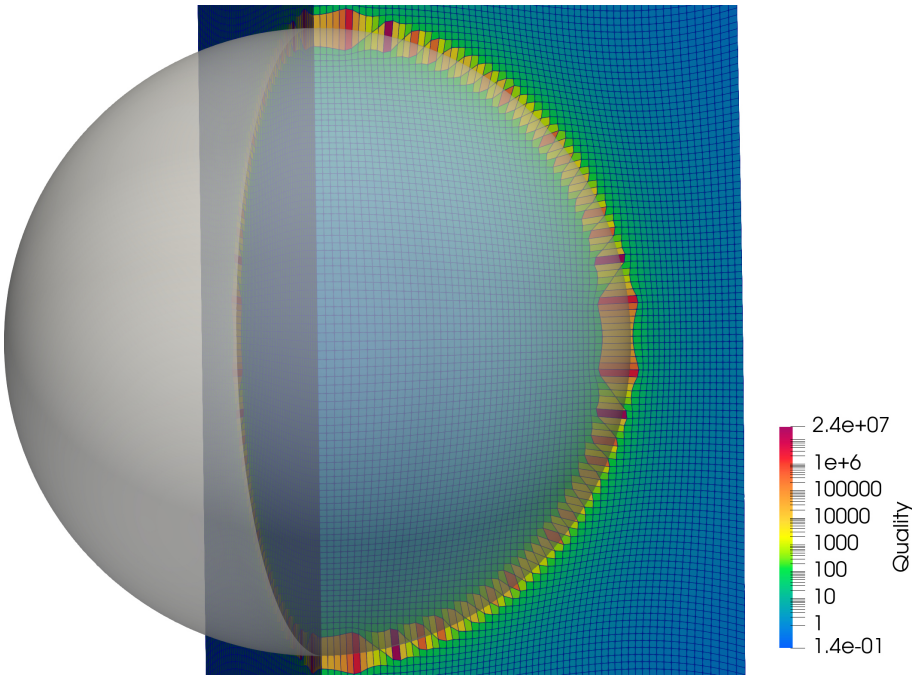


Figure 6.8: Adapting a cube to a sphere - quality

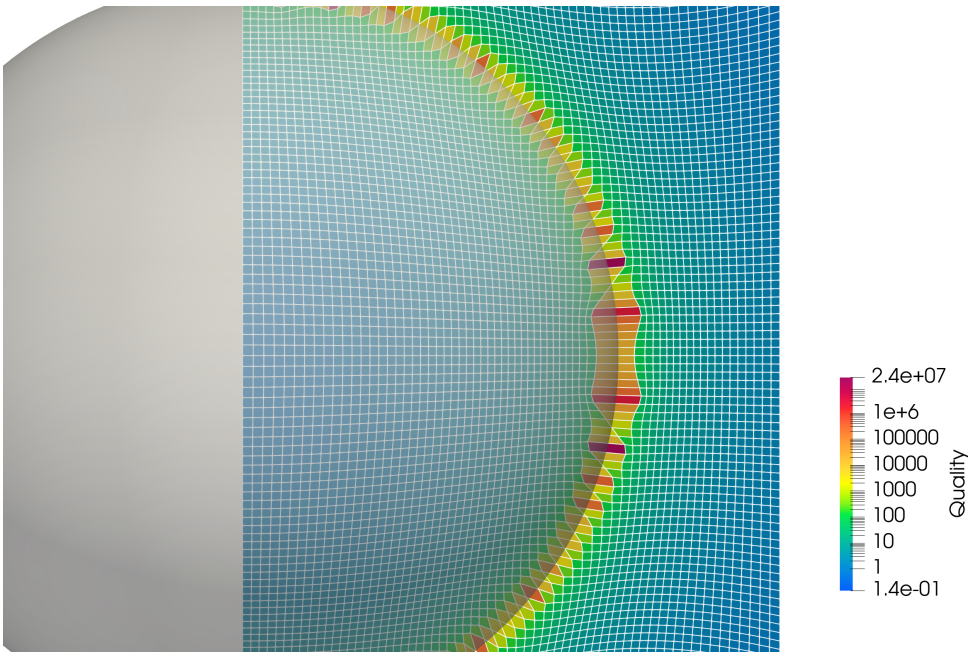


Figure 6.9: Adapting a cube to a sphere - quality - zoom

k	Armijo-Steps	Newton-defect	$\|\|\delta\|\|$	$F(\gamma_k)$	η_k	α	Iter δ	defect δ	Iter. s	used s
1	4	2.66E-01	9.26E+02	22.46	1.00	0.30	-	-	120	TRUE
2	4	1.54E-01	2.48E+02	22.02	1.00	0.30	-	-	120	TRUE
3	0	8.86E-02	5.84E+00	21.95	0.98	0.38	103	1.51E-04	113	FALSE
4	0	5.05E-02	2.16E+00	21.93	0.99	0.47	91	8.78E-05	118	FALSE
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
54	0	5.67E-06	1.31E-05	25.08	0.99	1.00	21	6.99E-08	82	FALSE
55	0	6.99E-08	3.51E-06	25.08	0.93	1.00	44	5.45E-09	105	FALSE
56	9	5.45E-09	4.00E-06	25.18	1.00	1.00	-	-	105	TRUE
57	3	2.68E-02	1.37E+01	25.18	1.00	1.00	-	-	103	TRUE
58	3	7.09E-03	7.66E+00	25.18	1.00	1.00	-	-	106	TRUE
59	0	2.39E-03	3.82E-01	25.18	0.96	1.00	100	6.73E-06	115	FALSE
60	0	7.36E-06	6.19E-04	25.18	0.96	1.00	22	2.25E-06	114	FALSE
61	0	2.25E-06	2.89E-04	25.18	0.96	1.00	113	7.26E-09	107	FALSE
62	0	7.26E-09	3.29E-07	25.18	0.90	1.00	20	2.23E-09	110	FALSE
63	0	2.23E-09	2.01E-06	25.15	1.00	1.00	-	-	112	TRUE
64	3	1.41E-02	5.67E+00	25.15	1.00	1.00	-	-	101	TRUE
65	3	3.69E-03	3.05E+00	25.15	1.00	1.00	-	-	105	TRUE
66	0	1.16E-03	1.50E-01	25.15	0.96	1.00	102	3.38E-06	110	FALSE

Table 6.2: Iterations adapting a cube to a sphere

7

Conclusions and Outlook

In this thesis we discussed some aspects of PDE-based mesh optimization. While non-PDE-based mesh optimization techniques exist, we chose the PDE-based approach because this allows the application of several theorems which guarantee the existence of the solution and the convergence of the algorithm. Because it is a well-known fact in continuum mechanics that large deformations can only be modeled with nonlinear models we did not investigate the behavior of linear models, for example smoothing with a Laplace-operator (as Paul tried in [Pau17]).
Therefore, we took a „slightly nonlinear“ model of the Neo-Hookean class to calculate the deformation. There exist many possible solvers for nonlinear optimization, but our goal was to benefit from the local fast convergence of the Newton-Scheme. The reason is that for „real-life“ applications where the meshdeformation is just a part of for example a CFD-simulation it is crucial to reduce the time spent in the preprocessing and meshgeneration, and mesh optimization is a part of this. For the optimization problems we generally find two different approaches:

- First discretise the model and then apply an optimization method for functions $F : \mathbb{R}^n \to \mathbb{R}$ on this discretisation.
- Formulate the optimization method and all conditions for a functional $F : H^k \to \mathbb{R}$ and then discretise those „sub-problems“.

In the optimization theory, both approaches often yield the same result, so theoretically it should not matter which approach we choose. When looking deeper into Newton's methods for our problem we found that most of them have modifications to globalize the convergence, but their downside is often that the linear equations that need to be solved loose their sparsity. Therefore, we chose the other approach and formulated a Quasi-Newton-Method in a Hilbert space. To do so, we had to find a continuous approximation of the second derivative. We then showed that this approximation fulfills the Dennis-Moré-Condition so that it conserves the local fast convergence of Newton's method. Since this approximation is based on a heuristic we have to check the angle-condition so that we know the approximation is „not too bad“. After this we investigated other properties

of the functional to ensure that the equation for the Newton-Update (which is a PDE in a function space) has a unique solution. During the literature research for this thesis we found that there were only few algorithmic approaches in a function space (for example [DLR15]), most of the literature was limited to existence theory.
Overall, we found that for our chosen functional a solution is only guaranteed to exist for a certain set of Lamé parameters. While this is a serious drawback of this model in the context of continuum mechanics, this is no problem for our approach: In the context of continuum mechanics the task is to simulate a certain material. Each material comes with its fixed set of material parameters, so the Lamé parameters are fixed. In our approach the situation changes: We are not interested in a particular material behavior, so we can choose them freely.
When putting all of this together, we found the expected problems of Newton's method: Usually, Newton's method needs some sort of dampening, and the convergence is usually not that good if the initial guess is too far away from the solution. If the mesh optimization is applied in a time-dependent simulation (for example a rising bubble) where the mesh does not change too much from one timestep to the other this might be the case, but we chose to test it as a standalone tool. We therefore had to find methods to make the solver more robust. This goal was achieved with two components:

- To calculate a damping parameter for Newton's method the step size has to fulfill the Armijo-condition. We talk about step size here and not about a damping parameter because when one writes it down, both mean the same but from a different perspective.
- To overcome the slow convergence in the first steps we chose to use the negative gradient as the direction of the update which results in a fixed point method. This might seem counter intuitive, but if the initial mesh is „too far away“ from an optimal solution this gives better results. Additionally, it saves the time to assemble and solve one partial differential equation.

With these modifications we then tested the approach to smoothen several 2d-meshes and went on to adapt it to a circle. We also briefly tested the approach in 3d. During our tests we found that the method looks very promising in 2d, but more investigations need to be done to draw conclusions in three dimensions.

Outlook

Obviously we did not have the capabilities to test every single aspect of this method. As we indicated during our tests, we still do not know the optimal stopping criterion for Newton's method. The reason is that for practical applications we should differ between „optimal“ and „good enough“ meshes. What does this mean? When looking closer (for example in Figure 5.10) we found that visually, the mesh does not change too much after the first few iterations. However, when checking the Newton-defect (Table 5.15) we found no reason to stop iterating. We also decided to not base the criterion on $||\delta||$ or its changes because when still far away from the optimum, they might indicate to stop the iteration. A better way to stop the iteration would surely help this in a real life application.
Another thing we noticed was that fixing the boundary nodes does not help the mesh-quality. From a practical point of view, it would help to let the nodes slide on the bound-

ary segments, but from a theoretical point of view we need to be able to still solve this problem in H_0^1 so that we can choose $(u,v) = \int \nabla u \nabla v$ as dot-product. This is a bit overlooked in this work, but it is a huge advantage that the PDE for the Riesz-representative of the antigradient is in fact a Poisson-equation: The Poisson-equation is well known and many people have researched it, and many colleagues from my chair had developed fast algorithms to solve it. This is the reason why we chose to not count these costs to the numerical costs: For research purposes, we took a quite general approach to solve the antigradient equation to be able to modify it easily, but in a real-life application this would certainly be done better.
We should also note that we used only PCG to solve the linear problem for the Newton-update. While this was certainly sufficient in most of our cases, there were some cases where a Multigrid-Solver might have paid off, especially if one does further investigations in 3d. To implement a multigrid solver for this problem comes with its own technical difficulties: We calculate the meshdeformation on the finest mesh, but then we have to use this information to calculate the deformations for a full grid hierarchy. Since the mesh changes in every iteration, one might think that you have to assemble the matrices on the coarser grids and the transfer operators in each Newton-Iteration, but it might be worth to investigate if this is really the case: We just pointed out earlier that the mesh does not change too much after the first few iterations, so maybe everything on the coarser meshes can be left untouched after these few iterations?
With this approach, further testing for 3d-meshes could be performed in a fast manner. We indicated in Section 6.4 that further tests are necessary to investigate if this method can be fully utilized in 3d. Further tests would be to adjust the choice of the cell volumes, to incorporate the final position of the cells in the choice of volumes or to add a penalty term to the functional that becomes large if a „large fraction“ of a cell is cut by the prescribed boundary. This term would have to be designed in such a way that it does not violate the original growth conditions of the functional.

Bibliography

[AF08] Robert A. Adams and John J. F. Fournier. *Sobolev Spaces*. Number 140 in Pure and Applied Mathematics. Academic Press, Amsterdam Heidelberg, 2. ed., reprinted edition, 2008.

[Alt12] Hans Wilhelm Alt. *Lineare Funktionalanalysis: Eine anwendungsorientierte Einführung*. Springer-Lehrbuch Masterclass. Springer Berlin Heidelberg, Berlin, Heidelberg, 6. aufl. 2012 edition, 2012.

[Bal76] John M. Ball. Convexity conditions and existence theorems in nonlinear elasticity. *Archive for Rational Mechanics and Analysis*, 63(4):337–403, December 1976.

[Bän91] Eberhard Bänsch. Local mesh refinement in 2 and 3 dimensions. *IMPACT of Computing in Science and Engineering*, 3(3):181–191, September 1991.

[BCD09] O. Boiron, G. Chiavassa, and R. Donat. A high-resolution penalization method for large Mach number flows in the presence of obstacles. *Computers & Fluids*, 38(3):703–714, March 2009.

[BMdY11] Boubakeur Benahmed, Hocine Mokhtar-Kharroubi, Bruno de Malafosse, and Adnan Yassine. Quasi-Newton methods in infinite-dimensional spaces and application to matrix equations. *Journal of Global Optimization*, 49(3):365–379, March 2011.

[Bra07] Dietrich Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 4., überarbeitete und erw. aufl edition, 2007.

[Cia78] Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems*. Number v. 4 in Studies in Mathematics and Its Applications. North-Holland Pub. Co. ; sole distributors for the U.S.A. and Canada, Elsevier North-Holland, Amsterdam ; New York : New York, 1978.

[Cia88] Philippe G. Ciarlet. *Mathematical Elasticity*. Number v. 20, 27, 29 in Studies in Mathematics and Its Applications. North-Holland ; Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co, Amsterdam ; New York : New York, N.Y., U.S.A, 1988.

[Cla20] Christian Clason. *Introduction to Functional Analysis*. Compact Textbooks in Mathematics. Birkhäuser, Cham, Switzerland, 2020.

[CN85] Philippe G. Ciarlet and Jindřich Nečas. Unilateral problems in nonlinear, three-dimensional elasticity. *Archive for Rational Mechanics and Analysis*, 87(4):319–338, December 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, August 1972.

[Dav04] Timothy A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, June 2004.

[Deu11] Peter Deuflhard. *Newton Methods for Nonlinear Problems*, volume 35 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[DLR15] Juan Carlos De Los Reyes. *Numerical PDE-constrained Optimization*. SpringerBriefs in Optimization. Springer, Cham Heidelberg, 2015.

[DY99] Y. H. Dai and Y. Yuan. A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property. *SIAM Journal on Optimization*, 10(1):177–182, January 1999.

[Fle64] R. Fletcher. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, February 1964.

[Gra08] Matthias Grajewski. *A New Fast and Accurate Grid Deformation Method for R-Adaptivity in the Context of High Performance Computing*. PhD thesis, TU Dortmund, 2008.

[HKT14] Michael Hinze, Michael Köster, and Stefan Turek. Space-Time Newton-Multigrid Strategies for Nonstationary Distributed and Boundary Flow Control Problems. In Günter Leugering, Peter Benner, Sebastian Engell, Andreas Griewank, Helmut Harbrecht, Michael Hinze, Rolf Rannacher, and Stefan Ulbrich, editors, *Trends in PDE Constrained Optimization*, volume 165, pages 383–401. Springer International Publishing, Cham, 2014.

[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, December 1952.

[HZ05] William W. Hager and Hongchao Zhang. A New Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search. *SIAM Journal on Optimization*, 16(1):170–192, January 2005.

[Jened] Robert Jendrny. *Operator-Adaptive Newton Solver for 3D Non-Newtonian Flow with Application to Extrusion Dies*. PhD thesis, TU Dortmund, Dortmund, to be published.

[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, March 2012.

[LV07] Qianlong Liu and Oleg V. Vasilyev. A Brinkman penalization method for compressible flows in complex geometries. *Journal of Computational Physics*, 227(2):946–966, December 2007.

[Man17] Saptarshi Mandal. *Efficient FEM Solver for Quasi-Newtonian Flow Problems with Application to Granular Materials*. PhD thesis, TU Dortmund, 2017.

[MI05] Rajat Mittal and Gianluca Iaccarino. IMMERSED BOUNDARY METHODS. *Annual Review of Fluid Mechanics*, 37(1):239–261, January 2005.

[Mos14] Jörn Mosler. Nonlinear Continuum Mechanic. 2014.

[MV15] Andreas Meister and C. Vömel. *Numerik linearer Gleichungssysteme: eine Einführung in moderne Verfahren ; mit MATLAB-Implementierungen von C. Vömel*. Lehrbuch. Springer Spektrum, Wiesbaden, 5., überarb. aufl edition, 2015.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 2nd ed edition, 2006.

[Pat18] Hemant Kumar Pathak. *An Introduction to Nonlinear Analysis and Fixed Point Theory*. Springer Singapore : Imprint: Springer, Singapore, 1st ed. 2018 edition, 2018.

[Pau17] Jordi Paul. *Nonlinear Hyperelasticity-based Mesh Optimisation*. PhD thesis, TU Dortmund, 2017.

[Pol69] B.T. Polyak. The conjugate gradient method in extremal problems. *USSR Computational Mathematics and Mathematical Physics*, 9(4):94–112, January 1969.

[PR69] E. Polak and G. Ribiere. Note sur la convergence de méthodes de directions conjuguées. *Revue française d'informatique et de recherche opérationnelle. Série rouge*, 3(16):35–43, 1969.

[Pra57] William Prager. On Ideal Locking Materials. *Transactions of the Society of Rheology*, 1(1):169–175, January 1957.

[RGR+21] Hannes Ruelmann, Markus Geveler, Dirk Ribbrock, Peter Zajac, and Stefan Turek. Basic Machine Learning Approaches for the Acceleration of PDE Simulations and Realization in the FEAT3 Software. In Fred J. Vermolen and Cornelis Vuik, editors, *Numerical Mathematics and Advanced Applications ENUMATH 2019*, volume 139, pages 449–457. Springer International Publishing, Cham, 2021.

[Rum96] M. Rumpf. A variational approach to optimal meshes. *Numerische Mathematik*, 72(4):523–540, February 1996.

[Sch] Ben Schweizer. *Partielle Differentialgleichungen: Eine anwendungsorientierte Einführung*. Partielle Differentialgleichungen.

[TGB+10] Stefan Turek, Dominik Göddeke, Christian Becker, Sven H. M. Buijssen, and Hilmar Wobker. FEAST-realization of hardware-oriented numerics for HPC simulations with finite elements. *Concurrency and Computation: Practice and Experience*, 22(16):2247–2265, November 2010.

[Tur99] Stefan Turek. *Efficient Solvers for Incompressible Flow Problems*, volume 6 of *Lecture Notes in Computational Science and Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[TWR03] Stefan Turek, Decheng Wan, and Liudmila S. Rivkind. The Fictitious Boundary Method for the Implicit Treatment of Dirichlet Boundary Conditions with Applications to Incompressible Flow Simulations. In Timothy J. Barth, Michael Griebel, David E. Keyes, Risto M. Nieminen, Dirk Roose, Tamar Schlick, and Eberhard Bänsch, editors, *Challenges in Scientific Computing - CISC 2002*, volume 35, pages 37–68. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[Yam08] Nobuo Yamashita. Sparse quasi-Newton updates with positive definite matrix completion. *Mathematical Programming*, 115(1):1–30, September 2008.