



CHEMNITZ UNIVERSITY OF  
TECHNOLOGY

---

# Numerical Aspects in Optimal Control of Elasticity Models with Large Deformations

Andreas Günzel

Dissertation submitted to the

**Faculty of Mathematics**

at

**Chemnitz University of Technology**

in accordance with the requirements for the degree Dr. rer. nat.

Advisor: Prof. Dr. Roland Herzog

Chemnitz, April 1, 2014







Gewidmet meinem Großvater

Hans Günnel







# Contents

Acknowledgments	viii
Abstract	ix
Chapter 1. Introduction	1
1.1 Motivation	1
1.2 Outline of the Thesis	3
1.3 Notation	5
1.4 Nomenclature	6
Chapter 2. A Mathematical Model of Elasticity with Finite Deformations	11
2.1 Modeling of Finite Deformations	12
2.2 Models for Elastic Material Behavior	14
2.2.1. Balance-of-Forces Approach	14
2.2.2. Energy Minimization Approach	18
2.2.3. Loads (Volume, Boundary, Inner Pressure, Fiber Tension)	23
2.3 Comparison to Linear Elasticity	32
2.4 Parametrization of the Reference Configuration $\Omega$	36
2.5 Hierarchical Plate Model	38
Chapter 3. Numerical Methods to Solve the Forward Problem	47
3.1 Newton's Method	48
3.2 Globalization by a Line Search	49
3.3 Krylov Subspace Methods	56
3.4 Discretization	61
3.5 Preconditioner and Multigrid Method	65
Chapter 4. Optimal Control Problems in Elasticity	72
4.1 Setting of the Optimal Control Problem	73



4.1.1. Quality functionals	74
4.1.2. Cost or Penalty Functionals	84
4.2 Lagrange-Newton: An All-at-once Approach	85
4.2.1. Solving the Lagrange Equation	86
4.2.2. Discretization	88
4.3 Quasi-Newton: A Reduced Formulation	89
4.3.1. Quasi-Newton Method	93
4.3.2. Broyden-Fletcher-Goldfarb-Shanno-Update	94
4.3.3. Simple Wolfe-Powell Line Search	97
4.3.4. Discretization	100
Chapter 5. Implementation in FENICS	104
5.1 Introduction to FENICS	105
5.2 Weak Formulations and UFL	109
5.2.1. Mutual Definitions	110
5.2.2. Objective Functions	113
5.2.3. Forward Problem	115
5.2.4. Lagrange Problem	118
5.2.5. Reduced Optimal Control Problem	121
5.2.6. Automatic Differentiation vs. Differentiation by Hand	124
5.2.7. Parametrization	126
5.2.8. Hierarchical Plate Model	131
5.3 Solver Routines	137
5.3.1. Multigrid Method	137
5.3.2. CG and MINRES Method	140
5.3.3. Forward Problem	141
5.3.4. Lagrange-Newton Problem	142
5.3.5. Reduced Problem	144
5.4 An Experimental Preconditioner	147
Chapter 6. Numerical Experiments	151
6.1 Experiments on the Forward Problem	153
6.1.1. Line Search and Iterates of Newton's Method	153
6.1.2. Plate Model	156
6.2 Experiments on the Optimal Control Problem	159
6.2.1. Elevated Bar with Fiber Tension Control	159
6.2.2. Regional Penalization	167
6.2.3. Flower Movement by Turgor Pressure	170
6.2.4. Enclosed Volume	173



---

Chapter 7. Conclusions and Perspectives	177
Appendix A. Appendix	181
A.1 Matrix Calculus	181
A.2 Derivatives of Energy Functionals	183
Appendix B. Theses	186
Appendix C. Curriculum Vitæ	188
Bibliography	191



# Acknowledgments

My deepest gratitude goes to my supervisor Roland Herzog. Not only for offering me the possibility to work at the TU Chemnitz and to write my thesis, but also for his support and patience with me. His advice and remarks were most helpful for my work and he taught me a lot in the last years.

Arnd Meyer introduced me to the area of elasticity and plate theory, for which I am very grateful. Also, I thank Anton Schiela for the discussions on optimal control and his remarks.

My position at the TU Chemnitz was partially funded by the BMBF project *Qualitätspakt Lehre* and the DFG Excellence Cluster MERGE, which is much appreciated.

I want to express my gratitude for the people working at the faculty of mathematics at the TU Chemnitz for the enjoyable work climate. First of all, I thank my working group: Roland Herzog, Martin Bernauer, Frank Schmidt, Gerd Wachsmuth, Susann Mach, Ilka Riedel, Tommy Etling, Felix Ospald (especially for his help with FMG), Jan Blechschmidt and Ailyn Schäfer. Not only for the mathematical discussions but also for the social events. This also includes Arnd Meyer, Hansjörg Schmidt, René Schneider, Matthias Pester and Jens Rückert. Furthermore, I would like to thank the secretaries Anne-Kristin Glanzberg, Kerstin Seidel and Heike Weichelt as well as the people from the MRZ, Margit Matt, Elke Luschnat, Mathias Meier and Roman Unger.

In the last years, I had the opportunity to teach several courses in mathematics and I am very thankful for this experience and the feedback my students gave me.

My parents supported me throughout all my life and I am very thankful for all their care and nurture. This includes my dear brother Heiko as well, on whom I always can rely on. Finally, I thank Marcella in my heart of hearts for her constant encouragement, reading the manuscripts carefully and taking up with me in the last months when I was busy writing this thesis. I can only hope that I will make it up to her someday.



# Abstract

This thesis addresses optimal control problems with elasticity for large deformations. A hyperelastic model with a polyconvex energy density is employed to describe the elastic behavior of a body. The two approaches to derive the nonlinear partial differential equation, a balance of forces and an energy minimization, are compared. Besides the conventional volume and boundary loads, two novel internal loads are presented. Furthermore, curvilinear coordinates and a hierarchical plate model can be incorporated into the formulation of the elastic forward problem.

The forward problem can be solved with Newton's method, though a globalization technique should be used to avoid divergence of Newton's method. The repeated solution of the Newton system is done by a CG or MINRES method with a multigrid V-cycle as a preconditioner.

The optimal control problem consists of the displacement (as the state) and a load (as the control). Besides the standard tracking-type objective, alternative objective functionals are presented for problems where a reasonable desired state cannot be provided. Two methods are proposed to solve the optimal control problem: an all-at-once approach by a Lagrange-Newton method and a reduced formulation by a quasi-Newton method with an inverse limited-memory BFGS update.

The algorithms for the solution of the forward problem and the optimal control problem are implemented in the finite-element software FENICS, with the geometrical multigrid extension FMG. Numerical experiments are performed to demonstrate the mesh independence of the algorithms and both optimization methods.







# 1 Introduction

## 1.1 Motivation

Elastic deformations are a common phenomenon in nature: for example, large parts of the tissues of animals are elastic, otherwise movements would be impossible or would cause damage due to material fatigue. But also in modern technology, elastic parts are often components of machines and devices, which allow a flexible shape or movement. There are cases where the shape of the deformed elastic part is of interest, and the question arises how a certain shape can be attained. We consider the case where an exterior or interior load acts on the body and we try to find a load that deforms the body in a desired way. This will be formulated as an optimal control problem.

As we seek a noticeable change in the shape of the body, we expect the deformation to be large, which requires an elasticity model for large (non-infinitesimal) deformations. These models are inherently nonlinear and challenging to solve. In this thesis, we focus on polyconvex hyperelasticity models, which can be formulated as an energy minimization problem. This will constitute our *forward problem*: the computation of the deformation of an elastic body for a given load.

This forward problem is the basis for the optimal control problem, where we seek to find a load such that the deformation is close to the desired one. One possible application in medicine of implant design is investigated by Lars Lubkoll, Anton Schiela and Martin Weiser, see Lubkoll et al. (2012). Consider a patient who lost a part of his jaw bone along with the skin due to an accident. The bone could be replaced by an implant that is reconstructed from the undamaged part of the intact jaw bone. The skin can be transplanted from another part of the body, but the thickness and elastic behavior is likely to be different than that of the original facial skin. As a consequence, the facial reconstruction might be unsatisfactory to the patient as symmetry in facial structures is a key aspect to appearance. In order to improve this procedure, the implant could be modeled in such a way that the resulting facial structure matches the undamaged right half of the jaw. The authors have shown that the optimal control problem with a contact between the rigid implant and the elastic skin can be reformulated as an optimal control problem with a boundary load. In this setting, the deformation of the skin is the state and the boundary load on the inside is the control. Other applications can be found in industrial manufacturing.



The elasticity model which is used in this thesis is fundamental to extensions like elasto-plastic material models for large deformations. Several mathematicians in Germany are working on the optimal control of elasto-plasticity. The *Sonderforschungsbereich (SFB) 666: Integral Sheet Metal Design with Higher Order Bifurcations* has a project on optimal control of deep-drawing processes, investigated by Stefan Ulbrich and Daniela Bratzke from the TU Darmstadt. Furthermore, Michael Stingl and Stefan Werner from the Friedrich-Alexander University Erlangen-Nürnberg worked on the optimal control of hot rolling processes, in which friction plays an important role. Gerd Wachsmuth and Roland Herzog from the TU Chemnitz and Christian Meyer from the TU Darmstadt made major contributions to the analysis and stationarity of optimal control problems in the case of small deformations, see Herzog et al. (2013).

Another related type of problem is shape optimization. Benedikt Wirth from the University of Münster proposed to measure the difference between two body shapes by the required physical energy to deform the current body such that it matches the desired shape, see Rumpf and Wirth (2009), Rumpf and Wirth (2011) and Penzler et al. (2012).

The contributions of this thesis is extended in several directions:

### **Modeling**

As we seek to describe biological phenomena, we need to model certain forces that are commonly found in animals and plants. First, we show how an inner pressure, called turgor pressure in plants, can be understood as an interior load and how it can be integrated into the elasticity model for large deformations. Other examples are muscle tissues where the muscle fiber causes a tension along its fiber direction. We present a model for this kind of interior load and show how it fits into the elasticity model.

### **Hierarchical Plate Model**

Large deformations often occur when the body can be easily deformed, for example because it is a thin plate. Here, full 3D formulations of the deformation problem suffer from discretization problems, often referred to as locking. In order to avoid this locking and exploit the structure of the deformation, several plate models have been proposed. While some are only applicable for very thin plates, we use the hierarchical plate model which can also be applied to thicker plates. We show how this ansatz is incorporated into the elasticity model and the optimal control problem.

### **Quality Functionals for the Optimal Control Problem**

In a large part of the mathematical literature on optimal control, a tracking-type objective functional with a desired state is presented and used. While this functional is not only plausible for many problems, it is also practical for the analysis of these problems. However, there are many situations in elasticity where we cannot



provide a desired state for the objective functional. We present a couple of examples where a desired state is not available and give alternatives how to formulate appropriate objective functionals for these examples. Furthermore, we use shape derivatives to obtain simple representations of the first derivative of these novel objective functionals.

### Optimization Methods

As the number of degrees of freedom might be large for partial differential equations in 3D, we need methods that are capable of solving large-scale problems. On the one hand, this means that an algorithm should ideally scale linearly with the problem size, otherwise the computational costs might be too high. In order to achieve this, we formulate the iterative optimization methods and the solver for the linear systems in a function space setting. This approach is natural for this type of problems and eventually allows us to construct a quasi-Newton method to solve the optimal control that shows a mesh independent behavior.

### Implementation

All presented algorithms and variational equations were implemented in the finite element software FENICS. Even though many routines, like Newton's method or the conjugate gradient method, were already present in FENICS, we had to implement them anew to modify them to our requirements. This also includes a line search method for the forward problem. The possible divergence of Newton's method for elasticity problems with large deformations can be resolved by a globalization technique. We propose a new line search objective which mimics the stored energy and is applicable in many situations like the case of inhomogeneous Dirichlet boundary conditions or the lack of a formulation of the forward problem based on a stored energy.

## 1.2 Outline of the Thesis

Chapter 2 – A Mathematical Model of Elasticity with Finite Deformations . This chapter presents the modeling of elasticity for large deformations. First we compare two different approaches to obtain a variational equality for the forward problem: the classical balance of forces and an energy minimization. Both approaches yield the same variational equality, though the energy minimization allows the usage of methods that can be motivated from optimization, e.g. a line search as a globalization technique. Next, we present the modeling of two interior loads and how they fit into the elasticity model. Finally, we demonstrate how curvilinear coordinates and the hierarchical plate model can be incorporated into the full 3D model.



Chapter 3 – Numerical Methods to Solve the Forward Problem . The forward problem is a nonlinear partial differential equation and Newton’s method is likely to diverge for large deformations. As a remedy, we present several globalization techniques, among them a new method built on a guiding function. In order to solve the linear system in Newton’s method, we use a conjugate gradient or minimal residual method. At this point, all algorithms are formulated in a function space setting, which allows the correct choice of norms and scalar products. Next, we use a finite element method to discretize the problem, including a multigrid method to build a preconditioner for the CG and MINRES method.

Chapter 4 – Optimal Control Problems in Elasticity . The elasticity model is the basis for the optimal control problem, with the displacement as the state and a load as the control. First, we present a couple of examples for the objective functional. We point out that a classical tracking type functional is not always applicable as we might not be able to provide a desired state. As an alternative, We give three examples for objective functionals. We present two optimization methods to solve the optimal control problem: a Lagrange-Newton method, and a quasi-Newton method with BFGS update. Again, both are formulated in Hilbert spaces, which is particularly rewarding when constructing the BFGS update for the quasi-Newton method.

Chapter 5 – Implementation in FENICS . The presented algorithms are implemented in the finite element software FENICS. We give a brief introduction to FENICS and how it can be used to solve partial differential equations. Next, we demonstrate how the several forms and functions of the optimal control problem can be implemented. This includes the curvilinear coordinates as well as the hierarchical plate model. Finally, we propose an experimental preconditioner that uses the current stiffness matrix rather than the stiffness matrix from the linear model of elasticity. However, the current matrix might be indefinite, and we propose a method to avoid this problem.

Chapter 6 – Numerical Experiments . In this chapter we present several numerical experiments to test the implemented algorithms. First, we compare the globalization techniques for the forward problem by the example of a thick plate. Furthermore, we use this example to compare the hierarchical plate model with the full 3D model. Next, the four optimal control problems demonstrate the different objective functions. Here, we also compare the Lagrange-Newton and quasi-Newton and their advantages. Additionally, we illustrate how curvilinear coordinates and the hierarchical plate model can be used in optimal control of elasticity.



### 1.3 Notation

This section gives an overview over the notation which is used throughout this thesis.

**Domains**  $\Omega$ ,  $\hat{\Omega}$  and  $\tilde{\Omega}$  are distinguished by their marker, e.g. a hat  $\hat{\cdot}$  or a tilde  $\tilde{\cdot}$ . Points within these domains have the same marker, e.g.  $\mathbf{x} \in \Omega$ ,  $\hat{\mathbf{x}} \in \hat{\Omega}$  or  $\tilde{\mathbf{x}} \in \tilde{\Omega}$ . The boundary of  $\Omega$  is denoted by  $\partial\Omega$  and the interior of  $\Omega$  is denoted by  $\text{int } \Omega$ .

**Scalars, vectors and tensors** are denoted by Latin or Greek letters.

- We use lowercase symbols like  $m$  or  $t$  for scalars.
- Bold symbols like  $\mathbf{x}$  or  $\mathbf{U}$  denote vectors in  $\mathbb{R}^3$ .
- Arrows  $\vec{\cdot}$  indicate vectors of  $\mathbb{R}^n$ , such as coefficient vectors.
- Uppercase Symbols like  $F$  refer to tensors, normally of order 2, e.g. matrices.

Exceptions to these rules are the local volume change  $J$ , because this convention is widely used in the literature, and the prolongation operator  $p$  and restriction operator  $r$  (both matrices), as  $P$  is more commonly used for the preconditioner and  $R$  for the Riesz operator.

Components of a vector  $\mathbf{f}$  or a tensor  $F$  are denoted by  $[\mathbf{f}]_i$ , respectively  $[F]_{ij}$ . The only exceptions are the spatial coordinates  $\mathbf{x} \in \Omega$  where we abbreviate the earlier notation by  $\mathbf{x}_i := [\mathbf{x}]_i$  due to their frequent use.

**Functions (fields) defined on domains**  $\Omega$ , like a scalar field  $m : \Omega \rightarrow \mathbb{R}$  or a vector field  $\mathbf{f} : \Omega \rightarrow \mathbb{R}^3$ , inherit the marker of the domain, e.g.  $\mathbf{f} : \Omega \rightarrow \mathbb{R}^3$ ,  $\hat{\mathbf{f}} : \hat{\Omega} \rightarrow \mathbb{R}^3$  or  $\tilde{\mathbf{f}} : \tilde{\Omega} \rightarrow \mathbb{R}^3$ , to emphasize the arguments they have and to possibly drop the arguments for better readability, e.g.  $\hat{\mathbf{f}} = \hat{\mathbf{f}}(\hat{\mathbf{x}})$ . The gradient of scalar fields or the Jacobian of vector fields with respect to the spatial coordinates  $\mathbf{x}$  are denoted both by  $\nabla(\cdot)$ . If the field is defined for example on  $\hat{\Omega}$  and the differentiation is with respect to  $\hat{\mathbf{x}}$ , the differential operator inherits the marker, that is  $\hat{\nabla}(\cdot)$ .

**Functionals defined on (Sobolev) function spaces** are denoted with uppercase letters like  $W$  or  $S$ . Even though we said to use uppercase letters for tensors, the usage of these functionals should make clear that these are not tensor-valued. Their derivatives with respect to an argument is indicated by a subscript, for example  $W_{,\mathbf{U}}$  denotes the derivative of  $W$  with respect to its argument  $\mathbf{U}$ . In this thesis, derivatives with respect to functions are to be understood as formal. The directional derivative of  $W$ , at a point  $\mathbf{U}$  and in a direction  $\delta\mathbf{U}$ , is denoted by  $W_{,\mathbf{U}}(\mathbf{U})[\delta\mathbf{U}]$ . The second derivative in the directions  $\delta\mathbf{U}$  and  $\delta\mathbf{V}$  then reads  $W_{,\mathbf{U}\mathbf{U}}(\mathbf{U})[\delta\mathbf{U}, \delta\mathbf{V}]$ . We use the symbol  $\delta$  for perturbations like  $\delta\mathbf{U}$  in the calculus for differentiation, whereas the symbol  $\Delta$  is used for search directions like  $\Delta\mathbf{U}$  in algorithms like Newton's method. For example, Newton's method would read in this notation

$$\mathbf{U}_{k+1} = \mathbf{U}_k + \Delta\mathbf{U}_k \quad \text{with} \quad W_{,\mathbf{U}\mathbf{U}}(\mathbf{U}_k)[\delta\mathbf{U}, \Delta\mathbf{U}_k] = -W_{,\mathbf{U}}(\mathbf{U}_k)[\delta\mathbf{U}], \quad \forall \delta\mathbf{U}.$$



The set of linear and bounded operators from a Hilbert space  $\mathcal{X}$  to a Hilbert space  $\mathcal{Y}$  are denoted by  $\mathcal{L}(\mathcal{X}, \mathcal{Y})$ . The inner product of a Hilbert space  $\mathcal{X}$  is  $(x_1, x_2)_{\mathcal{X}}$ ,  $x_1, x_2 \in \mathcal{X}$ . The dual space of a Hilbert space  $\mathcal{X}$  is  $\mathcal{X}^*$  and the duality pairing is denoted by  $[y, x]_{\mathcal{X}^*, \mathcal{X}}$ , with  $y \in \mathcal{X}^*$  and  $x \in \mathcal{X}$ .

## Tensor Calculus

- The set of  $3 \times 3$  matrices is denoted by  $\mathbb{M}^3 := \mathbb{R}^{3 \times 3}$  and the subset of matrices with positive determinant is  $\mathbb{M}_+^3 := \{A \in \mathbb{M}^3 : \det A > 0\}$ .
- The identity matrix is denoted by  $I$ , in most cases,  $I \in \mathbb{M}^3$ .
- The Kronecker symbol is denoted by  $\delta_{ij} := \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}$
- The unit vectors  $\vec{e}_i$  are defined by  $[\vec{e}_i]_j = \delta_{ij}$ .
- We do not use the Einstein summation convention in this work. Sums are always denoted by the symbol  $\sum$ , though sometimes we drop the index range if they are clear, e.g.  $\sum_i := \sum_{i=1}^3$ .
- We drop the multiplication dot “ $\cdot$ ” for tensor products, e.g.  $AB := A \cdot B$ . This follows the common literature on linear algebra but not the literature on differential geometry, e.g. Ciarlet (2005), or literature on mechanical engineering, e.g. Spencer (2004) or Bower (2009).
- The inner product between two matrices  $A : B$  in  $\mathbb{M}^3$  is defined as  $A : B := \text{tr}(A^\top B)$  with the trace operator  $\text{tr } A := \sum_{i=1}^3 [A]_{ii}$ . Please note that the definition  $A : B$  is not unified in the literature, some authors define it as  $\text{tr}(AB)$ .
- The Euclidean scalar product between two vectors  $a, b$  is simply  $a^\top b$ . The Euclidean norm of a vector is denoted by  $\|\cdot\|_2$ , that is  $\|\mathbf{x}\|_2^2 := \mathbf{x}^\top \mathbf{x}$ , while the Frobenius norm of a matrix is  $\|\cdot\|_F$ , that is  $\|A\|_F^2 := A : A$ .
- The cofactor of a regular matrix  $A$  is defined by  $\text{cof } A := (\det A) A^{-\top}$ .

**Integrals** over the domain  $\Omega$  are denoted by  $\int_{\Omega} \dots d\mathbf{x}$  and integrals over a part of the boundary  $\Gamma \subset \partial\Omega$  are denoted by  $\int_{\Gamma} \dots dS$ . The differentials  $d\mathbf{x}$  and  $dS$  inherit the marker of the domain, e.g.  $\int_{\hat{\Omega}} \dots d\hat{\mathbf{x}}$ .

**State and Control variables** in literature on optimal control are often called  $y$  and  $u$ . Unfortunately, this clashes with the widely used  $\mathbf{U}$  for the displacement which is our state. Therefore we use  $\mathbf{U}$  for the state (displacement) and  $\mathbf{C}$  for the control (load).

## 1.4 Nomenclature

### Sets and Domains



$\mathbb{M}^3$	set of $3 \times 3$ -matrices
$\mathbb{M}_+^3$	set of $3 \times 3$ -matrices with positive determinant
$\Omega$	reference configuration and undeformed body in 3D, section 2.1
$\hat{\Omega}$	current configuration and deformed body in 3D, section 2.1
$\tilde{\Omega}$	parameter space for curvilinear coordinates to describe $\Omega$ , section 2.4
$\Omega^{2D}$	midsurface of a plate, section 2.5
$\partial\Omega$	boundary of the domain $\Omega$
$\Gamma^N, \hat{\Gamma}^N$	Neumann boundary for the boundary load $\mathbf{g}$ , respectively $\hat{\mathbf{g}}$
$L_2(\Omega)$	Sobolev function space
$H_0^1(\Omega)$	Sobolev function space
$\mathcal{U}$	state space for the displacement, definition 2.2.10
$\mathcal{C}$	control space for the load, definition 2.2.15
$\mathcal{X}$	mixed space for the Lagrange function, definition 4.2.1

### Operators

$\nabla, \hat{\nabla}, \tilde{\nabla}$	gradients with respect to $\mathbf{x}, \hat{\mathbf{x}}, \tilde{\mathbf{x}}$
$\text{div}, \widehat{\text{div}}, \widetilde{\text{div}}$	divergence with respect to $\mathbf{x}, \hat{\mathbf{x}}, \tilde{\mathbf{x}}$
$(\cdot)_{,\mathbf{U}}$	derivative with respect to $\mathbf{U}$
$\varepsilon(\mathbf{U})$	linearized strain of the displacement $\mathbf{U}$ , eq. 2.48
$S$	solution operator, definition 4.3.1
$R$	Riesz operator for a Hilbert space, defined by (3.11)



## Tensor, Vector and Scalar Fields

$\boldsymbol{x}$	material coordinates of the undeformed domain $\Omega$ , section 2.1
$\widehat{\boldsymbol{x}}$	material coordinates of the deformed domain $\widehat{\Omega}$ , section 2.1
$\widetilde{\boldsymbol{x}}$	parameter of the parameter space $\widetilde{\Omega}$ , section 2.4
$\boldsymbol{U}$	displacement, section 2.1
$\boldsymbol{U}_i^{2D}$	part of the displacement for the plate model, section 2.5
$\delta \boldsymbol{U}$	perturbation of the displacement $\boldsymbol{U}$
$\Delta \boldsymbol{U}$	search direction for the displacement $\boldsymbol{U}$
$\boldsymbol{F}$	deformation gradient, section 2.1
$\widetilde{\boldsymbol{G}}$	Jacobi matrix, (2.52)
$J$	local volume change, section 2.1
$\boldsymbol{C}$	right Cauchy-Green tensor ( $\boldsymbol{C} = \boldsymbol{F}^\top \boldsymbol{F}$ )
$\boldsymbol{T}, \widehat{\boldsymbol{T}}$	Cauchy stress tensor, section 2.2
$\boldsymbol{f}$	volume load, section 2.2.3
$\boldsymbol{g}$	boundary load, section 2.2.3
$t$	inner pressure, section 2.2.3
$m$	tension along a fiber, section 2.2.3
$\boldsymbol{a}$	fiber direction, section 2.2.3
$\boldsymbol{C}$	control, section 2.2.3
$\delta \boldsymbol{C}$	perturbation of the control $\boldsymbol{C}$
$\Delta \boldsymbol{C}$	search direction for the control $\boldsymbol{C}$
$\boldsymbol{Z}$	multiplier in the Lagrange functional (4.18) or adjoint state in the adjoint equation 4.37
$\boldsymbol{X}$	abbreviation for the triple $(\boldsymbol{U}, \boldsymbol{C}, \boldsymbol{Z})$
$s, y$	auxiliary functions (4.42) for the BFGS update



**Functionals**

$G$	guiding function (3.5) for the guiding criterion (3.4)
$I$	objective functional of the optimal control problem (4.2)
$I^{\text{red}}$	reduced objective functional of the reduced optimal control problem (4.3)
$L(\cdot, \cdot, \cdot)$	Lagrange functional (4.18)
$P$	penalty or cost functional (4.17) for the control
$Q$	quality functional for the state
$Q^{\text{track}}$	tracking-type quality functional (4.4)
$Q^{\text{pen}}$	regional penalization (4.6)
$Q^s$	quality functional (4.9) for the desired direction
$Q^V$	enclosed volume between two plates
$w$	energy density (2.18)
$W$	stored energy (2.45)
$W^{(T)}$	energy (2.32) for the stress
$W^{(C)}$	energy for the control
$W^{(f)}$	energy for the volume load, table 2.1
$W^{(g)}$	energy for the boundary load, table 2.1
$W^{(t)}$	energy for the inner pressure, table 2.1
$W^{(m)}$	energy for the fiber tension, table 2.1

**Matrices and Vectors**

$A_h$	stiffness matrix from the nonlinear model of elasticity
$K_{\text{lin.elast}}$	stiffness matrix from the linear model of elasticity
$M_C$	mass matrix in the control space
$P$	preconditioner
$p_{\ell \rightarrow \ell+1}$	prolongation (3.24)

**Parameters**

$\lambda, \mu$	Lamé parameters
$a, b, c, d, e$	material parameters for the polyconvex energy density (2.18)
$2d$	thickness of a plate
$D^{2D}$	degree of the hierarchical plate model
$p_i$	Legendre polynomial of degree $i$ , remark 2.5.3
$D^{\text{GL}}$	degree of the Gauss-Legendre quadrature formula, remark 2.5.5
$x_j \omega_j$	integration point and its weight for the Gauss-Legendre quadrature, remark 2.5.5
aTol	absolute tolerance in a stopping criterion
rTol	relative tolerance in a stopping criterion
$s_C$	scaling for the initial hessian (4.43) of the BFGS update
$s_U, s_C, s_Z$	scaling factors of the block preconditioner (4.27)
$L$	number of mesh refinements



## Norms and Inner Products

$\ \cdot\ _2$	Euclidean norm of a vector
$\ \cdot\ _F$	Frobenius norm of a matrix
$\ \cdot\ _{\mathcal{U}^*}$	norm in the dual state space $\mathcal{U}^*$
$\ \cdot\ _{\mathcal{C}^*}$	norm in the dual control space $\mathcal{C}^*$
$\ \cdot\ _{\mathcal{X}^*}$	norm in the dual space $\mathcal{X}^*$
$(\cdot, \cdot)_{\mathcal{U}}$	inner product of the state space $\mathcal{U}$
$(\cdot, \cdot)_{\mathcal{C}}$	inner product of the control space $\mathcal{C}$
$[\cdot, \cdot]_{\mathcal{U}^*, \mathcal{U}}$	duality pairing for the state space $\mathcal{U}$ and its dual $\mathcal{U}^*$
$[\cdot, \cdot]_{\mathcal{C}^*, \mathcal{C}}$	duality pairing for the control space $\mathcal{U}$ and its dual $\mathcal{U}^*$

## Abbreviations

AD	automatic/algorithmic differentiation
BC	boundary condition
BFGS	Broyden-Fletcher-Goldfarb-Shanno (update formula)
CG	Conjugate Gradient (method)
dofs	degrees of freedom
FEM	finite element method
FFC	FENICS Form Compiler
LM-BFGS	limited Broyden-Fletcher-Goldfarb-Shanno (update formula)
LN	Lagrange-Newton (method)
LNnest	nested Lagrange-Newton (method)
MINRES	Minimal Residual (method)
NURBS	non-uniform rational B-splines
QN	quasi-Newton (method)
PDE	partial differential equation
UFL	Unified Form Language



## 2 A Mathematical Model of Elasticity with Finite Deformations

### Contents

---

2.1	Modeling of Finite Deformations	12
2.2	Models for Elastic Material Behavior	14
2.2.1.	Balance-of-Forces Approach	14
2.2.2.	Energy Minimization Approach	18
2.2.3.	Loads (Volume, Boundary, Inner Pressure, Fiber Tension)	23
2.3	Comparison to Linear Elasticity	32
2.4	Parametrization of the Reference Configuration $\Omega$	36
2.5	Hierarchical Plate Model	38

---

In this chapter we give an introduction to the mathematical modeling of elasticity for large deformations. Our object of interest is an elastic body which is deformed by exterior or interior loads. First, in section 2.1, we describe the geometry of the deformation, which can be achieved by the displacement in the case of stationary elastic material behavior. Based on the displacement, we can formulate geometrical quantities such as the deformation gradient  $F$  or the local volume change  $J$ . In section 2.2, we will show the classical derivation of the model of elasticity by starting with the balance-of-forces. In this derivation, a stress tensor shows up and we need to establish a relationship between the displacement and this stress tensor. In case of hyper-elastic materials, this relationship is modeled by an energy density  $w$ . We will see that this energy density  $w$  can also be used to take a different approach than the balance-of-forces: the energy minimization. Here we will formulate a stored energy, which is a measure for the work which is done to deform the body. The formal first order optimality condition of the energy minimization problem will be the same variational equality as in the balance-of-forces approach. The minimization of the stored energy is also an essential part of the existence result due to John Ball, who proved the existence of a displacement for polyconvex energy densities.

In section 2.2.3, we will give some more details on the modeling of the loads. The two classical types, volume and boundary loads, can be found in almost every book on elasticity, though the assumptions for these loads are often not stated explicitly.



After representing the derivation of these two external loads, we show how two internal loads can be modeled: an inner pressure and a tension along a fiber tension, both motivated by biological phenomena. To put these two internal loads into the context of the energy minimization, we will state energy functionals that correspond to these loads, showing that the loads are conservative.

We will conclude the derivation of the elasticity model with a comparison to the linear model of elasticity in section 2.3. We will see the two main reasons why the linear model is valid only for small strains and cannot be applied for large deformations.

Sometimes it is easier to describe the undeformed body in curvilinear coordinates, which we will discuss in section 2.4. Our prior notation for the elasticity model allows an easy inclusion of such curvilinear coordinates into the variational equality. This also holds true for the hierarchical plate model in section 2.5. Here we present a plate model that can be understood as a reduction technique, that is we assume that the displacement has a certain structure. This assumption turns out to be a sound one for thin plates.

## 2.1 Modeling of Finite Deformations

In this work, the derivation of the mathematical problem follows Ciarlet (1988). The object of interest is an undeformed mechanical body which occupies the domain  $\Omega \subset \mathbb{R}^3$ , that means  $\Omega$  is open, connected and has a Lipschitz boundary. This body is now deformed due to an internal or external force. That means that each material point  $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \in \Omega$  of the undeformed body is moved to a new position  $\hat{\mathbf{x}}(\mathbf{x})$  of the deformed body  $\hat{\Omega} = \hat{\mathbf{x}}(\Omega) \subset \mathbb{R}^3$ , with a function  $\hat{\mathbf{x}} : \Omega \rightarrow \hat{\Omega}$ , see figure 2.1. We assume the function  $\hat{\mathbf{x}}$  to be injective, otherwise two material particles would occupy the same spot, so there exists an inverse function  $\hat{\mathbf{x}}^{-1} : \hat{\Omega} \rightarrow \Omega$ . The undeformed body  $\Omega$  is often called *reference configuration* and the deformed body  $\hat{\Omega}$  *current configuration*.

Functions to describe physical quantities (e.g. the material density) can be defined on either  $\Omega$  or  $\hat{\Omega}$ , depending on whether one uses the Eulerian or the Lagrangian description. In this work, functions defined on the current configuration  $\hat{\Omega}$  are marked with a hat  $\hat{\cdot}$ , not only to emphasize their dependencies, but also to clarify the argument in case we drop the argument ( $\hat{\mathbf{x}}$ ) for better readability.

A common way to describe a deformation is the *displacement* field  $\mathbf{U} : \Omega \rightarrow \mathbb{R}^3$  which is the difference

$$\mathbf{U}(\mathbf{x}) := \hat{\mathbf{x}}(\mathbf{x}) - \mathbf{x}. \quad (2.1)$$

This quantity is sufficient to describe an elastic deformation  $\hat{\mathbf{x}}(\mathbf{x}) = \mathbf{x} + \mathbf{U}(\mathbf{x})$  in our framework (unlike visco-elasticity or plasticity where additional quantities are required to describe and evolve the current deformation).



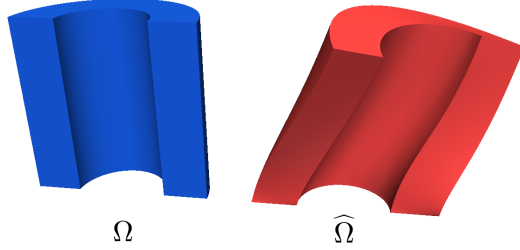


FIGURE 2.1. The undeformed body  $\Omega$  (blue) and the deformed body  $\widehat{\Omega}$  (red).

In later formulations, the *deformation gradient*  $F$  plays a central role. It is defined as the Jacobian (matrix)<sup>1</sup>

$$F(\mathbf{x}) := \frac{\partial \widehat{\mathbf{x}}}{\partial \mathbf{x}}(\mathbf{x}), \quad [F]_{ij} = \frac{\partial \widehat{x}_i}{\partial x_j}, \quad i, j = 1, 2, 3, \quad (2.2)$$

and its inverse matrix is

$$F^{-1}(\widehat{\mathbf{x}}) := \frac{\partial \mathbf{x}}{\partial \widehat{\mathbf{x}}}(\widehat{\mathbf{x}}), \quad [F^{-1}]_{ij} = \frac{\partial x_i}{\partial \widehat{x}_j}, \quad i, j = 1, 2, 3. \quad (2.3)$$

It is easy to see that it is the inverse matrix of  $F$ , e. g.

$$[F^{-1}F]_{ij} = \sum_k [F^{-1}]_{ik} [F]_{kj} = \sum_k \frac{\partial x_i}{\partial \widehat{x}_k} \frac{\partial \widehat{x}_k}{\partial x_j} = \delta_{ij}.$$

In order to reformulate  $F$  in terms of the displacement  $\mathbf{U}$ , we introduce the differential operators<sup>2</sup>

$$[\nabla \mathbf{V}]_{ij}(\mathbf{x}) := \frac{\partial V_i}{\partial x_j}(\mathbf{x}), \quad i, j = 1, 2, 3, \quad (2.4)$$

and

$$[\widehat{\nabla} \widehat{\mathbf{V}}]_{ij}(\widehat{\mathbf{x}}) := \frac{\partial \widehat{V}_i}{\partial \widehat{x}_j}(\widehat{\mathbf{x}}), \quad i, j = 1, 2, 3, \quad (2.5)$$

<sup>1</sup>Despite its name, the deformation gradient  $F$  should not be viewed as a gradient in the classical sense. It is the derivative of  $\widehat{\mathbf{x}}$  with respect to  $\mathbf{x}$ .

<sup>2</sup>Another common notation, for example see Meyer (2007), Bertram (2008) or Ogden (1984), is built around the gradient operator  $\text{Grad}(\cdot)$  which is the transpose of the above definition  $\nabla(\cdot)$ . This usually accompanies the definition “ $A : B := \text{tr}(AB)$ ”.



for differentiable functions  $\mathbf{V} : \Omega \rightarrow \mathbb{R}^3$  and  $\widehat{\mathbf{V}} : \widehat{\Omega} \rightarrow \mathbb{R}^3$ . Applying the chain rule yields the relationship

$$[\widehat{\nabla}\widehat{\mathbf{V}}]_{ij}(\widehat{\mathbf{x}}(\mathbf{x})) := \sum_{k=1}^3 \frac{\partial V_i}{\partial x_k} \frac{\partial x_k}{\partial \widehat{x}_j}(\mathbf{x}), \quad i, j = 1, 2, 3,$$

or, in terms of matrices,

$$\widehat{\nabla}\widehat{\mathbf{V}}(\widehat{\mathbf{x}}(\mathbf{x})) = \nabla \mathbf{V}(\mathbf{x}) F^{-1}(\mathbf{x}). \quad (2.6)$$

With this notation,  $F$  can be written in the form

$$F(\mathbf{x}) = \frac{\partial(\text{id} + \mathbf{U})}{\partial \mathbf{x}}(\mathbf{x}) = I + \nabla \mathbf{U}(\mathbf{x}), \quad (2.7)$$

with the identity matrix  $I \in \mathbb{M}^3$ . Another frequently met quantity is the *local volume change*

$$J(\mathbf{x}) := \det F(\mathbf{x}) = \det(I + \nabla \mathbf{U}(\mathbf{x})). \quad (2.8)$$

We remark that the local volume change  $J(\mathbf{x})$  should be positive for all  $\mathbf{x} \in \Omega$ . Otherwise, the deformed body is not guaranteed to be physically admissible and the following modeling cannot be justified. *A positive local volume change is essential to the modeling of elasticity.* A further discussion on this topic can be found in section 2.3.

## 2.2 Models for Elastic Material Behavior

The classical displacement-traction problem is described by an undeformed body  $\Omega$  which is clamped on a part  $\Gamma^D \subset \partial\Omega$  of the boundary of the body. Due to boundary loads  $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^3$  acting on the remaining part  $\Gamma^N = \partial\Omega \setminus \Gamma^D$  of the boundary and volume loads  $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^3$  acting on the whole body, the reference configuration  $\Omega$  is deformed into the current configuration  $\widehat{\Omega}$ . In this section, a brief description of the modeling of elastic material behavior of the body is given. In this work, we assume that the material behavior is isotrop and homogeneous, though extension to anisotropic material behavior is possible in principle. We will start with the classical modeling based on a balance of forces. For more details, see Ciarlet (1988), Ogden (1984), Antman (1984), Bertram (2008), Bower (2009) or Bower (2012).

### 2.2.1. Balance-of-Forces Approach

A classical principle in mechanics, namely Newton's law, states that if a body is at rest, all forces and stresses must be balanced. That means that for each force there is an equivalent counter force. As we assume our problem to be time independent or stationary, the deformed body has to be at rest. This means that the balance of forces is formulated in the deformed body, as this is the current configuration where the body rests. Let us start with a small arbitrary control volume  $\widehat{\omega} \subset \text{int } \widehat{\Omega}$ . The



balance of forces consists of two parts: forces over the boundary and the volume loads acting on the control volume. The forces over the boundary  $\partial\hat{\omega}$  can be described by the so-called *Cauchy stress tensor*  $\hat{T} \in \mathbb{M}^3$ .

**Theorem 2.2.1** (Cauchy's theorem). Let  $\hat{\mathbf{n}}(\hat{\mathbf{x}})$  be the outer normal of  $\hat{\omega}$  at  $\hat{\mathbf{x}} \in \partial\hat{\omega}$ . There exists a symmetric tensor field  $\hat{T} : \hat{\Omega} \rightarrow \mathbb{M}^3$  (independent of the choice of  $\hat{\omega}$ ) such that the force at  $\hat{\mathbf{x}} \in \partial\hat{\omega}$  can be written as  $\hat{T}(\hat{\mathbf{x}})\hat{\mathbf{n}}(\hat{\mathbf{x}})$ .

*Proof.*

For a proof, see (Ciarlet, 1988, Th. 2.3-1) or (Antman, 1984, Th. 7.14).

The resulting force due to stresses along the boundary  $\partial\hat{\omega}$  can then be written as

$$\int_{\partial\hat{\omega}} \hat{T}(\hat{\mathbf{x}})\hat{\mathbf{n}}(\hat{\mathbf{x}}) \, d\partial\hat{\omega} = \int_{\hat{\omega}} \widehat{\text{div}}\hat{T}(\hat{\mathbf{x}}) \, d\hat{\mathbf{x}},$$

after applying Green's first identity from lemma A.2.4.

To simplify notation, we assume without loss of generality that the volume load  $\hat{\mathbf{f}} : \hat{\Omega} \rightarrow \mathbb{R}^3$  acts on the volume (e.g. Newton per cubic meter) rather than acting on the material mass (Newton per gram in the case of gravity). The latter can be reformulated as a load acting on the volume (e.g. by multiplying with the material density  $\rho$  in the case of gravity). However, one should keep this in mind as it is essential for transformations from  $\hat{\mathbf{f}} : \hat{\Omega} \rightarrow \mathbb{R}^3$  to  $\mathbf{f} : \Omega \rightarrow \mathbb{R}^3$ .

Combining inner stresses  $\hat{T}$  and volume loads  $\hat{\mathbf{f}}$  results in the equation

$$\int_{\hat{\omega}} \widehat{\text{div}}\hat{T}(\hat{\mathbf{x}}) + \hat{\mathbf{f}}(\hat{\mathbf{x}}) \, d\hat{\mathbf{x}} = \mathbf{0}.$$

Since this equation has to hold for all subdomains  $\hat{\omega} \subset \hat{\Omega}$  and we assume the continuum hypothesis, see (Salenço, 2001, ch. 1), the first part of the balance of forces reads

$$\widehat{\text{div}}\hat{T}(\hat{\mathbf{x}}) + \hat{\mathbf{f}}(\hat{\mathbf{x}}) = \mathbf{0} \quad \text{in } \hat{\Omega}. \quad (2.9)$$

Another type of forces are boundary forces  $\hat{\mathbf{g}} : \hat{\Gamma}^N \rightarrow \mathbb{R}^3$ . As they act on the surface of the deformed body and counter-act the stresses  $\hat{T}\hat{\mathbf{n}}$  along the surface, the second part of balance of forces directly reads

$$\hat{T}(\hat{\mathbf{x}})\hat{\mathbf{n}}(\hat{\mathbf{x}}) = \hat{\mathbf{g}}(\hat{\mathbf{x}}) \quad \text{on } \hat{\Gamma}^N. \quad (2.10)$$

The last part is the clamping on  $\Gamma^D$ , that is

$$\mathbf{U}(\mathbf{x}) = \mathbf{0} \quad \text{on } \Gamma^D. \quad (2.11)$$



Combining the above equations (2.9), (2.10) and (2.11), the balance of forces reads

$$\begin{aligned} \widehat{\text{div}}\widehat{T}(\widehat{\mathbf{x}}) + \widehat{\mathbf{f}}(\widehat{\mathbf{x}}) &= \mathbf{0} && \text{in } \widehat{\Omega} \\ \widehat{T}(\widehat{\mathbf{x}})\widehat{\mathbf{n}}(\widehat{\mathbf{x}}) &= \widehat{\mathbf{g}}(\widehat{\mathbf{x}}) && \text{on } \widehat{\Gamma}^N \\ \mathbf{U}(\mathbf{x}) &= \mathbf{0} && \text{on } \Gamma^D. \end{aligned} \quad (2.12)$$

Multiplying with a test function  $\widehat{\mathbf{V}} : \widehat{\Omega} \rightarrow \mathbb{R}^3$  and applying Green's identity yields the weak formulation

$$\int_{\widehat{\Omega}} \widehat{T} : \widehat{\nabla} \widehat{\mathbf{V}} \, d\widehat{\mathbf{x}} = \int_{\widehat{\Omega}} \widehat{\mathbf{f}}^\top \widehat{\mathbf{V}} \, d\widehat{\mathbf{x}} + \int_{\widehat{\Gamma}^N} \widehat{\mathbf{g}}^\top \widehat{\mathbf{V}} \, d\widehat{S} \quad \forall \widehat{\mathbf{V}} \text{ with } \widehat{\mathbf{V}}|_{\widehat{\Gamma}^D} = \mathbf{0}, \quad (2.13)$$

where the matrix inner product is denoted by  $A : B := \text{tr}(A^\top B)$ . This weak formulation cannot be directly used to solve the problem, due to two reasons:

- (1) The unknown displacement  $\mathbf{U}$  determines the deformation  $\widehat{\mathbf{x}} = \mathbf{x} + \mathbf{U}(\mathbf{x})$ , hence the current configuration  $\widehat{\Omega}$  and therefore all functions in the equation (2.13) depend on  $\mathbf{U}$ , even the differential operator  $\widehat{\nabla}$  (and  $\widehat{\text{div}}$ ).
- (2) The Cauchy stress tensor  $\widehat{T}$  depends on the unknown displacement  $\mathbf{U}$ . A relationship between those two quantities has to be established.

To address the first problem, the weak formulation (2.13) is transformed onto the reference configuration  $\Omega$ :

- A volume element  $d\widehat{\mathbf{x}}$  is transformed by  $d\widehat{\mathbf{x}} = \det F \, d\mathbf{x}$ .
- We recall the equation (2.6) for the differential operator  $\widehat{\nabla} \widehat{\mathbf{V}} = \nabla \mathbf{V} F^{-1}$ .
- The stress tensor  $\widehat{T}$  can be transformed by the *Piola transformation*

$$\begin{aligned} T(\mathbf{x}) &= (\det F(\mathbf{x})) \widehat{T}(\widehat{\mathbf{x}}(\mathbf{x})) F^{-\top}(\mathbf{x}) \\ \text{or} \quad \widehat{T}(\widehat{\mathbf{x}}(\mathbf{x})) &= (\det F(\mathbf{x}))^{-1} T(\mathbf{x}) F^\top(\mathbf{x}), \end{aligned} \quad (2.14)$$

see (Ciarlet, 1988, th. 1.7-1) The new stress tensor  $T$  is called *first Piola-Kirchhoff stress tensor*.

- The test function  $\widehat{\mathbf{V}}$  defined on the current configuration is transformed onto the reference configuration by setting  $\mathbf{V}(\mathbf{x}) := \widehat{\mathbf{V}}(\widehat{\mathbf{x}}(\mathbf{x}))$
- The transformation of the loads  $\widehat{\mathbf{f}}$  and  $\widehat{\mathbf{g}}$  will be done in section 2.2.3.



Combining the above yields the transformation of the weak formulation

$$\begin{aligned}
& \int_{\hat{\Omega}} \hat{T} : \hat{\nabla} \hat{\mathbf{V}} \, d\hat{\mathbf{x}} \\
&= \int_{\Omega} \operatorname{tr} \left( (\det F)^{-1} F T^{\top} \nabla \mathbf{V} F^{-1} \right) (\det F) \, d\mathbf{x} \\
&= \int_{\Omega} \operatorname{tr} \left( T^{\top} \nabla \mathbf{V} \right) \, d\mathbf{x} = \int_{\Omega} T : \nabla \mathbf{V} \, d\mathbf{x} \\
&\text{and} \\
& \int_{\hat{\Omega}} \hat{\mathbf{f}}^{\top} \hat{\mathbf{V}} \, d\hat{\mathbf{x}} + \int_{\hat{\Gamma}^N} \hat{\mathbf{g}}^{\top} \hat{\mathbf{V}} \, d\hat{S} = \int_{\Omega} \mathbf{f}^{\top} \mathbf{V} \, d\mathbf{x} + \int_{\Gamma^N} \mathbf{g}^{\top} \mathbf{V} \, dS.
\end{aligned}$$

As a result, the weak formulation of the balance of forces in the reference configuration reads

$$\begin{aligned}
\int_{\Omega} T(\mathbf{x}) : \nabla \delta \mathbf{U}(\mathbf{x}) \, d\mathbf{x} &= \int_{\Omega} \mathbf{f}(\mathbf{x})^{\top} \delta \mathbf{U}(\mathbf{x}) \, d\mathbf{x} + \int_{\Gamma^N} \mathbf{g}(\mathbf{x})^{\top} \delta \mathbf{U}(\mathbf{x}) \, dS \\
&\quad \forall \delta \mathbf{U} \text{ with } \delta \mathbf{U}|_{\Gamma^D} = 0.
\end{aligned} \tag{2.15}$$

The second problem is to determine a relationship between the first Piola-Kirchhoff stress tensor  $T$  and the displacement  $\mathbf{U}$ . In general, this can be done by a so-called *response function*, which models the relation between the stress and the strain. For example, in the case of linear elasticity with infinitesimally small deformations, the stress-strain-relation is linear and can be well justified by various principles of continuum mechanics. In the case of large deformations however, this relation is far more complicated. So far, various phenomenological models have been proposed to describe the elastic behavior of materials.

In this work, we consider a *continuous, homogeneous and isotropic elastic* material behavior that is modeled by a so-called *polyconvex* stored-energy density  $w$  depending on the deformation gradient  $F$ . As for now, it should be sufficient to state that the first Piola-Kirchhoff stress tensor  $T$  can be written as

$$[T(\mathbf{x})]_{ij} = \frac{\partial w}{\partial [F]_{ij}}(F(\mathbf{x})) \quad i, j = 1, 2, 3, \tag{2.16}$$

for a given energy density  $w : \mathbb{M}_+^3 \rightarrow \mathbb{R}$ . Inserting this into the variational equation (2.15) yields a nonlinear partial differential equation of second order for the unknown  $\mathbf{U}$ . More details about  $w$  will be given in the next section 2.2.2.



### 2.2.2. Energy Minimization Approach

An alternative approach to the previous derivation is the minimization of the so-called stored energy. It is based on the observation that the current configuration minimizes (locally) a stored energy functional  $W$  (if it exists) depending on the displacement  $\mathbf{U}$ . This means our deformation problem with the displacement  $\mathbf{U}$  can be written as an optimization problem

$$\min_{\mathbf{U} \in \mathcal{U}} W(\mathbf{U}, \mathbf{C}), \quad (2.17)$$

with a function space  $\mathcal{U}$  and a load  $\mathbf{C}$ , both to be defined later. The computation of  $W$  depends on a so-called *energy density*  $w$  which characterizes the material behavior. One can find several proposals for such densities in the literature, for example see Bertram (2008), Antman (1984) and Ciarlet (1988) for an overview, as well as Ogden (1984) or Simo and Hughes (1998), and references therein. We choose our energy density to be a function  $w : \mathbb{M}_+^3 \rightarrow \mathbb{R}$  such that

$$w(F) = a \|\mathbf{F}\|_{\mathbb{F}}^2 + b \|\operatorname{cof} F\|_{\mathbb{F}}^2 + c(\det F)^2 - d \ln(\det F) + e, \quad (2.18)$$

with the Frobenius norm  $\|\cdot\|_{\mathbb{F}}$  and certain material constants  $a, b, c, d > 0$ ,  $e \in \mathbb{R}$ . This is a special case of Ogden's materials from Ogden (1972) or a compressible Moonley-Rivlin material, see Ciarlet and Geymonat (1982). The constant  $e$  is chosen such that the undeformed body has no energy, that is  $w(I) = 0$ .

**Remark 2.2.2** (Equivalent formulations of the energy density  $w$ ). The energy density  $w$  can be expressed with respect to various quantities. Most formulations can be transformed into others by a direct calculation. However, calculating derivatives might be easier in certain formulations. We give a few popular choices on which the energy density might depend:

- The energy density  $w$  from (2.18) can also be written in terms of the strain tensor  $\mathbf{E} := \frac{1}{2}(\nabla \mathbf{U} + \nabla \mathbf{U}^\top + \nabla \mathbf{U}^\top \nabla \mathbf{U})$ , e.g.
 
$$\bar{w}(\mathbf{E}) = \bar{a} \operatorname{tr}(\mathbf{E}) + \bar{b}_1 (\operatorname{tr} \mathbf{E})^2 + \bar{b}_2 \operatorname{tr}(\mathbf{E}^2) + c(\det F)^2 - d \ln(\det F) + e,$$
 with  $b = -\frac{\bar{b}_2}{2}$  and  $a = \frac{\bar{a}}{2} - \bar{b}_2$ , see Weiser et al. (2007).
- Another formulation uses the invariants of the right Cauchy-Green tensor  $\mathbf{C} := \mathbf{F}^\top \mathbf{F}$ , e.g.  $I_1(\mathbf{C}) = \operatorname{tr} \mathbf{C}$ ,  $I_2(\mathbf{C}) = \operatorname{tr} \operatorname{cof} \mathbf{C}$  and  $I_3(\mathbf{C}) = \det \mathbf{C}$  or directly the eigenvalues of  $\mathbf{C}$ , see (Ciarlet, 1988, ch. 4.10), Bertram (2008) or Antman (1984). It is also possible to use  $\mathbf{C}$  itself to write down the energy density.
- Some authors, like Meyer (2007), prefer so-called pseudo-invariants  $\frac{1}{i} \operatorname{tr}(\mathbf{C}^i)$ ,  $i = 1, 2, 3$ . In this formulation, the calculation of derivatives with respect to  $\mathbf{F}$  might be easier than in most formulations.



This energy density has the following properties which are needed not only for an existence result in Ball (1977) but they also satisfy certain observations from real experiments. Our first axiom is the frame indifference of  $w$ , that means the value of the energy does not depend on the choice of the coordinate system.

**Theorem 2.2.3** (Frame indifference). The function  $w$  from (2.18) is frame-indifferent, that means that the function value  $w(F)$  is indifferent to any rotation of  $F$  by a matrix  $Q \in \mathbb{M}_+^3$  with  $Q^\top Q = I$ , i. e.

$$w(F) = w(QF) \quad \forall Q \in \mathbb{M}_+^3 \text{ with } Q^\top Q = I. \quad (2.19)$$

*Proof.*

We have the following:

- (1)  $\det Q = 1$  because of  $Q^\top Q = I$  and  $Q \in \mathbb{M}_+^3$
- (2) the frame indifference of the Frobenius norm, that is  
 $\|QF\|_F^2 = \text{tr}(F^\top Q^\top Q F) = \text{tr}(F^\top F) = \|F\|_F^2$
- (3)  $\text{cof}(QF) = \text{cof } Q \text{ cof } F = Q \text{ cof } F$  because  $\text{cof } Q = Q^{-\top} = Q$ .

Using these properties yields

$$\begin{aligned} w(QF) &= a\|QF\|_F^2 + b\|\text{cof}(QF)\|_F^2 + c(\det(QF))^2 - d \ln(\det(QF)) + e \\ &= a\|F\|_F^2 + b\|Q \text{ cof } F\|_F^2 + c(\det F)^2 - d \ln(\det F) + e \\ &= w(F). \end{aligned}$$

The next assumption comes from the observation that you cannot shrink a body to have no volume. This is modeled by an energy density diverging to  $\infty$  as the volume tends to zero. From the point of view of constrained optimization, this property can also be interpreted as a barrier functional for the constraint  $J = \det F > 0$ . This property is very important because otherwise negative local volume change might occur which results in physically inadmissible deformations.

**Theorem 2.2.4** (Behavior for shrinking volume). The function  $w$  from (2.18) satisfies

$$\lim_{\det F \rightarrow 0} w(F) = \infty. \quad (2.20)$$

*Proof.* Obviously, the last term in  $w$ , that is  $-d \ln \det F$ , diverges to  $\infty$  as  $\det F \rightarrow 0$  since  $d > 0$ . As the other terms are nonnegative for  $a, b, c \geq 0$ , the sum diverges to  $\infty$ .



The assumption (2.20) implies that  $w$  cannot be convex with respect to  $F$ , see (Ciarlet, 1988, ch. 4.8) for a proof. Therefore it is replaced by a weaker requirement of polyconvexity, which is required later for Ball's existence result in theorem 2.2.9.

**Theorem 2.2.5** (Polyconvexity of the energy density  $w$ ). The function  $w$  from (2.18) is polyconvex, that means there exists a convex function  $\bar{w} : \mathbb{M}_+^3 \times \mathbb{M}_+^3 \times (0, \infty) \rightarrow \mathbb{R}$  such that

$$w(F) = \bar{w}(F, \operatorname{cof} F, \det F), \quad \forall F \in \mathbb{M}_+^3. \quad (2.21)$$

*Proof.* We choose

$$\bar{w}(F, H, J) = a\|F\|_{\mathbb{F}}^2 + b\|H\|_{\mathbb{F}}^2 + cJ^2 - d \ln J + e, \quad F, H \in \mathbb{M}_+^3, J \in (0, \infty).$$

The Frobenius norm  $\|\cdot\|_{\mathbb{F}}$  is a convex function for all matrices because of the triangle inequality, therefore the first two terms are convex for  $a, b \geq 0$ . The last two terms are convex because their second derivative  $2c + dJ^{-2}$  is positive for  $c, d > 0$  and  $J > 0$ .

The next assumption states that at the undeformed state  $\mathbf{U} \equiv \mathbf{0}$ , the energy density  $w$  matches the energy density belonging to the linear, isotropic elasticity model for infinitesimally small strains.

**Theorem 2.2.6** (Behavior for small deformations). The energy density  $w$  from (2.18) behaves like the St. Venant-Kirchhoff energy density around the undeformed state  $\mathbf{U}$ , that is

$$w(F) = \frac{\lambda}{2} (\operatorname{tr} E)^2 + \mu \operatorname{tr}(E^2) + \mathcal{O}(\|E\|_{\mathbb{F}}^3), \quad E = \frac{1}{2}(F^\top F - I), \quad (2.22)$$

with the Lamé material parameters  $\lambda, \mu > 0$ .

*Proof.* We refer to the proof (Ciarlet, 1988, Th. 4.10-2), which also shows the relation of the constants  $a, b, c, d, e$  and the Lamé constants  $\lambda, \mu$ .

**Remark 2.2.7** (Relation between  $a, b, c, d, e$  and  $\lambda, \mu$ ). The proof of (Ciarlet, 1988, Th. 4.10-2) shows how the material parameters  $a, b, c, d, e$  have to be chosen with respect to the Lamé constants  $\lambda, \mu$ , though it is not an injective relation. A possible choice is

$$a = \frac{\mu}{2} - \frac{\lambda}{8}, \quad b = \frac{\lambda}{8}, \quad c = \frac{\lambda}{8}, \quad d = \mu + \frac{\lambda}{2}, \quad e = -\frac{3\mu}{2} - \frac{\lambda}{8}. \quad (2.23)$$

The last requirement for Ball's existence result is a certain coercivity inequality which describes how much the energy density should increase for large deformations.



**Theorem 2.2.8** (Coercivity inequality). The energy density  $w$  from (2.18) satisfies the inequality

$$w(F) \geq c_1 (\|F\|_F^p + \|\operatorname{cof} F\|_F^q + (\det F)^r) + c_2, \quad \forall F \in \mathbb{M}_+^3 \quad (2.24)$$

with constants  $c_1 > 0$ ,  $p \geq 2$ ,  $q \geq \frac{p}{p-1}$ ,  $r > 1$ ,  $c_2 \in \mathbb{R}$ .

*Proof.* Again, we refer to the proof given in (Ciarlet, 1988, Th. 4.10-2).

We are now ready to state the existence result from Ball (1977) for the energy density  $w$ , taken from (Ciarlet, 1988, ch. 7.7).

**Theorem 2.2.9** (Ball's Existence result for the displacement  $\mathbf{U}$ ). Let  $\Omega \subset \mathbb{R}^3$  be a domain and  $w : \mathbb{M}_+^3 \rightarrow \mathbb{R}$  be a stored energy density with the following properties:

- (1)  $w$  is polyconvex, see (2.21)
- (2)  $w \rightarrow \infty$  for  $J = \det F \rightarrow 0$ , see (2.20)
- (3)  $w$  is coercive, see (2.24)

Let furthermore  $\Gamma = \Gamma^D \cup \Gamma^N$  be a  $dS$ -measurable partition of the boundary  $\partial\Omega$  with an area  $|\Gamma^D| > 0$ , and let  $\mathbf{U}^D : \Gamma^D \rightarrow \mathbb{R}^3$  be a measurable function such that the set

$$\begin{aligned} \bar{\mathcal{U}} = \{ \mathbf{U} \in W^{1,p}(\Omega)^3; \operatorname{cof}(I + \nabla \mathbf{U}) \in L^q(\Omega)^{3 \times 3}, \det(I + \nabla \mathbf{U}) \in L^r(\Omega), \\ \mathbf{U} = \mathbf{U}^D \text{ a.e. on } \Gamma^D, \det(I + \nabla \mathbf{U}) > 0 \text{ a.e. in } \Omega \} \end{aligned} \quad (2.25)$$

is not empty. Let  $\mathbf{f} \in L^{p^3}(\Omega)^3$  and  $\mathbf{g} \in L^{p^4}(\Gamma^N)^3$  be such that the linear form

$$l(\mathbf{U}) = \int_{\Omega} \mathbf{f}(\mathbf{x})^\top \mathbf{U}(\mathbf{x}) \, d\mathbf{x} + \int_{\Gamma^N} \mathbf{g}(\mathbf{x})^\top \mathbf{U}(\mathbf{x}) \, d\Gamma$$

is bounded. The global energy is

$$W(\mathbf{U}) = \int_{\Omega} w(F(\mathbf{x})) \, d\mathbf{x} - l(\mathbf{U}),$$

and assume that  $\inf_{\mathbf{U} \in \bar{\mathcal{U}}} W(\mathbf{U}) < \infty$ . Then there exists at least one function  $\mathbf{U}_*$  such that

$$\mathbf{U}_* \in \bar{\mathcal{U}} \text{ and } W(\mathbf{U}_*) = \inf_{\mathbf{U} \in \bar{\mathcal{U}}} W(\mathbf{U}).$$

*Proof.* See the original source Ball (1977) or the proof in (Ciarlet, 1988, ch. 7.7).

Ball's existence result proves that there exists at least one solution, though it is possible that multiple solutions exist. If we have homogeneous Dirichlet boundary



conditions  $\mathbf{U}^D \equiv \mathbf{0}$ , the set  $\bar{\mathcal{U}}$  in the proof is nonempty because  $\mathbf{U} \equiv \mathbf{0} \in \bar{\mathcal{U}}$ . As we require a Hilbert space as a state space for our algorithms later, we modify the space (2.25) from Ball's proof. For this purpose, we choose the same space as in linear elasticity, that is the Sobolev space  $H^1(\Omega)^3$ . We keep the constraint  $J = \det F > 0$  implicitly by extending the definition of the energy density to  $w : \mathbb{M}^3 \rightarrow \mathbb{R} \cup \{\infty\}$  with

$$w(F) = \begin{cases} a\|\mathbf{F}\|_{\mathbb{F}}^2 + b\|\text{cof } \mathbf{F}\|_{\mathbb{F}}^2 + c(\det F)^2 - d\ln(\det F) + e & \text{for } \det F > 0 \\ \infty & \text{for } \det F \leq 0 \end{cases}. \quad (2.26)$$

**Definition 2.2.10** (State space). Corresponding to the linear model of elasticity, we choose the *state space*  $\mathcal{U}$  to be

$$\mathcal{U} = \left\{ \mathbf{U} \in H^1(\Omega)^3, \mathbf{U} = \mathbf{0} \text{ a.e. on } \Gamma^D, \right\}. \quad (2.27)$$

We choose the inner product  $(\cdot, \cdot)_{\mathcal{U}}$  to be induced by a linear elasticity model, that is for  $\mathbf{U}, \mathbf{V} \in \mathcal{U}$ ,

$$\begin{aligned} (\mathbf{U}, \mathbf{V})_{\mathcal{U}} &:= a_{\text{elast}}(\mathbf{U}, \mathbf{V}) \\ &= \int_{\Omega} \lambda \text{tr}(\nabla \mathbf{U}) \text{tr}(\nabla \mathbf{V}) + 2\mu(\varepsilon(\mathbf{U}) : \varepsilon(\nabla \mathbf{V})) \, d\mathbf{x}, \end{aligned} \quad (2.28)$$

with  $\varepsilon(\mathbf{U}) = \frac{1}{2}(\nabla \mathbf{U} + \nabla \mathbf{U}^{\top})$ .

Having an energy density, we can integrate it over the undeformed domain  $\Omega$  and add functionals for the volume load  $\mathbf{f}$  and boundary load  $\mathbf{g}$ , gaining the global *stored energy functional*

$$W(\mathbf{U}, \mathbf{f}, \mathbf{g}) = \int_{\Omega} w(F(\mathbf{x})) \, d\mathbf{x} - \int_{\Omega} \mathbf{f}(\mathbf{x})^{\top} \mathbf{U}(\mathbf{x}) \, d\mathbf{x} - \int_{\Gamma^N} \mathbf{g}(\mathbf{x})^{\top} \mathbf{U}(\mathbf{x}) \, dS, \quad (2.29)$$

with the deformation gradient  $F(\mathbf{x}) = \mathbf{I} + \nabla \mathbf{U}(\mathbf{x})$ . As we seek to minimize this energy, we consider the (formal) first order optimality condition of (2.17), which is the variational equation

$$\begin{aligned} 0 &= W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\delta \mathbf{U}] \quad \forall \delta \mathbf{U} \in \mathcal{U} \\ &= \int_{\Omega} w_{,F}(F(\mathbf{x}))[\nabla \delta \mathbf{U}(\mathbf{x})] \, d\mathbf{x} - \int_{\Omega} \mathbf{f}(\mathbf{x})^{\top} \delta \mathbf{U}(\mathbf{x}) \, d\mathbf{x} - \int_{\Gamma^N} \mathbf{g}(\mathbf{x})^{\top} \delta \mathbf{U}(\mathbf{x}) \, dS. \end{aligned} \quad (2.30)$$

Comparing this equation with the earlier weak formulation (2.15) derived from the balance of forces, one can see that the two formulations are equivalent if the relation



(2.16) between the first Piola-Kirchhoff stress tensor and the energy density  $w$  is rewritten as

$$T : \nabla \delta \mathbf{U} = w_{,F}(\mathbf{F}(\mathbf{x}))[\nabla \delta \mathbf{U}(\mathbf{x})]. \quad (2.31)$$

Hence, both approaches yield the same variational equation for the displacement-traction problem. For an easier use in later sections, we denote the energy due to the Cauchy stress with

$$W^{(T)}(\mathbf{U}) := \int_{\Omega} w(\mathbf{F}(\mathbf{x})) \, d\mathbf{x}, \quad (2.32)$$

and its derivative w. r. t.  $\mathbf{U}$  with

$$W_{,\mathbf{U}}^{(T)}(\mathbf{U})[\delta \mathbf{U}] := \int_{\Omega} w_{,F}(\mathbf{F}(\mathbf{x}))[\nabla \delta \mathbf{U}(\mathbf{x})] \, d\mathbf{x}. \quad (2.33)$$

The first derivative  $w_{,F}$  and the second derivative  $w_{,FF}$  can be found in the appendix as lemma A.2.1.

**Remark 2.2.11.**

- The stored energy is essential to the energy minimization and Ball's proof. However, for certain loads (e. g. pressure load on the boundary), it is still an open question whether an according functional can be added in the stored energy, and therefore if a stored energy exists for this load. We will only consider loads with energy functionals which are called *conservative* loads. Example loads will be given in the next section 2.2.3.
- The formulation as an optimization problem extends the methods to solve the problem. A discussion about this aspect can be found in section 3.2.
- The uniqueness of the solution  $\mathbf{U}$  cannot be ensured in elasticity with large deformation, see (Ciarlet, 1988, Sect. 5.8) for examples. The first order optimality conditions (2.30) can only ensure stationary points and there exists the possibility of multiple local minima of the stored energy. This might occur if a rod buckles or is twisted around its axis. In these examples, where the non-uniqueness of the state  $\mathbf{U}$  plays a central role, the problem should be altered to be instationary or incremental steps should be added (which could be understood as a pseudo time). However, we will not consider these examples in this thesis and assume that the method to solve the energy minimization returns a single solution.

### 2.2.3. Loads (Volume, Boundary, Inner Pressure, Fiber Tension)

In this section, we give more details on the modeling and transformation from  $\hat{\Omega}$  to  $\Omega$  of the classical volume load  $\mathbf{f}$  and the boundary load  $\mathbf{g}$ , which are both external forces. We then show the modeling and transformation of two internal forces: an



inner pressure  $t$  and a tension  $m$  along a given fiber direction  $\mathbf{a}$ . For each load, we will start on the current configuration because the load actually works on the deformed body. We then transform it to the reference configuration to implement it in our problem (2.15). Finally we give the respective energy functional that can be added to the stored energy (2.29).

### Volume Load $\mathbf{f}$ .

Forces like gravity can be modeled as a *volume load*. That is each material point  $\hat{\mathbf{x}} \in \hat{\Omega}$  on the deformed body has a vector  $\hat{\mathbf{f}}_m(\hat{\mathbf{x}})$  describing the force per mass, e. g. in the physical unit N/kg. Since  $\hat{\mathbf{f}}_m$  acts on the mass, we multiply it by the material density  $\hat{\rho} : \hat{\Omega} \rightarrow (0, \infty)$  to get the volume load  $\hat{\mathbf{f}} = \hat{\rho}\hat{\mathbf{f}}_m$  which acts on the volume instead of the mass. The resulting force due to  $\hat{\mathbf{f}}$  over a control volume  $\hat{\omega} \subset \hat{\Omega}$  then is

$$\int_{\hat{\omega}} \hat{\mathbf{f}}(\hat{\mathbf{x}}) d\hat{\mathbf{x}} = \int_{\hat{\omega}} \hat{\rho}(\hat{\mathbf{x}}) \hat{\mathbf{f}}_m(\hat{\mathbf{x}}) d\hat{\mathbf{x}}.$$

As we have seen in section 2.2.1, this equation has to be transformed to the reference configuration  $\Omega$  in order to get a PDE that can be solved numerically with a finite element method. First, we assume that the value of  $\hat{\mathbf{f}}_m(\hat{\mathbf{x}})$  at a material point does not change due to the deformation, that is  $\mathbf{f}_m(\mathbf{x}) := \hat{\mathbf{f}}_m(\hat{\mathbf{x}}(\mathbf{x}))$ <sup>3</sup> Second, the mass conservation states that due to the deformation from  $\omega$  to  $\hat{\omega}$ , the value of mass does not change. The mass of  $\hat{\omega}$  and  $\omega$  is

$$\int_{\hat{\omega}} \hat{\rho}(\hat{\mathbf{x}}) d\hat{\mathbf{x}} = \int_{\omega} \rho(\mathbf{x}) d\mathbf{x}.$$

Since the subdomain  $\hat{\omega}$  can be arbitrarily chosen, we get the relationship

$$\hat{\rho}(\hat{\mathbf{x}}(\mathbf{x})) d\hat{\mathbf{x}} = \rho(\mathbf{x}) d\mathbf{x} \quad \text{or} \quad \hat{\rho}(\hat{\mathbf{x}}(\mathbf{x})) \det F(\mathbf{x}) = \rho(\mathbf{x}). \quad (2.34)$$

Now we can transform the volume load  $\hat{\mathbf{f}}$  to  $\mathbf{f}$  by

$$\begin{aligned} \int_{\hat{\omega}} \hat{\mathbf{f}}(\hat{\mathbf{x}}) d\hat{\mathbf{x}} &= \int_{\hat{\omega}} \hat{\rho}(\hat{\mathbf{x}}) \hat{\mathbf{f}}_m(\hat{\mathbf{x}}) d\hat{\mathbf{x}} \\ &= \int_{\omega} \rho(\mathbf{x}) \mathbf{f}_m(\mathbf{x}) d\mathbf{x} = \int_{\omega} \mathbf{f}(\mathbf{x}) d\mathbf{x}, \end{aligned}$$

with  $\mathbf{f}(\mathbf{x}) = \rho(\mathbf{x}) \mathbf{f}_m(\mathbf{x})$ . To sum it up, we have

$$\mathbf{f}(\mathbf{x}) = \hat{\mathbf{f}}(\hat{\mathbf{x}}(\mathbf{x})), \quad (2.35)$$

<sup>3</sup>This would not hold true if one models a force due to e. g. a magnetic field which is not constant because the current force would depend on the current position of the material point.



due to the fact that the force acts on the mass rather than volume and the mass conservation. The incorporation of the volume load into the variational equation (2.15) was already shown. Also the energy functional

$$W^{(f)}(U, \mathbf{f}) = - \int_{\Omega} \mathbf{f}(\mathbf{x})^{\top} \mathbf{U}(\mathbf{x}) \, d\mathbf{x},$$

was mentioned previously in Ball's existence theorem 2.2.9. Its first derivative is

$$W_{,U}^{(f)}(U, \mathbf{f})[\delta U] = - \int_{\Omega} \mathbf{f}(\mathbf{x})^{\top} \delta \mathbf{U}(\mathbf{x}) \, d\mathbf{x},$$

and its second derivative is obviously zero.

### Boundary Load $g$ .

We model a boundary load which acts on the surface  $\widehat{\Gamma}^N$  of the deformed body. That is each material point  $\widehat{\mathbf{x}} \in \widehat{\Gamma}^N$  on the deformed body has a vector  $\widehat{\mathbf{g}}(\widehat{\mathbf{x}})$  describing the force per area, e. g. in the physical unit MPa = N/mm<sup>2</sup>. Let  $\widehat{\gamma} \subset \widehat{\Gamma}^N$  be a control area of the surface of  $\widehat{\Omega}$ . The resulting force acting on  $\widehat{\gamma}$  is

$$\int_{\widehat{\gamma}} \widehat{\mathbf{g}}(\widehat{\mathbf{x}}) \, d\widehat{S}.$$

We assume that the direction of this force does not change due to the deformation, that is  $\mathbf{g}(\mathbf{x})$  and  $\widehat{\mathbf{g}}(\widehat{\mathbf{x}}(\mathbf{x}))$  have the same direction<sup>4</sup> Additionally, we assume that the resulting force on  $\widehat{\gamma}$  does not change due to deformation<sup>5</sup> Combining those two assumptions we get the transformation

$$\int_{\widehat{\gamma}} \widehat{\mathbf{g}}(\widehat{\mathbf{x}}) \, d\widehat{S} = \int_{\gamma} \mathbf{g}(\mathbf{x}) \, dS.$$

To sum up, we have

$$\widehat{\mathbf{g}}(\widehat{\mathbf{x}}(\mathbf{x})) = \mathbf{g}(\mathbf{x}), \tag{2.36}$$

due to the fact that the force acts on the surface area and the resulting force is constant. The way how to implement this boundary load into the variational equation (2.15) was shown in a previous section. The respective energy functional already mentioned in theorem 2.2.9 is

$$W^{(g)}(U, \mathbf{g}) = - \int_{\Gamma^N} \mathbf{g}(\mathbf{x})^{\top} \mathbf{U}(\mathbf{x}) \, dS.$$

<sup>4</sup>This would not hold true for a hydrostatic pressure load on the boundary because it always acts perpendicular to the surface and thus the direction depends on the current surface normal.

<sup>5</sup>Again, this does not hold true for a constant hydrostatic pressure load on the boundary because an increase in the area also leads to an increase in the resulting force.



The first derivative of this energy functional is

$$W_{,\mathbf{U}}^{(g)}(\mathbf{U}, \mathbf{g})[\delta \mathbf{U}] = - \int_{\Gamma^N} \mathbf{g}(\mathbf{x})^\top \delta \mathbf{U}(\mathbf{x}) \, dS,$$

and its second derivative is zero.

### Inner Pressure $t$ .

An inner pressure, unlike an external pressure, can be viewed as a stress inside the body. First, we give an example where inner pressures occur.

**Example 2.2.12** (Turgor Pressure). Turgor pressure appears in cellular biology among plant, bacteria and fungi cells, see (Campbell, 1997, pg. 835f) or Raven et al. (2001). It is the pressure of the cell plasma against the cell wall and is caused by the osmotic flow of water inside or outside the cell. Three states describing the levels of water inside the cell are shown in figure 2.2.3. If water flows into a cell, the volume of the cell increases and might press against the cell wall. This introduces an inner pressure in the body, called *turgor pressure*. We assume this pressure to be hydrostatic. The plasmolyzed state can shrink the volume of the cell as it is a common observation of withered plants or dried fruits. Therefore the turgor pressure can be positive or negative.

Let the internal pressure be given by a scalar field  $\hat{t} : \hat{\Omega} \rightarrow \mathbb{R}$  on the current configuration, e.g. in the physical unit MPa = N/mm<sup>2</sup>. Let  $\hat{\omega} \subset \hat{\Omega}$  be a control domain of the deformed body. We assume the pressure to be hydrostatic, that means that the resulting force over a surface is perpendicular to the surface. This is equivalent to modeling it as a stress  $\hat{t}(\hat{\mathbf{x}})I$  at a material point  $\hat{\mathbf{x}} \in \hat{\Omega}$ , with the identity matrix  $I \in \mathbb{M}^3$ , compare Cauchy's Theorem 2.2.1. The stress tensor  $\hat{T}$  can be seen as a counter action to the inner pressure  $t$ , therefore the balance of forces on  $\hat{\omega}$  reads

$$\begin{aligned} 0 &= \int_{\partial \hat{\omega}} \hat{T}(\hat{\mathbf{x}}) \hat{\mathbf{n}}(\hat{\mathbf{x}}) - \hat{t} \hat{\mathbf{n}}(\hat{\mathbf{x}}) \, d\partial \hat{\omega} = \int_{\partial \hat{\omega}} (\hat{T}(\hat{\mathbf{x}}) - \hat{t}(\hat{\mathbf{x}})I) \hat{\mathbf{n}}(\hat{\mathbf{x}}) \, d\partial \hat{\omega} \\ &= \int_{\hat{\omega}} \widehat{\operatorname{div}} \left( \hat{T}(\hat{\mathbf{x}}) - \hat{t}(\hat{\mathbf{x}})I \right) \, d\hat{\omega}. \end{aligned}$$

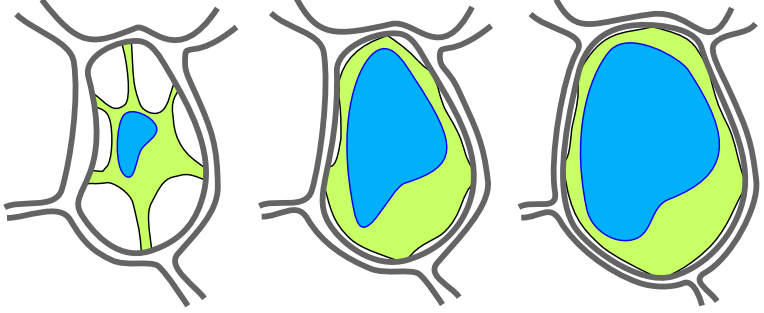
As the subdomain  $\hat{\omega}$  can be arbitrarily chosen, the following equation has to hold,

$$\widehat{\operatorname{div}} \left( \hat{T}(\hat{\mathbf{x}}) - \hat{t}(\hat{\mathbf{x}})I \right) = \mathbf{0} \quad \text{in } \hat{\Omega}. \quad (2.37)$$

A corresponding weak formulation to (2.37) on the current configuration is

$$\int_{\hat{\Omega}} \left( \hat{T}(\hat{\mathbf{x}}) - \hat{t}(\hat{\mathbf{x}})I \right) : \widehat{\nabla} \hat{\mathbf{V}}(\hat{\mathbf{x}}) \, d\hat{\mathbf{x}} = 0 \quad \forall \hat{\mathbf{V}}.$$





**FIGURE 2.2. Turgor pressure caused by osmotic water flow:** One can see three different states of water levels (in blue) in a cell (green) and the surrounding cell walls (gray). If the water flows outwards of a cell, its volume will decrease and it will eventually reach the *plasmolyzed* state in the left picture. Here the cell has less water than it could normally hold. The center picture shows the *flaccid* state, where the cell has enough water to approximately fill the cell. The *turgid* state is illustrated in the right picture, where the inflow of water causes the cell to expand so much that it presses against the cell walls.

Again, this is transformed onto the reference configuration  $\Omega$  by the Piola transformation (2.14) for the stress tensor  $\hat{T}$  and the transformation rules for the domain and the differential operator  $\hat{\nabla}$ . We also assume that the pressure in a material point does not depend on the displacement  $\mathbf{U}$  but only on its material point  $\mathbf{x} = \mathbf{x}(\hat{\mathbf{x}})$ , that is  $t(\mathbf{x}) = \hat{t}(\hat{\mathbf{x}}(\mathbf{x}))$ . Combining the above yields the weak formulation in the reference configuration

$$\int_{\Omega} (T(\mathbf{x}) - t(\mathbf{x}) \underbrace{(\det F(\mathbf{x})) F^{-\top}(\mathbf{x})}_{=\text{cof } F(\mathbf{x})}) : \nabla \mathbf{V}(\mathbf{x}) \, d\mathbf{x} = 0 \quad \forall \mathbf{V}. \quad (2.38)$$

One can see that the inner pressure  $t$  enters the balance of forces in a different way than the external volume or boundary load. To complete the modeling, we can define the energy for the inner pressure to be

$$W^{(t)}(\mathbf{U}, t) = - \int_{\Omega} t(\mathbf{x}) (\det (I + \nabla \mathbf{U}(\mathbf{x}))) \, d\mathbf{x} = \int_{\Omega} t(\mathbf{x}) (\det F(\mathbf{x})) \, d\mathbf{x}, \quad (2.39)$$



then its directional derivative, see lemma A.2.2, is

$$W_{,\mathbf{U}}^{(t)}(\mathbf{U}, t)[\delta\mathbf{U}] = - \int_{\Omega} t(\mathbf{x}) \operatorname{cof} F(\mathbf{x}) : \nabla \delta\mathbf{U}(\mathbf{x}) \, d\mathbf{x},$$

which matches the corresponding part in (2.38).

There is a special case if the inner pressure  $t$  is constant on  $\Omega$ , e.g.  $t(\mathbf{x}) = t_0 \in \mathbb{R}$ . Using Green's identity and the Piola identity  $\operatorname{div} \operatorname{cof} F = 0$ , see (Ciarlet, 1988, pg. 39), the derivative  $W_{,\mathbf{U}}^{(t)}$  can be rewritten as

$$\begin{aligned} W_{,\mathbf{U}}^{(t)}(\mathbf{U}, t)[\delta\mathbf{U}] &= -t_0 \int_{\Omega} \operatorname{cof} F : \nabla \delta\mathbf{U} \, d\mathbf{x} \\ &= -t_0 \int_{\Omega} \sum_{ij} [\operatorname{cof} F]_{ji} \delta U_{i,j} \, d\mathbf{x} && \text{(use Green's identity)} \\ &= -t_0 \int_{\Gamma} \sum_{ij} [\operatorname{cof} F]_{ji} \delta U_i \mathbf{n}_j \, dS && + t_0 \int_{\Omega} \sum_{ij} ([\operatorname{cof} F]_{ji})_{,j} \delta U_i \, d\mathbf{x} \\ &= -t_0 \int_{\Gamma} \left( \delta \mathbf{U}^{\top} \operatorname{cof} F \mathbf{n} \right) \, dS && + t_0 \int_{\Omega} (\operatorname{div} \operatorname{cof} F) \delta \mathbf{U} \, d\mathbf{x} \\ &= -t_0 \int_{\Gamma} \left( \delta \mathbf{U}^{\top} \operatorname{cof} F \mathbf{n} \right) \, dS. \end{aligned}$$

This term for the variational equation (2.15) is the same as for constant external hydrostatic pressure loads  $t_0$  as it can be found in (Ciarlet, 1988, ch. 2.7). This implies that constant internal pressures can be viewed as constant external pressures and vice versa. However, since it is an open question if a nonconstant external pressure is a conservative load, we exclude an external hydrostatic pressure for our work.

### Tension $m$ along Fibers.

We now consider and model a load that runs along a given fiber direction. This is motivated by biological models again.

**Example 2.2.13** (Muscle fibers). Muscle fibers can be found in many animals including birds, mammals, fish, and amphibians, see (Campbell, 1997, pg. 1139f). In most cases, the muscle controls a skeleton joint, where fibers are aligned in the same direction. However, there are also muscles that control the shape of a certain tissue, e.g. the tongue or the heart muscle. The tongue has certain areas where the direction of the muscle fibers are fairly grouped. Still, it can perform a wide variation of movements and is even co-responsible for the human speaking abilities. As for the other example, the heart muscle controls by its movement and shape



the blood circulation through the heart chambers and eventually through the whole body. Irregularities like cardiac dysrhythmia or ventricular fibrillation show how complex this mechanism is and how hard it is to control. Take for example the defibrillation which is still a painful and sometimes traumatic therapy today. In this field, mathematics is used to improve therapy and the chance of survival of the patients, see for example Nagaiah et al. (2013).

Even though these tissues are expected to have an anisotropic material behavior, we simplify it by replacing it with an isotropic elastic material in this thesis.

We start the modeling of an internal tension along a fiber on the current configuration. Let the vector field  $\hat{\mathbf{a}} : \hat{\Omega} \rightarrow \mathbb{R}^3$  describe the fiber direction inside the deformed body. For simplicity, we assume that the fiber direction is a unit vector, that is the Euclidean norm  $\|\hat{\mathbf{a}}(\hat{\mathbf{x}})\|_2 = 1$  for all  $\hat{\mathbf{x}} \in \hat{\Omega}$ . Let the scalar field  $\hat{m} : \hat{\Omega} \rightarrow \mathbb{R}$  describe the tension along a fiber, e.g. in the unit  $\text{MPa} = \text{N/mm}^2$ . The control volume  $\hat{\omega} \subset \hat{\Omega}$  of the deformed body can be arbitrarily chosen.

The force of the tension always runs along the fibers, that means that the resulting force on a surface of  $\hat{\omega}$  is  $m \hat{\mathbf{a}}(\hat{\mathbf{a}}^\top \hat{\mathbf{n}})$ .<sup>6</sup> Again, the stress tensor  $\hat{T}$  is seen as a counter action to the tension  $m$ , therefore the balance of forces on  $\hat{\omega}$  is

$$\begin{aligned} 0 &= \int_{\partial \hat{\omega}} \left( \hat{T}(\hat{\mathbf{x}}) - \hat{m}(\hat{\mathbf{x}}) \hat{\mathbf{a}}(\hat{\mathbf{x}}) \hat{\mathbf{a}}(\hat{\mathbf{x}})^\top \right) \hat{\mathbf{n}} \, d\partial \hat{\omega} \\ &= \int_{\hat{\omega}} \widehat{\text{div}} \left( \hat{T}(\hat{\mathbf{x}}) - \hat{m}(\hat{\mathbf{x}}) \hat{\mathbf{a}}(\hat{\mathbf{x}}) \hat{\mathbf{a}}(\hat{\mathbf{x}})^\top \right) \, d\hat{\omega}. \end{aligned}$$

As the subdomain  $\hat{\omega}$  can be arbitrarily chosen, we get the equation

$$\widehat{\text{div}} \left( \hat{T}(\hat{\mathbf{x}}) - \hat{m}(\hat{\mathbf{x}}) \hat{\mathbf{a}}(\hat{\mathbf{x}}) \hat{\mathbf{a}}(\hat{\mathbf{x}})^\top \right) = \mathbf{0} \quad \text{in } \hat{\Omega}, \quad (2.40)$$

and its weak variational form

$$\int_{\hat{\Omega}} \left( \hat{T}(\hat{\mathbf{x}}) - \hat{m}(\hat{\mathbf{x}}) \hat{\mathbf{a}}(\hat{\mathbf{x}}) \hat{\mathbf{a}}(\hat{\mathbf{x}})^\top \right) : \widehat{\nabla} \hat{\mathbf{V}}(\hat{\mathbf{x}}) \, d\hat{\mathbf{x}} = 0 \quad \forall \hat{\mathbf{V}}. \quad (2.41)$$

In order to transform this weak formulation onto  $\Omega$ , we need to model the transformation of  $\hat{m}$  and  $\hat{\mathbf{a}}$ :

- During the deformation, the fiber orientation is rotated along the deformation. This can be written as

$$\hat{\mathbf{a}}(\hat{\mathbf{x}}(\mathbf{x})) = \frac{F(\mathbf{x})\mathbf{a}(\mathbf{x})}{\|F(\mathbf{x})\mathbf{a}(\mathbf{x})\|_2}.$$

<sup>6</sup>The last scaling term ( $\hat{\mathbf{a}}^\top \hat{\mathbf{n}}$ ) is due to the fact that the tangential component of  $m \hat{\mathbf{a}}$  does not affect the resulting force on  $\hat{\omega}$  because it is an inner stress. The same argument can also be found for the Cauchy stress tensor  $\hat{T}$  in Cauchy's Theorem 2.2.1.



- The tension  $m : \Omega \rightarrow \mathbb{R}$  is assumed to scale with the volume, that is

$$m(\mathbf{x}) = \widehat{m}(\widehat{\mathbf{x}}) \det F(\mathbf{x}). \quad (2.42)$$

This can be justified for muscle cells in the following way: Let  $\omega$  and  $\widehat{\omega} = \widehat{\mathbf{x}}(\omega)$  be subdomains representing a cell in the undeformed and deformed body. This cell has the ability to enforce a tension along its fiber direction. The tension along the fiber  $\widehat{\mathbf{a}}$  in a point  $\widehat{\mathbf{x}} \in \widehat{\omega}$  is

$$\widehat{\mathbf{a}}(\widehat{\mathbf{x}})^\top \left( \widehat{m}(\widehat{\mathbf{x}}) \widehat{\mathbf{a}}(\widehat{\mathbf{x}}) \widehat{\mathbf{a}}(\widehat{\mathbf{x}})^\top \right) \widehat{\mathbf{a}}(\widehat{\mathbf{x}}) = \widehat{m}(\widehat{\mathbf{x}}),$$

because  $\widehat{\mathbf{a}}^\top \widehat{\mathbf{a}} = 1$ . The same holds true for the undeformed configuration  $\omega$ ,

$$\mathbf{a}(\mathbf{x})^\top \left( m(\mathbf{x}) \mathbf{a}(\mathbf{x}) \mathbf{a}(\mathbf{x})^\top \right) \mathbf{a}(\mathbf{x}) = m(\mathbf{x}).$$

We assume that this tension does not change if the volume of the cell changes due to deformations, that means the cell itself retains its power during the deformation. Therefore the total tension caused by the cell has to be equal for both configurations, that means

$$\int_{\widehat{\omega}} \widehat{m}(\widehat{\mathbf{x}}) \, d\widehat{\mathbf{x}} = \int_{\omega} \widehat{m}(\widehat{\mathbf{x}}(\mathbf{x})) \det F(\mathbf{x}) \, d\mathbf{x} = \int_{\omega} m(\mathbf{x}) \, d\mathbf{x}.$$

As the cells will be viewed as a continuum in order to fit into our model of elasticity, the subdomain  $\omega$  can be chosen arbitrarily and equation (2.42) follows.

We can now transform the weak formulation (2.41) onto  $\Omega$  and get the equation

$$\int_{\Omega} T : \nabla \delta \mathbf{U} - m \|\mathbf{F}\mathbf{a}\|_2^{-2} \left( \mathbf{F}\mathbf{a} (\mathbf{F}\mathbf{a})^\top \right) : (\nabla \delta \mathbf{U} \mathbf{F}^{-1}) \, d\mathbf{x} = 0 \quad \forall \delta \mathbf{U}.$$

A part of the second term in the integral can be simplified, e. g.

$$\begin{aligned} \left( \mathbf{F}\mathbf{a} (\mathbf{F}\mathbf{a})^\top \right) : (\nabla \delta \mathbf{U} \mathbf{F}^{-1}) &= (\mathbf{F}\mathbf{a})^\top (\nabla \delta \mathbf{U} \mathbf{F}^{-1}) (\mathbf{F}\mathbf{a}) \\ &= \mathbf{a}^\top \mathbf{F}^\top (\nabla \delta \mathbf{U} \mathbf{a}). \end{aligned}$$

Inserting this yields the weak formulation

$$\int_{\Omega} T(\mathbf{x}) : \nabla \delta \mathbf{U}(\mathbf{x}) - m(\mathbf{x}) \frac{(\mathbf{a}(\mathbf{x})^\top \mathbf{F}^\top(\mathbf{x}) \nabla \delta \mathbf{U}(\mathbf{x}) \mathbf{a}(\mathbf{x}))}{\|F(\mathbf{x})\mathbf{a}(\mathbf{x})\|_2^2} \, d\mathbf{x} = 0 \quad \forall \delta \mathbf{U}. \quad (2.43)$$



**Theorem 2.2.14** (Energy functional for the fibre tension). The energy functional for the fiber tension  $m$  in the weak formulation (2.43) is

$$W^{(m)}(\mathbf{U}, m) = - \int_{\Omega} m(\mathbf{x}) \ln \|F(\mathbf{x})\mathbf{a}(\mathbf{x})\|_2 \, d\mathbf{x}. \quad (2.44)$$

*Proof.* The derivative  $W_{\mathbf{U}}^{(m)}(\mathbf{U}, m)[\delta\mathbf{U}]$  can be found in lemma A.2.3 and matches the term from the variational equation (2.43).

**Summary** Finally, we incorporate the different loads presented in this section to a more unified model. For simplicity of notation, in the case of a single load, let us denote this load by  $\mathbf{C}$ , e. g. if only a volume load is applied, we have  $\mathbf{C} = \mathbf{f}$ . A list of all loads can be found in table 2.1.

load type	$\mathbf{C}$	energy $W^{(\mathbf{C})}$	derivative $W_{\mathbf{U}}^{(\mathbf{C})}(\mathbf{U}, \mathbf{C})[\delta\mathbf{U}]$
Volume load	$\mathbf{f}$	$-\int_{\Omega} \mathbf{f}^{\top} \mathbf{U} \, d\mathbf{x}$	$-\int_{\Omega} \mathbf{f}^{\top} \delta\mathbf{U} \, d\mathbf{x}$
Boundary load	$\mathbf{g}$	$-\int_{\Gamma^N} \mathbf{g}^{\top} \mathbf{U} \, dS$	$-\int_{\Gamma^N} \mathbf{g}^{\top} \delta\mathbf{U} \, dS$
Inner pressure	$t$	$-\int_{\Omega} t (\det F) \, d\mathbf{x}$	$-\int_{\Omega} t (\det F) F^{-\top} : \nabla \delta\mathbf{U} \, d\mathbf{x}$
Fiber tension	$m$	$-\int_{\Omega} m \ln (\ F\mathbf{a}\ ) \, d\mathbf{x}$	$-\int_{\Omega} m \frac{\mathbf{a}^{\top} F^{\top} \nabla \delta\mathbf{U} \mathbf{a}}{\mathbf{a}^{\top} F^{\top} F \mathbf{a}} \, d\mathbf{x}$
Cauchy stress	$T$	$\int_{\Omega} w(F) \, d\mathbf{x}$	$\int_{\Omega} w_{,F}(F) [\nabla \delta\mathbf{U}] \, d\mathbf{x}.$

TABLE 2.1. Different loads that are modeled in section 2.2.3. The Cauchy stress is added to this list for completeness although it is not a load.

**Definition 2.2.15** (Control space). Let  $\mathbf{C}$  be one of the loads presented earlier and  $\Omega^{\text{ctrl}} \subset \Omega$  respectively  $\Gamma^{\text{ctrl}} \subset \Gamma$  the part of the domain or boundary where the load is applied. We define the *control space*  $\mathcal{C}$  to be the  $L_2$ -space on  $\Omega^{\text{ctrl}}$  respectively  $\Gamma^{\text{ctrl}}$ , with the corresponding scalar product  $(\cdot, \cdot)_{\mathcal{C}}$ . More details are given in table 2.2.



load type	$\mathbf{C}$	control space $\mathcal{C}$	scalar product $(\cdot, \cdot)_{\mathcal{C}}$
Volume load	$\mathbf{f}$	$\mathcal{C}^{(\mathbf{f})} = L_2(\Omega^{\text{ctrl}})^3$	$(\mathbf{f}_1, \mathbf{f}_2)_{\mathbf{f}} = \int_{\Omega^{\text{ctrl}}} \mathbf{f}_1(\mathbf{x})^\top \mathbf{f}_2(\mathbf{x}) \, d\mathbf{x}$
Boundary load	$\mathbf{g}$	$\mathcal{C}^{(\mathbf{g})} = L_2(\Gamma^{\text{ctrl}})^3$	$(\mathbf{g}_1, \mathbf{g}_2)_{\mathbf{g}} = \int_{\Gamma^{\text{ctrl}}} \mathbf{g}_1(\mathbf{x})^\top \mathbf{g}_2(\mathbf{x}) \, dS$
Inner pressure	$t$	$\mathcal{C}^{(t)} = L_2(\Omega^{\text{ctrl}})$	$(t_1, t_2)_t = \int_{\Omega^{\text{ctrl}}} t_1(\mathbf{x}) t_2(\mathbf{x}) \, d\mathbf{x}$
Fiber tension	$m$	$\mathcal{C}^{(m)} = L_2(\Omega^{\text{ctrl}})$	$(m_1, m_2)_m = \int_{\Omega^{\text{ctrl}}} m_1(\mathbf{x}) m_2(\mathbf{x}) \, d\mathbf{x}$

TABLE 2.2. Different loads with their control spaces and corresponding scalar products.

With this notation, the general energy minimization problem for a given load  $\mathbf{C} \in \mathcal{C}$  is

$$\begin{aligned} & \min_{\mathbf{U} \in \mathcal{U}} W(\mathbf{U}, \mathbf{C}) \\ & \text{with } W(\mathbf{U}, \mathbf{C}) = W^{(T)}(\mathbf{U}) + W^{(\mathbf{C})}(\mathbf{U}, \mathbf{C}). \end{aligned} \quad (2.45)$$

The formal first order optimality conditions are

$$\begin{aligned} 0 &= W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\delta\mathbf{U}] \\ &= W_{,\mathbf{U}}^{(T)}(\mathbf{U})[\delta\mathbf{U}] + W_{,\mathbf{U}}^{(\mathbf{C})}(\mathbf{U}, \mathbf{C})[\delta\mathbf{U}]. \end{aligned} \quad (2.46)$$

We will call this equation the *forward problem*. Of course, the problem can be extended to a finite number of loads as well, simply by adding their corresponding energy functionals to the stored energy

$$W(\mathbf{U}, \mathbf{C}_1, \mathbf{C}_2, \dots) = W^{(T)}(\mathbf{U}) + \sum_i W^{(\mathbf{C}_i)}(\mathbf{U}, \mathbf{C}_i).$$

**Remark 2.2.16.** Ball's existence result from theorem 2.2.9 applies only to the case of volume loads  $\mathbf{f}$  and boundary loads  $\mathbf{g}$ .

## 2.3 Comparison to Linear Elasticity

In the case of very small displacements  $\mathbf{U}$ , the variational equation (2.46) can be linearized with respect to  $\mathbf{U}$ , neglecting higher order terms of  $\mathbf{U}$ . One constitutive assumption of stored energies is that this linearization should match the linear model



of elasticity, see theorem 2.2.6. The linear model to describe an isotropic elastic material behavior is build on the *St. Venant* model defined by the energy density

$$w^{\text{StV}}(F) = \frac{\lambda}{2}(\text{tr } E)^2 + \mu(E : E), \quad (2.47)$$

where the *Green-St. Venant strain tensor*  $E$  is defined as  $E := \frac{1}{2}(F^\top F - I)$  and the *Lamé parameters* are  $\lambda > 0$  and  $\mu > 0$ . This energy functional is not polyconvex, for a proof we refer to (Ciarlet, 1988, Th. 4.10-1). As the Green-St. Venant strain tensor  $E$  itself is nonlinear with respect to  $\mathbf{U}$ , there is a special notation for the *linearized strain tensor*

$$\varepsilon(\mathbf{U}) := \frac{1}{2}(\nabla \mathbf{U} + \nabla \mathbf{U}^\top). \quad (2.48)$$

Minimizing the global St. Venant energy

$$W^{\text{lin}}(\mathbf{U}, \mathbf{C}) := \int_{\Omega} \frac{\lambda}{2}(\text{tr } \varepsilon(\mathbf{U}))^2 + \mu(\varepsilon(\mathbf{U}) : \varepsilon(\mathbf{U})) \, d\mathbf{x} + (\text{terms from load } \mathbf{C}), \quad (2.49)$$

with the linearized strain tensor  $\varepsilon$  instead of  $E$ , one obtains the first order optimality condition

$$0 = W_{,\mathbf{U}}^{\text{StV}}[V] = \int_{\Omega} \lambda \text{tr}(\nabla \mathbf{U}) \text{tr}(\nabla \mathbf{V}) + 2\mu(\varepsilon(\mathbf{U}) : \varepsilon(\mathbf{V})) \, d\mathbf{x} + (\text{terms from load } \mathbf{C}). \quad (2.50)$$

This variational equation is also obtained if the weak formulation (2.46) for large deformations is linearized at  $\mathbf{U} \equiv \mathbf{0}$ , see theorem 2.2.6. Hence the model in linear elasticity can be viewed as a linearization of the nonlinear problem at  $\mathbf{U} \equiv \mathbf{0}$ . As a consequence, it is expected to be applicable only for very small deformations, which can be underpinned by two major reasons:

### Self-Penetration

One can also see that the property (2.20), that is  $w(F) \rightarrow +\infty$  as  $J \rightarrow 0$ , does not hold for the St. Venant energy (2.47). As a consequence, there are deformations where the local volume change  $J(\mathbf{x})$  is negative in certain regions, that means the material *penetrates itself* locally, which is a violation of continuum mechanics. We illustrate this with an example.

**Example 2.3.1** (Self-penetration of a compressed unit cube). Let  $\Omega = (0, 1)^3$  be a unit cube which is clamped at the left side, that is  $\Gamma^D = 0 \times [0, 1] \times [0, 1]$ . There is a constant volume load  $\mathbf{f}(\mathbf{x}) = (f, 0, 0)$  and the Lamé parameters are  $\lambda = 1$  and  $\mu = 1$ . If  $f$  is large enough, e.g.  $f = 2.2$ , the volume change  $J$  is no longer positive for all points  $\mathbf{x} \in \Omega$ , see figure 2.3.

### Rigid Body Movements



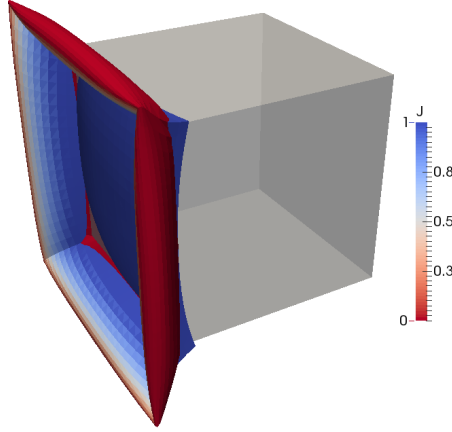


FIGURE 2.3. The undeformed cube  $\Omega$  in gray is compressed to the colored body. The preservation of a positive volume does not hold and the body penetrates itself.

A rigid body can undergo two types of deformations, namely *translation* and *rotation*, that do not cause any stresses. They are called *rigid body movements* and can be written as  $\hat{\mathbf{x}}(\mathbf{x}) = \mathbf{U}_{\text{trans}} + Q\mathbf{x}$ ,  $\mathbf{x} \in \Omega$ , consisting of a translation  $\mathbf{U}_{\text{trans}} \in \mathbb{R}^3$  and a rotation with a rotation matrix  $Q \in \mathbb{M}_+^3$ ,  $Q^\top Q = I$ . The corresponding displacement is

$$\mathbf{U}(\mathbf{x}) = \mathbf{U}_{\text{trans}} + (Q - I)\mathbf{x}, \quad \mathbf{x} \in \Omega.$$

A translation  $\mathbf{U}_{\text{trans}}$  is a rigid body movement for the linear and nonlinear model, because both depend on the gradient  $\nabla \mathbf{U}$  and for a translation we always have  $\nabla \mathbf{U}_{\text{trans}} = \mathbf{0}$ . As a consequence, a translation  $\mathbf{U}_{\text{trans}}$  would not show up in the balance of forces. If no clamping is applied, that is  $\Gamma^D = \emptyset$ , the translation  $\mathbf{U}_{\text{trans}}$  can be viewed as a three-dimensional kernel of the linear, respectively nonlinear, operator belonging to the model. However, in many practical problems, a clamping appears naturally.

As for rotations, they are not rigid body movements in the linear model. To see this, we take a look at the Green-St.Venant strain tensor  $E$  and its linearized version  $\varepsilon$ ,

$$\begin{aligned} 2E &= F^\top F - I &= \nabla \mathbf{U} + \nabla \mathbf{U}^\top + \nabla \mathbf{U}^\top \nabla \mathbf{U}, \\ 2\varepsilon & &= \nabla \mathbf{U} + \nabla \mathbf{U}^\top. \end{aligned}$$



Inserting  $\mathbf{U}(\mathbf{x}) = \mathbf{U}_{\text{trans}} + (\mathbf{Q} - \mathbf{I})\mathbf{x}$ , that is  $\nabla \mathbf{U} = \mathbf{Q} - \mathbf{I}$ , into  $E$  and  $\varepsilon$  yields

$$\begin{aligned} 2E &= \mathbf{Q} - \mathbf{I} + \mathbf{Q}^\top - \mathbf{I} + (\mathbf{Q} - \mathbf{I})^\top (\mathbf{Q} - \mathbf{I}) \\ &= \mathbf{Q} - \mathbf{I} + \mathbf{Q}^\top - \mathbf{I} + \mathbf{Q}^\top \mathbf{Q} - \mathbf{Q}^\top - \mathbf{Q} + \mathbf{I} = 0 \\ 2\varepsilon &= \mathbf{Q} + \mathbf{Q}^\top - 2\mathbf{I} = (\mathbf{Q}^\top - \mathbf{I})(\mathbf{Q} - \mathbf{I}). \end{aligned}$$

We can see that the linearized strain tensor  $\varepsilon$  is only zero for the identity  $\mathbf{Q} = \mathbf{I}$ . This is the reason why *rotations are not rigid body movements in linear elasticity* because the Green-St.Venant strain tensor  $E$  or  $\varepsilon$  are measures for the occurring strains in a deformed body. These strains induce certain stresses defined by the material law. The presented material law in 2.18 and the St.Venant energy (2.47) have the property that  $T(\mathbf{x}) = 0 \Leftrightarrow E(\mathbf{x}) = 0$  or  $\varepsilon(\mathbf{x}) = 0$  respectively. Large deformations often exhibit local rotations, e.g. the tip of a bending bar is sloped. These rotations themselves cause stresses in the linear model, which are not observed in reality.

We summarize this section by emphasizing the differences between the linear and nonlinear model of elasticity in Table 2.3.

linear model from equation (2.50)	nonlinear model from equation (2.46)
suitable only for small deformations	suitable for larger deformations
only one configuration $\Omega$	two configurations $\Omega$ and $\widehat{\Omega}$
no transformations of variational equations are necessary	transformations $\widehat{\Omega} \rightarrow \Omega$ are necessary for the variational equations
rotations are <i>not</i> rigid body movements	rotations are rigid body movements
$2\varepsilon = \nabla \mathbf{U} + \nabla \mathbf{U}^\top$	$2E = \nabla \mathbf{U} + \nabla \mathbf{U}^\top + \nabla \mathbf{U}^\top \nabla \mathbf{U}$
energy is quadratic in $\mathbf{U}$	energy ensures $w \rightarrow 0$ as $J \rightarrow 0$
Cauchy stress $T$ is linear in $\mathbf{U}$	Cauchy stress $T$ is nonlinear in $\mathbf{U}$

TABLE 2.3. Comparison of the linear and the nonlinear model of elasticity.



## 2.4 Parametrization of the Reference Configuration $\Omega$

So far, the position of the material point  $\mathbf{x} \in \Omega$  was a given argument. However, for certain bodies like tubes, a parametrization might be helpful to describe the problem with curvilinear coordinates  $\mathbf{x}(\tilde{\mathbf{x}})$ . For a given *parameter space*  $\tilde{\Omega} \subset \mathbb{R}^3$ , the reference configuration can be described as

$$\Omega = \left\{ \mathbf{x} = \mathbf{x}(\tilde{\mathbf{x}}), \tilde{\mathbf{x}} \in \tilde{\Omega} \right\}, \quad (2.51)$$

with a sufficiently smooth and bijective function  $\mathbf{x} : \tilde{\Omega} \rightarrow \Omega$ , see figure 2.4. In the following, the tilde  $\sim$  emphasizes that a function takes arguments in the parameter space  $\tilde{\Omega}$  or a differentiation is with respect to the parameter  $\tilde{\mathbf{x}}$ .

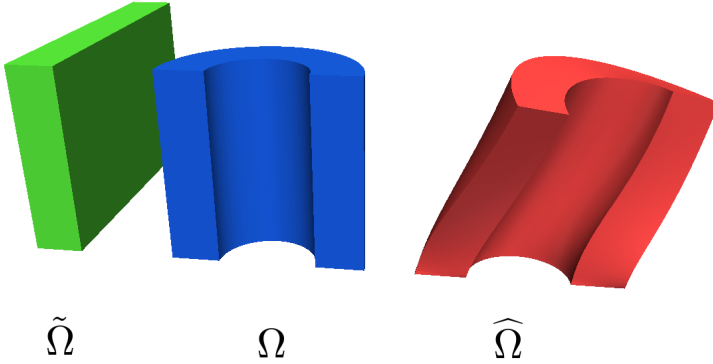


FIGURE 2.4. Cylindrical coordinates from example 2.4.1: the parameter space  $\tilde{\Omega}$  is shown in green, while the undeformed body  $\Omega$  is plotted in blue and the deformed body  $\hat{\Omega}$  in red.

Our task is to reformulate the weak formulation (2.30) in  $\tilde{\Omega}$  in order to use numerical methods like finite elements. First we define the Jacobi-matrix

$$\tilde{G}(\tilde{\mathbf{x}}) := \frac{\partial \mathbf{x}}{\partial \tilde{\mathbf{x}}}(\tilde{\mathbf{x}}), \quad [\tilde{G}]_{ij} = \frac{\partial x_i}{\partial \tilde{x}_j}, \quad i, j = 1, 2, 3. \quad (2.52)$$

The transformation of vector fields  $\tilde{\mathbf{V}} : \tilde{\Omega} \rightarrow \mathbb{R}^3$  is defined by  $\tilde{\mathbf{V}}(\tilde{\mathbf{x}}) := \mathbf{V}(\mathbf{x}(\tilde{\mathbf{x}}))$ , while formulas for the volume and area elements can be found in Ciarlet (2005), that is

$$d\mathbf{x} = |\det \tilde{G}| d\tilde{\Omega} \quad \text{and} \quad dS = |\det \tilde{G}(\tilde{\mathbf{x}})| \|\tilde{G}^{-\top}(\tilde{\mathbf{x}}) \tilde{\mathbf{n}}(\tilde{\mathbf{x}})\|_2 d\tilde{S}. \quad (2.53)$$



We assume in the following, that  $\det \tilde{G}(\tilde{\mathbf{x}}) > 0$ ,  $\forall \tilde{\mathbf{x}} \in \tilde{\Omega}$ , to ensure that the parametrization is injective, see (Ciarlet, 2005, pg. 18). As we have seen earlier, the energy minimization (2.45) and its first order optimality as a weak variational equation (2.46) can be formulated in terms of the displacement  $\mathbf{U}$  and the deformation gradient  $F$ . Therefore it is sufficient to give presentations of these two quantities. The transformation of  $\mathbf{U}$  is straight-forward, i. e.

$$\tilde{\mathbf{U}}(\tilde{\mathbf{x}}) := \mathbf{U}(\mathbf{x}(\tilde{\mathbf{x}})). \quad (2.54)$$

To transform the deformation gradient  $F$  we first define the differential operator  $\tilde{\nabla}$  as

$$[\tilde{\nabla} \tilde{\mathbf{U}}(\tilde{\mathbf{x}})]_{ij} = \frac{\partial \tilde{U}_i}{\partial \tilde{x}_k}(\tilde{\mathbf{x}}). \quad (2.55)$$

The relationship of  $\tilde{\nabla}$  to the differential operator  $\nabla$  can be computed by the chain rule

$$[\nabla \delta \mathbf{U}]_{ij}(\mathbf{x}(\tilde{\mathbf{x}})) := \sum_k \frac{\partial \delta \tilde{U}_i}{\partial \tilde{x}_k} \frac{\partial \tilde{x}_k}{\partial x_j}(\tilde{\mathbf{x}}), \quad i, j = 1, 2, 3,$$

or, in term of matrices,

$$[\nabla \delta \mathbf{U}](\mathbf{x}(\tilde{\mathbf{x}})) = \tilde{\nabla} \delta \tilde{\mathbf{U}}(\tilde{\mathbf{x}}) \tilde{G}^{-1}(\tilde{\mathbf{x}}), \quad i, j = 1, 2, 3. \quad (2.56)$$

Hence, we get the parametrized deformation gradient

$$\tilde{F}(\tilde{\mathbf{x}}) := F(\mathbf{x}(\tilde{\mathbf{x}})) = I + \nabla \mathbf{U}(\mathbf{x}(\tilde{\mathbf{x}})) = I + \tilde{\nabla} \tilde{\mathbf{U}}(\tilde{\mathbf{x}}) \tilde{G}^{-1}(\tilde{\mathbf{x}}). \quad (2.57)$$

We are now able to rewrite the weak variational equation (2.46) (with all loads attached) in terms of  $\tilde{\mathbf{U}}$ , yielding

$$\begin{aligned} 0 = & \int_{\tilde{\Omega}} w_{,F} \left( I + \tilde{F}(\tilde{\mathbf{x}}) \right) \left[ \tilde{\nabla} \delta \tilde{\mathbf{U}}(\tilde{\mathbf{x}}) \tilde{G}^{-1}(\tilde{\mathbf{x}}) \right] | \det \tilde{G}(\tilde{\mathbf{x}}) | d\tilde{\Omega} \\ & - \int_{\tilde{\Omega}} \mathbf{f}(\mathbf{x}(\tilde{\mathbf{x}}))^\top \delta \tilde{\mathbf{U}}(\tilde{\mathbf{x}}) | \det \tilde{G}(\tilde{\mathbf{x}}) | d\tilde{\Omega} \\ & - \int_{\tilde{\Gamma}^N} \mathbf{g}(\mathbf{x}(\tilde{\mathbf{x}}))^\top \delta \tilde{\mathbf{U}}(\tilde{\mathbf{x}}) | \det \tilde{G}(\tilde{\mathbf{x}}) | \| \tilde{G}^\top(\tilde{\mathbf{x}}) \tilde{\mathbf{n}}(\tilde{\mathbf{x}}) \| d\tilde{\Gamma} \\ & + \int_{\tilde{\Omega}} t(\mathbf{x}(\tilde{\mathbf{x}})) \operatorname{cof} \tilde{F}(\tilde{\mathbf{x}}) : (\tilde{\nabla} \delta \tilde{\mathbf{U}}(\tilde{\mathbf{x}}) \tilde{G}^{-1}(\tilde{\mathbf{x}})) | \det \tilde{G}(\tilde{\mathbf{x}}) | d\tilde{\Omega} \\ & + \int_{\tilde{\Omega}} m(\mathbf{x}(\tilde{\mathbf{x}})) \left( \mathbf{a}(\mathbf{x}(\tilde{\mathbf{x}}))^\top \tilde{F}^\top(\tilde{\mathbf{x}}) \tilde{\nabla} \delta \tilde{\mathbf{U}}(\tilde{\mathbf{x}}) \tilde{G}^{-1}(\tilde{\mathbf{x}}) \mathbf{a}(\mathbf{x}(\tilde{\mathbf{x}})) \right) | \det \tilde{G}(\tilde{\mathbf{x}}) | d\tilde{\Omega}. \end{aligned} \quad (2.58)$$



The parametrization of the loads  $\mathbf{f}$ ,  $\mathbf{g}$ ,  $t$  and  $m$  is not necessary, though it is possible. The same applies to the fiber directions  $\mathbf{a}$ , which might be easily described in the parametrization. However, this is more a point of the implementation and can be adapted by the user.

**Example 2.4.1** (Cylindrical Coordinates). Cylindrical coordinates are often used to describe bodies with a rotational symmetry, like tubes or rods. The coordinates

$$\mathbf{x}(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \tilde{\mathbf{x}}_3) = \begin{pmatrix} \tilde{\mathbf{x}}_1 \cos \tilde{\mathbf{x}}_2 \\ \tilde{\mathbf{x}}_1 \sin \tilde{\mathbf{x}}_2 \\ \tilde{\mathbf{x}}_3 \end{pmatrix} \quad \tilde{\mathbf{x}} \in \tilde{\Omega} := [r_1, r_2] \times [0, \pi] \times [0, h], \quad (2.59)$$

are a natural parametrization to describe such a domain, see figure 2.5. For the Jacobi-matrix  $\tilde{G}$  we have

$$\tilde{G}(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \tilde{\mathbf{x}}_3) = \begin{pmatrix} \cos \tilde{\mathbf{x}}_2 & -\tilde{\mathbf{x}}_1 \sin \tilde{\mathbf{x}}_2 & 0 \\ \sin \tilde{\mathbf{x}}_2 & \tilde{\mathbf{x}}_1 \cos \tilde{\mathbf{x}}_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (2.60)$$

and its inverse  $\tilde{G}^{-1}$  and the local volume  $\det \tilde{G}$  are

$$\tilde{G}^{-1}(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \tilde{\mathbf{x}}_3) = \begin{pmatrix} \cos \tilde{\mathbf{x}}_2 & \sin \tilde{\mathbf{x}}_2 & 0 \\ -\frac{1}{\tilde{\mathbf{x}}_1} \sin \tilde{\mathbf{x}}_2 & \frac{1}{\tilde{\mathbf{x}}_1} \cos \tilde{\mathbf{x}}_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \det \tilde{G}(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \tilde{\mathbf{x}}_3) = \tilde{\mathbf{x}}_1.$$

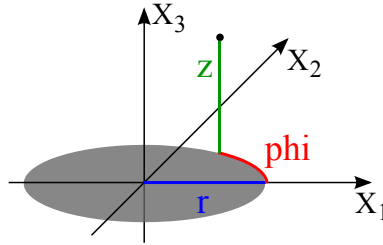


FIGURE 2.5. Cylindrical coordinates  $(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \tilde{\mathbf{x}}_3) = (r, \varphi, z)$  and the coordinates  $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$  for the reference configuration.

## 2.5 Hierarchical Plate Model

In many mechanical applications (e. g. elasto-plastic deep-drawing processes of plates), the physical body is very thin. This might cause large discretization errors in a standard three-dimensional finite element method. Several methods have been proposed to deal with this problem, among them a *hierarchical plate model*, suggested by



Babuška and Li (1991), Babuška et al. (1992), Babuška et al. (1993), Schwab and Wright (1995), Ovtchinnikov and Xanthis (1995) and others.

A *plate*, in contrast to a *shell*, is defined to be planar, that means it can be described by its midsurface  $\Omega^{2D} \subset \mathbb{R}^2$  and a perpendicular coordinate for the thickness, like in

$$\Omega = \Omega^{2D} \times (-d, d), \quad (2.61)$$

where the plate has the thickness  $2d > 0$ . The core idea of most plate models is the reduction from the three-dimensional domain  $\Omega$  to the midsurface  $\Omega^{2D}$ , that is all functions take arguments in  $\Omega^{2D}$ . This might be exact for certain given loads, but it can be only an approximation for the displacement  $\mathbf{U}$ . However, a good approximation that keeps most of the structure of  $\mathbf{U}$  can still pay off. We illustrate the idea of the hierarchical plate model with an example.

**Example 2.5.1** (A First Idea for a Plate Model). The ansatz is similar to many plate models: we start by splitting the displacement  $\mathbf{U}$ . Here, we decompose  $\mathbf{U} : \Omega \rightarrow \mathbb{R}^3$  into a part  $\mathbf{U}_0^{2D} : \Omega^{2D} \rightarrow \mathbb{R}^3$  to describe the deformed midsurface  $\widehat{\Omega}^{2D} = \mathbf{U}_0^{2D}(\Omega^{2D})$  and a part  $\mathbf{U}_1^{2D} : \Omega^{2D} \rightarrow \mathbb{R}^3$  for the thickness coordinate  $\mathbf{x}_3$ , see figure 2.6. This can be formulated as

$$\begin{aligned} \mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) &= \mathbf{U}_0^{2D}(\mathbf{x}_1, \mathbf{x}_2) + \mathbf{x}_3 \mathbf{U}_1^{2D}(\mathbf{x}_1, \mathbf{x}_2), \\ (\mathbf{x}_1, \mathbf{x}_2) &\in \Omega^{2D}, \mathbf{x}_3 \in (-d, d). \end{aligned}$$

This is not restricted to perpendicular midsurface normals like the Kirchhoff-Love hypothesis, but also allows shear deformations and thickness change. Of course, the real three dimensional displacement in general might be richer than the above description could offer. However, this can still be a good approximation for thin plates.

We now generalize this idea by adding more terms  $\mathbf{U}_i^{2D}$  to improve the approximation of the displacement over the thickness.

**Definition 2.5.2** (Hierarchical plate model). Let  $\mathbf{U}_i^{2D} : \Omega^{2D} \rightarrow \mathbb{R}^3$ ,  $i = 0, \dots, D^{2D}$  be sufficiently differentiable functions. We call the ansatz

$$\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \sum_{i=0}^{D^{2D}} p_i\left(\frac{1}{d} \mathbf{x}_3\right) \mathbf{U}_i^{2D}(\mathbf{x}_1, \mathbf{x}_2), \quad (2.62)$$

for  $(\mathbf{x}_1, \mathbf{x}_2) \in \Omega^{2D}$ ,  $\mathbf{x}_3 \in (-d, d)$ , the *hierarchical plate model* of the degree  $D^{2D}$ . The polynomials  $p_i$  are the Legendre polynomials of order  $i$ .

**Remark 2.5.3** (Legendre polynomials). The usage of Legendre polynomials  $p_i$  is due to numerical stability. If one would use powers of  $\mathbf{x}_3$ , e. g.  $(\mathbf{x}_3)^i$ , as coefficients



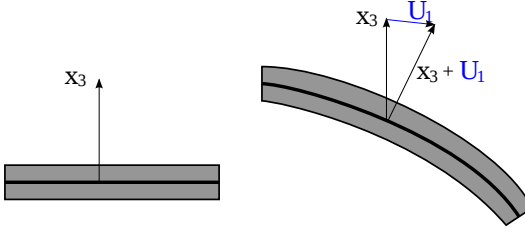


FIGURE 2.6. Cross section of a plate: The thick line represents the midsurface, left for the undeformed body and right for the deformed body. The role of  $\mathbf{U}_1^{2D}$  is to allow a change in the direction of the thickness, from  $\mathbf{x}_3$  in the reference configuration to  $\mathbf{x}_3 + \mathbf{U}_1^{2D}$  in the current configuration.

in front of  $\mathbf{U}_i^{2D}$ , numerical elimination would occur if the thickness  $2d$  is too small. The Legendre polynomials  $p_i$  can be found in Zeidler (2013), e. g.

$$\begin{aligned} p_0(x) &= 1 & p_1(x) &= x & p_2(x) &= \frac{1}{2}(3x^2 - 1) \\ p_3(x) &= \frac{1}{2}(5x^3 - 3x) & p_4(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3), \end{aligned} \quad (2.63)$$

and are plotted in figure 2.7.

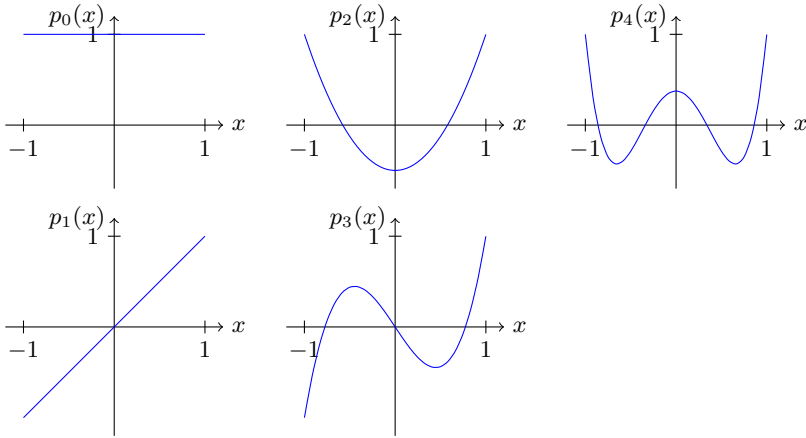


FIGURE 2.7. The first five Legendre polynomials defined in 2.63.



Our first task is to find a representation of the gradient operator  $\nabla$  for this ansatz (2.62). Let us recall that the  $\nabla$  is the derivative with respect to  $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ . The differentiation of  $\mathbf{U}_i^{2D}$  with respect to the first two spatial coordinates,  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , is the standard differentiation of the functions  $\mathbf{U}_i^{2D}$ , e. g.

$$\frac{\partial \mathbf{U}}{\partial \mathbf{x}_j}(\mathbf{x}) = \sum_{i=0}^{D^{2D}} p_i\left(\frac{1}{d} \mathbf{x}_3\right) \frac{\partial \mathbf{U}_i^{2D}}{\partial \mathbf{x}_j}(\mathbf{x}_1, \mathbf{x}_2), \quad j = 1, 2.$$

The derivative with respect to  $\mathbf{x}_3$  does not involve a differentiation of the functions  $\mathbf{U}_i^{2D}$  but is the derivative of the Legendre polynomials over the coordinate  $\mathbf{x}_3$ , e. g.

$$\frac{\partial \mathbf{U}}{\partial \mathbf{x}_3}(\mathbf{x}) = \sum_{i=0}^{D^{2D}} p'_i\left(\frac{1}{d} \mathbf{x}_3\right) \mathbf{U}_i^{2D}(\mathbf{x}_1, \mathbf{x}_2).$$

Combining these two formulas we get the gradient operator  $\nabla$  for the hierarchical plate model (2.62),

$$\nabla \mathbf{U}(\mathbf{x}) \in \mathbb{R}^{3 \times 3} := \sum_{i=0}^{D^{2D}} \left( \begin{array}{c} p_i\left(\frac{1}{d} \mathbf{x}_3\right) \frac{\partial \mathbf{U}_i^{2D}}{\partial \mathbf{x}_1}(\mathbf{x}_1, \mathbf{x}_2) \mid p_i\left(\frac{1}{d} \mathbf{x}_3\right) \frac{\partial \mathbf{U}_i^{2D}}{\partial \mathbf{x}_2}(\mathbf{x}_1, \mathbf{x}_2) \mid \frac{1}{d} p'_i\left(\frac{1}{d} \mathbf{x}_3\right) \mathbf{U}_i^{2D}(\mathbf{x}_1, \mathbf{x}_2) \end{array} \right), \quad (2.64)$$

where the three entries are column vectors. The first two columns are the gradient of the sum of two-dimensional functions  $\mathbf{U}_i^{2D}$  while the third column is a certain sum of these functions  $\mathbf{U}_i^{2D}$ . We will show in section 5.2.8 how this gradient can be implemented.

**Example 2.5.4** (Plate Gradient for Degree  $D^{2D} = 1$ ). For the case of  $D^{2D} = 1$ , we have a linear ansatz  $\mathbf{x}_3 \mathbf{U}_1^{2D}$  in addition to the midsurface displacement  $\mathbf{U}_0^{2D}$ , e. g.

$$\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{U}_0^{2D}(\mathbf{x}_1, \mathbf{x}_2) + \frac{1}{d} \mathbf{x}_3 \mathbf{U}_1^{2D}(\mathbf{x}_1, \mathbf{x}_2).$$

The differentiation with respect to  $\mathbf{x}_3$  then is simply  $\frac{1}{d} \mathbf{U}_1^{2D}$  and we have the gradient

$$\nabla \mathbf{U}(\mathbf{x}) = \left( \begin{array}{c} \mathbf{U}_{0,1}^{2D} + \mathbf{x}_3 \mathbf{U}_{1,1}^{2D} \mid \mathbf{U}_{0,2}^{2D} + \mathbf{x}_3 \mathbf{U}_{1,2}^{2D} \mid \frac{1}{d} \mathbf{U}_1^{2D}(\mathbf{x}_1, \mathbf{x}_2) \end{array} \right),$$

with  $\mathbf{U}_{i,j}^{2D} := \frac{\partial \mathbf{U}_i^{2D}}{\partial \mathbf{x}_j}(\mathbf{x}_1, \mathbf{x}_2)$ ,  $i = 0, 1$  and  $j = 1, 2$ .

Having this notation, the deformation gradient  $F$  for the hierarchical plate model is the same as before, that is

$$F(\mathbf{x}) = I + \nabla \mathbf{U}(\mathbf{x}).$$

These representations of  $\mathbf{U} = \sum p_i \mathbf{U}_i^{2D}$  and of  $\nabla$  from (2.64) for our hierarchical plate model are now inserted in the previous full three-dimensional model. For



example, the stored energy  $W$  from (2.45) then reads

$$\begin{aligned} W(\mathbf{U}, \mathbf{C}) &= \int_{\Omega} w(I + \nabla \mathbf{U}(\mathbf{x})) \, d\mathbf{x} && + W^{(C)}(\mathbf{U}, \mathbf{C}) \\ &= \int_{\Omega} w\left(I + \nabla \sum p_i \left(\tfrac{1}{d} \mathbf{x}_3\right) \mathbf{U}_i^{2D}(\mathbf{x}_1, \mathbf{x}_2)\right) \, d\mathbf{x} && + W^{(C)}\left(\sum p_i \mathbf{U}_i^{2D}, \mathbf{C}\right). \end{aligned} \quad (2.65)$$

The first order optimality condition (2.46) then becomes

$$\begin{aligned} 0 &= W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\delta \mathbf{U}] \\ &= \int_{\Omega} w_{,F}(I + \nabla \mathbf{U}(\mathbf{x})) [\nabla \delta \mathbf{U}(\mathbf{x})] \, d\mathbf{x} + W_{,\mathbf{U}}^{(C)}\left(\sum \mathbf{U}_i^{2D}, \mathbf{C}\right)[\delta \mathbf{U}], \end{aligned} \quad (2.66)$$

with the test function  $\delta \mathbf{U} = \sum p_i \delta \mathbf{U}_i^{2D}(\mathbf{x})$ . Before we discuss the load in the context of the plate model, we address how to compute the three-dimensional integrals over  $\Omega$ .

### Integration over the Thickness

As we assumed the spatial coordinate  $\mathbf{x}_3$  to be perpendicular to the midsurface  $\Omega^{2D}$ , the integration can be done separately, for example

$$\int_{\Omega} \dots d\mathbf{x} = \int_{\Omega^{2D}} \left( \int_{-d}^d \dots d\mathbf{x}_3 \right) d\mathbf{x}_1 d\mathbf{x}_2. \quad (2.67)$$

The integral over the thickness can be approximated by numerical integration, for example by a Gauss-Legendre quadrature, taken from (Plato, 2004, ch. 6.8) and Zeidler (2013).

**Remark 2.5.5** (Gauss-Legendre quadrature). Let  $f : [-d, d] \rightarrow \mathbb{R}$  be a smooth function. The integral over  $[-d, d]$  can be approximated by the Gauss-Legendre quadrature formula of degree  $D^{\text{GL}}$ , that is

$$\int_{-d}^d f(x) \, dx \approx d \sum_{j=1}^{D^{\text{GL}}} \omega_j f(dx_j), \quad (2.68)$$

with certain integration points (abscissas)  $x_j \in [-1, 1]$  and weights  $\omega_i > 0$ ,  $j = 1, \dots, D^{\text{GL}}$ . The formula already includes the transformation from  $[-d, d]$  to  $[-1, 1]$ . The formulas for the degrees  $D^{\text{GL}} = 2, 3, 4, 5$  can be found in table 2.4.



$D^{\text{GL}}$	$x_j \in [-1, 1]$	$\omega_j > 0$
2	$\pm \frac{1}{3}\sqrt{3}$	1
3	0	$\frac{8}{9}$
	$\pm \frac{1}{5}\sqrt{15}$	$\frac{5}{9}$
4	$\pm \frac{1}{35}\sqrt{525 - 70\sqrt{30}}$	$\frac{1}{36} (18 + \sqrt{30})$
	$\pm \frac{1}{35}\sqrt{525 + 70\sqrt{30}}$	$\frac{1}{36} (18 - \sqrt{30})$
5	0	$\frac{128}{225}$
	$\pm \frac{1}{21}\sqrt{245 - 14\sqrt{70}}$	$\frac{1}{900} (332 + 13\sqrt{70})$
	$\pm \frac{1}{21}\sqrt{245 + 14\sqrt{70}}$	$\frac{1}{900} (332 - 13\sqrt{70})$

TABLE 2.4. Abscissas and weights for the Gauss-Legendre quadrature formulas up to degree  $D^{\text{GL}} = 5$  taken from Weisstein (2014), Zwillinger (2003).

We apply this formula to approximate the integrals of the weak formulation (2.66). For the first term  $W_U^{(T)}$ , we get

$$\begin{aligned}
& \int_{\Omega} w_{,F}(I + \nabla \mathbf{U}(\mathbf{x})) [\nabla \delta \mathbf{U}] \, d\mathbf{x} \\
&= \int_{\Omega^{2D}} \left( \int_{-d}^d w_{,F}(I + \nabla \mathbf{U}(\mathbf{x})) [\nabla \delta \mathbf{U}(\mathbf{x})] \, dx_3 \right) d\mathbf{x}_1 d\mathbf{x}_2 \\
&\approx \int_{\Omega^{2D}} \left( d \sum_{j=1}^{D^{\text{GL}}} \omega_j w_{,F}(I + \nabla \mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, x_j)) [\nabla \delta \mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, x_j)] \right) d\mathbf{x}_1 d\mathbf{x}_2.
\end{aligned} \tag{2.69}$$

The evaluation of  $\mathbf{U}$ , or  $\delta \mathbf{U}$ , for a certain  $\mathbf{x}_3 = dx_j$  is actually calculating the Legendre polynomials at  $x_i$ , e. g.  $p_i(x_i)$  and computing the sum over all  $U_i^{2D}$ . Section 5.2.8 will demonstrate how this can be done in the finite element library FENICS. Other terms can be approximated analogously.



### Ansatz for the Loads

The load  $\mathbf{C}$  will become the control in the later optimal control, therefore we need an ansatz for it as well. The hierarchical plate model was motivated in cases where the domain  $\Omega$  is planar and thin. The inner pressure  $t$  and the fiber tension  $m$  were mainly motivated by biological systems, but both phenomena are unlikely to occur on thin plates. Therefore we drop these two load types in the context of the plate model. In technical applications it is rather difficult to apply volume loads that vary over the thickness coordinate  $\mathbf{x}_3$ , so the volume load is assumed to be constant over the thickness, e. g.

$$\mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2).$$

Applying the Gauss-Legendre formula on the energy functional  $W^{(\mathbf{f})}$  for the volume load  $\mathbf{f}$ , which is constant over  $\mathbf{x}_3$ , we get

$$\begin{aligned} W^{(\mathbf{f})}(\mathbf{U}, \mathbf{f}) &= \int_{\Omega} \mathbf{f}(\mathbf{x})^{\top} \mathbf{U}(\mathbf{x}) \, d\mathbf{x} \\ &= \int_{\Omega^{2D}} \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2)^{\top} \left( \sum_{i=1}^{D^{GL}} \omega_i \mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, x_i) \right) d\mathbf{x}_1 d\mathbf{x}_2. \end{aligned} \quad (2.70)$$

The first derivative can be calculated analogously.

Next, we consider a boundary load  $\mathbf{g}$ . A load acting on the rim  $\partial\Omega^{2D} \times [-d, d]$  would be technically challenging and the total force would scale with the thickness  $2d$  which was assumed to be very small. Therefore we drop boundary loads on the rim and assume them to act on the upper or lower surface, e. g.  $\Gamma^N = \Omega^{2D} \times \{-d, d\}$ . Let  $\mathbf{g}(\cdot, \cdot, -d)$  be the boundary load that acts on the lower surface  $\Omega^{2D} \times \{-d\}$  and  $\mathbf{g}(\cdot, \cdot, d)$  be the one acting on the upper surface  $\Omega^{2D} \times \{d\}$ . The energy functional then reads

$$\begin{aligned} W^{(\mathbf{g})}(\mathbf{U}, \mathbf{g}) &= \int_{\Gamma^N} \mathbf{g}(\mathbf{x})^{\top} \mathbf{U}(\mathbf{x}) \, dS \\ &= \int_{\Omega^{2D}} \mathbf{g}(\mathbf{x}_1, \mathbf{x}_2, -d)^{\top} \mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, -d) + \mathbf{g}(\mathbf{x}_1, \mathbf{x}_2, d)^{\top} \mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, d) d\mathbf{x}_1 d\mathbf{x}_2. \end{aligned} \quad (2.71)$$

In section 5.2.8 we will give details on how we implemented the plate model and its ansatz functions.

### Comparison to other Plate Models

The **Mindlin-Reissner** plate model is based on the early works of Mindlin (1945), Reissner (1944) and Reissner (1945), who suggested this model independently. The



ansatz in the case of linear elasticity reads

$$\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \left( \mathbf{x}_3[\mathbf{U}_1^{2D}]_1(\mathbf{x}_1, \mathbf{x}_2), \mathbf{x}_3[\mathbf{U}_1^{2D}]_2(\mathbf{x}_1, \mathbf{x}_2), [\mathbf{U}_0^{2D}]_3(\mathbf{x}_1, \mathbf{x}_2) \right)^\top, \quad (2.72)$$

taken from Paumier and Raoult (1997). There are additional assumptions which however contradict the ansatz for  $\mathbf{U}$ . This is not uncommon for plate models and demands caution in the modeling. If we drop a certain assumption, Paumier and Raoult (1997) have shown that this model is consistent with the hierarchical plate model for  $d \rightarrow 0$ . We can also see, that the ansatz for the Mindlin-Reissner plate model can be extended to the hierarchical plate model with  $\mathbf{U} = \mathbf{U}_0^{2D} + \mathbf{x}_3 \mathbf{U}_1^{2D}$  for larger deformations. Thus, possible deformations of the Mindlin-Reissner model are included in the hierarchical plate model with the degree  $D^{2D} \geq 1$ .

The hierarchical plate model also includes possible displacements of the popular **Kirchhoff-Love-theory**, based on the works of Kirchhoff (1850) and Love (1888), see (Ciarlet, 1997, ch. 1.7). This model uses the assumption that the normal of the midsurface stays orthogonal to the midsurface during the deformation, that means that shear is not possible. Additionally, the thickness is assumed to stay constant, hence no change in the thickness is possible either<sup>7</sup>. This yields the ansatz for the deformation

$$\hat{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathbf{x} + \mathbf{U}^{2D}(\mathbf{x}_1, \mathbf{x}_2) + \mathbf{x}_3 \hat{\mathbf{n}}^{2D}(\mathbf{x}_1, \mathbf{x}_2) \quad (\mathbf{x}_1, \mathbf{x}_2) \in \Omega^{2D}, \mathbf{x}_3 \in (-d, d),$$

where  $\hat{\mathbf{n}}^{2D}(\mathbf{x}_1, \mathbf{x}_2)$  is the normal of the deformed midsurface at the material point  $\hat{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2, 0)$ , which can be computed by

$$\hat{\mathbf{n}}_0^{2D} = \frac{\mathbf{U}_{,\mathbf{x}_1}^{2D} \times \mathbf{U}_{,\mathbf{x}_2}^{2D}}{\|\mathbf{U}_{,\mathbf{x}_1}^{2D} \times \mathbf{U}_{,\mathbf{x}_2}^{2D}\|}.$$

These two assumptions seem reasonable for very thin plates and deformations where the change of thickness is negligible. However, it is not clear how well these assumptions can be justified for large deformations. Comparing the ansatz functions of the Kirchhoff hypothesis and the hierarchical plate model (2.62), one can see that possible deformations under the Kirchhoff hypothesis can also be described in the hierarchical plate model. This does not hold true the other way around, so the hierarchical plate model allows more deformations and hence is less restrictive. Of course, this comes at the price of introducing new unknowns. However, the Kirchhoff hypothesis has its own difficulties. The displacement  $\mathbf{U}$  of the Kirchhoff hypothesis already depends on the first derivatives of  $\mathbf{U}^{2D}$  and hence the deformation gradient  $F$  will depend on the second derivatives of  $\mathbf{U}$ . To solve the problem with a finite element method, the discretization requires elements which are globally continuously differentiable, like Bogner-Fox-Schmidt elements or MITC elements, see for example

<sup>7</sup>There are extensions of the Kirchhoff-model that allow a change of the thickness by adding an additional unknown for the change of thickness.



Arnold and Falk (1989), Lee and Bathe (2010) and Rückert (2013) and references therein.



# 3 Numerical Methods to Solve the Forward Problem

## Contents

3.1	Newton's Method	48
3.2	Globalization by a Line Search	49
3.3	Krylov Subspace Methods	56
3.4	Discretization	61
3.5	Preconditioner and Multigrid Method	65

This chapter is devoted to solution methods for the elastic forward problem. That means we seek to find the unknown displacement field  $\mathbf{U}$  which is determined by the energy minimization (2.45). The formal first order optimality condition is a nonlinear partial differential equation which we solve with Newton's method. In section 3.1, we formulate Newton's method in function space, rather than discretizing it first and then solving the discretized problem. However, we still face the problem of a possible divergence of Newton's method, which is often observed in elasticity with large deformations. We will discuss possible methods to overcome this problem in section 3.2. In order to apply Newton's method, we repeatedly need to solve the Newton system which is a linear equation. In section 3.3, we present two Krylov subspace methods. As the underlying operator  $A$  is self-adjoint, we can employ a minimal-residual (MINRES) method to solve it. Furthermore, as we know that the solution  $\mathbf{U}$  is a local minimum of the energy minimization problem and the operator  $A$  is the second derivative of the stored energy, we can also use a truncated conjugate gradient (CG) method for the Newton system. Again, both methods are formulated in a function space setting, which will offer us some insights on norms for the stopping criteria and the role of the preconditioner.

At this point, we will briefly introduce the discretization by a finite element method (FEM) in section 3.4. The FEM is a broadly used discretization technique for many problems that require solving partial differential equations (PDE). Next, in section 3.5, we show how the multigrid method is incorporated as a preconditioner for the Krylov subspace method and the idea of nested iterations.



### 3.1 Newton's Method

We seek to find a local minimum  $\mathbf{U}_\star \in \mathcal{U}$  of the energy minimization problem (2.45) for a given load  $\mathbf{C}$ , that is

$$\mathbf{U}_\star = \arg \min_{\mathbf{U} \in \mathcal{U}} W(\mathbf{U}, \mathbf{C})$$

with  $W(\mathbf{U}, \mathbf{C}) = W^{(T)}(\mathbf{U}) + W^{(C)}(\mathbf{U}, \mathbf{C})$ ,

with the necessary first order optimality condition which is the (nonlinear) variational equality (2.46), that is

$$\begin{aligned} 0 &= W_{,U}(\mathbf{U}_\star, \mathbf{C})[\delta \mathbf{U}] \\ &= W_{,U}^{(T)}(\mathbf{U}_\star)[\delta \mathbf{U}] + W_{,U}^{(C)}(\mathbf{U}_\star, \mathbf{C})[\delta \mathbf{U}], \quad \forall \delta \mathbf{U} \in \mathcal{U}. \end{aligned}$$

There are two approaches now:

- *Discretize-then-Optimize*: We first introduce a finite-dimensional function space  $\mathcal{U}_h$  by a finite element method and then consider the discretized optimization problem as a problem in the coefficient vector in  $\mathbb{R}^n$  and solve it, for example, with Newton's method in  $\mathbb{R}^n$ .
- *Optimize-then-Discretize*: We first formulate Newton's method in the infinite dimensional function space  $\mathcal{U}$  and solve the (linear) Newton system by introducing a finite dimensional function space  $\mathcal{U}_h$ , solving the resulting linear system of equations.

For some problems, the two approaches might yield the same algorithm at the end (if certain scalar products are chosen right) and it is said that discretization and optimization commute. However, this is not always the case. We choose the second approach, *Optimize-then-Discretize*, because it seems more natural and may offer helpful insights into the method, e. g. how to choose scalar products or norms. This will also help to get an algorithm that is mesh independent, that means the algorithm behaves similarly on refined meshes, e. g. with respect to iteration numbers.

We consider the problem (2.46),

$$0 = W_{,U}(\mathbf{U}_\star)[\delta \mathbf{U}], \quad \forall \delta \mathbf{U} \in \mathcal{U},$$

and want to solve it by Newton's method. Let  $\mathbf{U}_0 \in \mathcal{U}$  be an initial guess, e. g.  $\mathbf{U}_0 \equiv \mathbf{0}$ . The Newton step in the  $(k+1)$ -st iteration is

$$\mathbf{U}_{k+1} = \mathbf{U}_k + \alpha \Delta \mathbf{U}_k, \tag{3.1}$$

with a step length  $\alpha > 0$  and a search direction  $\Delta \mathbf{U}_k$  defined by the (linear) Newton system

$$W_{,UU}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}, \Delta \mathbf{U}_k] = -W_{,U}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}], \quad \forall \delta \mathbf{U} \in \mathcal{U}. \tag{3.2}$$



In more details, that is

$$\begin{aligned} W_{,U}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}] &= W_{,U}^{(T)}(\mathbf{U}_k)[\delta \mathbf{U}] + W_{,U}^{(C)}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}] \\ &= \int_{\Omega} w_{,F}(I + \nabla \mathbf{U}_k)[\nabla \delta \mathbf{U}] \, d\mathbf{x} + W_{,U}^{(C)}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}], \end{aligned}$$

and

$$\begin{aligned} W_{,UU}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}, \Delta \mathbf{U}_k] &= W_{,UU}^{(T)}(\mathbf{U}_k)[\delta \mathbf{U}, \Delta \mathbf{U}_k] + W_{,UU}^{(C)}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}, \Delta \mathbf{U}_k] \\ &= \int_{\Omega} w_{,FF}(I + \nabla \mathbf{U}_k)[\nabla \delta \mathbf{U}, \nabla \Delta \mathbf{U}_k] \, d\mathbf{x} + W_{,UU}^{(C)}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}, \Delta \mathbf{U}_k]. \end{aligned}$$

The last term  $W_{,UU}^{(C)}$  vanishes if the load  $\mathbf{C}$  is a volume load  $\mathbf{f}$  or a boundary load  $\mathbf{g}$ , see the formulas in table 2.1. We summarize Newton's method to solve the variational problem (2.46) in algorithm 3.1.1. A natural stopping criterion would be a relative or absolute tolerance for the derivative  $W_{,U}(\mathbf{U}_k, \mathbf{C})$ , e. g.

$$\frac{\|W_{,U}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*}}{\|W_{,U}(\mathbf{U}_0, \mathbf{C})\|_{\mathcal{U}^*}} \leq \text{rTol} \quad \text{or} \quad \|W_{,U}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*} \leq \text{aTol}, \quad (3.3)$$

with a user defined relative tolerance  $\text{rTol} > 0$  and an absolute tolerance  $\text{aTol} > 0$ .

**Algorithm 3.1.1** (Newton's method to solve  $W_{,U}(\mathbf{U}, \mathbf{C}) = 0$ ).

**Input:** initial  $\mathbf{U}_0$ , energy  $W$ , load  $\mathbf{C}$ ,  $k = 0$

**Output:** state  $\mathbf{U}$

- 1: **while** ( $\|W_{,U}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*}$  is too large) **do**
- 2:   get  $\Delta \mathbf{U}_k$  by solving  $W_{,UU}(\mathbf{U}_k, \mathbf{C})[\delta \mathbf{U}, \Delta \mathbf{U}_k] = -W_{,U}(\mathbf{U}_k, \mathbf{C})[\Delta \mathbf{U}_k]$
- 3:   choose step length  $\alpha_k$  by a line search
- 4:   set  $\mathbf{U}_{k+1} := \mathbf{U}_k + \alpha \Delta \mathbf{U}_k$
- 5:   set  $k := k + 1$
- 6: **end while**
- 7: **return** displacement  $\mathbf{U}_k$

## 3.2 Globalization by a Line Search

As known, a standard Newton's method with a fixed step length  $\alpha = 1$  cannot guarantee convergence in general. This holds true in particular for large elastic deformations, as shown in the following example.



**Example 3.2.1** (Thick Plate). Let  $\Omega = (0, 2) \times (0, 2) \times (0, \frac{1}{10})$  be a plate. The plate is clamped at the rim

$$\Gamma^D = \{\mathbf{x} \in \partial\Omega : \mathbf{x}_1 \in \{0, 2\} \text{ or } \mathbf{x}_2 \in \{0, 2\}\}.$$

A volume load  $\mathbf{f} = (0, 0, \frac{1}{4})^\top$  acts on the whole body  $\Omega$ . We choose a value for Young's modulus of  $E = 1$  and a Poisson ratio of  $\nu = 0.3$ , which results in the material parameters

$$a = \frac{25}{208}, \quad b = \frac{15}{208}, \quad c = \frac{15}{208}, \quad d = \frac{35}{52}, \quad e = -\frac{135}{208},$$

for the polyconvex energy density  $w$  from (2.18).

**Remark 3.2.2** (Non-convergence of Newton's method). We consider the example 3.2.1 and use Newton's method on it. The start at  $\mathbf{U}_0 \equiv \mathbf{0}$  results in the first iterate  $\mathbf{U}_1 = \Delta\mathbf{U}_0$  which is shown in figure 3.1. The following iterates fail to converge.

The reason of the divergence lies in the local negative volume change  $J(\mathbf{x}) < 0$  for some  $\mathbf{x} \in \Omega$ . As the first Newton system at  $\mathbf{U}_0 \equiv \mathbf{0}$  is equivalent to the model of linear elasticity, the result  $\delta\mathbf{U}_0$  might not satisfy the positivity of the local volume change  $J = \det F > 0$  for very large deformations. If the step length is chosen as  $\alpha = 1$ , the new iterate  $\mathbf{U}_1 = \delta\mathbf{U}_0$  has a local negative volume change and is physically inadmissible and hence the elastic model cannot be expected to yield a reasonable result anymore. While the stored energy  $W$  is  $\infty$  for local negative volume changes, the forms for the derivatives  $W_{,U}$  and  $W_{,UU}$  might be still evaluable at  $\mathbf{U}_1$ . Hence the negative volume might not be detected in the algorithm and the Newton solver ends in divergence.

There are various ways to overcome this problem and get convergence of Newton's method. As we try to solve the energy minimization problem (2.45), these techniques can also be viewed as globalization methods to ensure the convergence of Newton's method as an optimization method.

### Incremental Method and Homogenization

A very common and simple method is an incremental increase of the load  $\mathbf{C}$ , that means we scale the load  $\mathbf{C}_s := s\mathbf{C}$  with a parameter  $s \in [0, 1]$ . We assume we know a  $s_0 \in [0, 1]$  such that Newton's method converges to the intermediate solution  $\mathbf{U}_{s_0}$ . From there we increase the increment and start Newton's method at the initial guess  $\mathbf{U}_{s_0}$ . We repeat this procedure and gain a sequence of parameters  $0 \leq s_0 < s_1 < \dots < s_n = 1$  and eventually can solve the original problem. This algorithm is not to be understood as a line search inside Newton's method but rather a step-by-step solving with Newton's method. Algorithm 3.2.3 shows the incremental strategy with a simple trial and error strategy, that means we try to



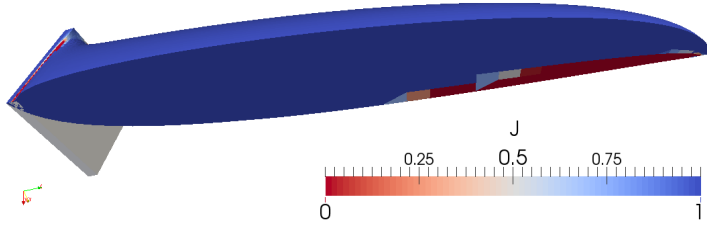


FIGURE 3.1. The first iterate of a standard Newton's method for example 3.2.1. Only the quarter  $(0, 1) \times (0, 1) \times (0, \frac{1}{10})$  of the domain  $\Omega$  is plotted and the  $x_3$ -axis points to the right. The gray plate is a quarter of the undeformed domain  $\Omega$  and the blue domain the solution of the first Newton step with a step length  $\alpha = 1$ . The red areas at the clamping boundary  $\Gamma^D$  and in the center of the plate indicate a negative local volume change  $J(\mathbf{x}) < 0$ .

solve for  $s_k$  and if Newton's method does not converge, we decrease  $s_k$  until we get convergence. From there we start to increase  $s_k$  again.

**Algorithm 3.2.3** (Incremental strategy).

**Input:** initial  $s_0 = 0$ , all loads in dependency of  $s$

**Output:** state  $\mathbf{U}$  for  $s = 1$

```

1: set  $k := 0$ 
2: while  $s < 1$  do
3:   try to solve the forward system for  $s_k$ 
4:   if (successful) then
5:     choose a new  $s_{k+1}$  such that  $s_k < s_{k+1} \leq 1$  and set  $k := k + 1$ 
6:   else
7:     choose  $\bar{s}_k$  such that  $s_{k-1} < \bar{s}_k < s_k$  and set  $s_k := \bar{s}_k$ 
8:   end if
9: end while
10: return displacement  $\mathbf{U}$ 

```

The incremental method is quite easy to implement, however the question how to choose  $s_k$  remains open. A fixed series of parameters  $0 < s_0 < s_1 < \dots < s_n = 1$  is not practical for the later optimal control, therefore an automatization is required. The simple trial and error method from the algorithm might work in general, though homogenization techniques might improve that search. Still, this could be very



expensive to do, as many Newton steps might be wasted just to find out that a current parameter  $s_k$  is not sufficiently small.

#### Simple Backtracking to ensure $J > 0$

Another straightforward idea would be to check the positivity of the local volume change  $J = \det F > 0$  for a current iterate  $\mathbf{U}_{k+1} = \mathbf{U}_k + \alpha \Delta \mathbf{U}_k$  with  $\alpha = 1$ . If the check fails, the step length  $\alpha$  is decreased until the positivity of volume change  $J > 0$  is satisfied for all points  $\mathbf{x} \in \Omega$ . This ensures a physically admissible deformation  $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{U}$  and hence is a remedy of non-convergence caused by negative local volume changes. Algorithm 3.2.4 shows this method. However it might happen that the Newton step is still too large even though the volume change is positive, resulting overall in a poor performance of Newton's method. As the new iterate can still be very close to a negative volume change in a certain point  $\mathbf{x}$ , it might not be a good start for the next Newton iteration.

#### Algorithm 3.2.4 (Simple backtracking).

**Input:** search direction  $\Delta \mathbf{U}_k$ , old iterate  $\mathbf{U}_k$ ,  $s_\alpha \in (0, 1)$

**Output:** step length  $\alpha$

```

1: set  $\alpha := 1$ ,  $k := 0$ , done:=FALSE
2: while done=FALSE do
3:   set  $\mathbf{U} := \mathbf{U}_k + \alpha \Delta \mathbf{U}_k$ 
4:   check for neg. local vol. changes  $J(\mathbf{x}) = \det(I + \nabla \mathbf{U}(\mathbf{x})) \forall \mathbf{x} \in \Omega$ .
5:   if (neg. vol. detected) then
6:     set  $\alpha := s_\alpha \alpha$ 
7:   else
8:     set done:=TRUE
9:   end if
10: end while
11: return displacement  $\mathbf{U} = \mathbf{U}_k + \alpha \Delta \mathbf{U}_k$ 

```

#### Armijo-Backtracking on the Stored Energy $W$

As we have the energy minimization problem (2.45) and apply Newton's method as an optimization method on it, a line search naturally appears in the algorithm as a globalization technique. A simple choice could be an Armijo-backtracking line search to ensure that the stored energy  $W$  decreases with each new iterate, see (Nocedal and Wright, 2006, ch. 3.1). As the stored energy  $W$  is only finite for displacements with positive local volume change, the Armijo condition also ensures the positivity of the local volume change  $J = \det F > 0$ . For an easier reading of algorithm 3.2.5, we abbreviate the function  $W$  along the search direction  $\Delta \mathbf{U}_k$ , that is

$$\varphi(\alpha) := W(\mathbf{U}_k + \alpha \Delta \mathbf{U}_k, \mathbf{C})$$

$$\varphi'(0) := W_{,\mathbf{U}}(\mathbf{U}_k, \mathbf{C})[\Delta \mathbf{U}].$$



**Algorithm 3.2.5** (Armijo-backtracking on the stored energy  $W$ ).

**Input:** Energy function  $\varphi$  along the search direction, parameter  $\sigma \in (0, \frac{1}{2})$ ,  $s_\alpha \in (0, 1)$

**Output:** step length  $\alpha$

```

1: set  $\alpha := 1$ ,  $k := 0$ , done:=FALSE
2: compute the stored energy  $\varphi(0)$  and the slope  $\varphi'(0)$ 
3: while done=FALSE do
4:   set  $\mathbf{U} := \mathbf{U}_k + \alpha\Delta\mathbf{U}_k$  and compute the stored energy  $\varphi(\alpha) := W(\mathbf{U}, \mathbf{C})$ 
5:   if ( $\varphi(\alpha) > \varphi(0) + \alpha\sigma\varphi'(0)$ ) then
6:     set  $\alpha := s_\alpha\alpha$ 
7:   else
8:     set done:=TRUE
9:   end if
10: end while
11: return displacement  $\mathbf{U} = \mathbf{U}_k + \alpha\Delta\mathbf{U}_k$ 

```

This method only works if a stored energy is available, which however is assumed in this work. Another problem not discussed in this work would be inhomogeneous Dirichlet boundary conditions (BCs), as they are included in the state space  $\mathcal{U}$ .<sup>1</sup> Hence the initial guess  $\mathbf{U}_0$  requires the inhomogeneous Dirichlet BCs as well, since otherwise the Armijo-condition might fail in the very first iteration (inhomogeneous Dirichlet BCs normally increase the stored energy).

### Guiding Function

The basic idea of this method is introducing a so-called *guiding criterion* with a *guiding function*  $G : \mathcal{U} \rightarrow [0, \infty]$  whose task is twofold:

- (1) Judge if a step yields a physically admissible iterate in order to ensure that the elastic model can be used (just like *Checking Positive Volume*).
- (2) Damp very large Newton steps if necessary, which stabilizes Newton's method. However it should be avoided that it impairs the local convergence rate.

Both goals can be achieved by a backtracking line search with the guiding criterion

$$G(\mathbf{U}_k + \alpha\Delta\mathbf{U}_k) \leq G(\mathbf{U}_k) + \beta|\Omega|, \quad (3.4)$$

<sup>1</sup>It is not known yet how to add inhomogeneous Dirichlet boundary conditions to a stored energy. Penalization terms might be an idea but the choice of the penalization term and parameter remains. Normally one has to perform the energy minimization in the subspace that satisfies the inhomogeneous Dirichlet BCs, but this requires an initial state  $\mathbf{U}_0$  with the inhomogeneous Dirichlet BCs. This is difficult in case of “large” inhomogeneous Dirichlet BC data.



the guiding function

$$G(\mathbf{U}) = \begin{cases} \int_{\Omega} |\ln(\det(I + \nabla \mathbf{U}(\mathbf{x})))| \, d\mathbf{x} \\ \infty & \text{if the integral is not defined} \end{cases}, \quad (3.5)$$

and a guiding parameter  $\beta > 0$ . The step length  $\alpha$  starts at 1 and is decreased by a certain factor, e.g.  $s_\alpha = \frac{1}{2}$ , if the criterion is not satisfied, see algorithm 3.2.6.

**Algorithm 3.2.6** (Backtracking for guiding criterion (3.4)).

**Input:** Current iterate  $\mathbf{U}_k$ , search direction  $\Delta \mathbf{U}_k$ ,  $s_\alpha \in (0, 1)$

**Output:** step length  $\alpha$

```

1: set  $\alpha := 1$ ,  $k := 0$ , done:=FALSE
2: compute  $G(\mathbf{U}_k)$  and the offset  $\beta|\Omega|$ 
3: while done=FALSE do
4:   set  $\mathbf{U} := \mathbf{U}_k + \alpha \Delta \mathbf{U}_k$  and compute  $G(\mathbf{U})$ 
5:   if ( $G(\mathbf{U}) > G(\mathbf{U}_k) + \beta|\Omega|$ ) then
6:     set  $\alpha := s_\alpha \alpha$ 
7:   else
8:     set done:=TRUE
9:   end if
10: end while
11: return displacement  $\mathbf{U} = \mathbf{U}_k + \alpha \Delta \mathbf{U}_k$ 

```

First of all, the guiding criterion (3.4) excludes physically inadmissible deformations. If an iterate  $\mathbf{U}_{k+1} = \mathbf{U}_k + \alpha \Delta \mathbf{U}_k$  has a point  $\mathbf{x} \in \Omega$  with negative volume change  $\det(I + \nabla \mathbf{U}(\mathbf{x})) < 0$ , the logarithm  $\ln \det(I + \nabla \mathbf{U}(\mathbf{x}))$  in the in guiding function is not defined. Therefore, the value of the guiding function is  $G(\mathbf{U}_{k+1}) = \infty$ . As we assume to start with a physically admissible initial guess  $\mathbf{U}_0$  with  $G(\mathbf{U}_0) < \infty$ , the positive volume change is preserved during Newton's method because a physically inadmissible iterate  $\mathbf{U}_{k+1}$  could not satisfy the criterion.

The guiding criterion also damps Newton's method for very large steps far away from the undeformed state  $\mathbf{U} \equiv \mathbf{0}$  (states near the undeformed state seem not to cause trouble). Let us assume we start Newton's method at  $\mathbf{U}_0 \equiv \mathbf{0}$  with

$$G(\mathbf{U}_0) = \int_{\Omega} |\ln(\det(I + \nabla \mathbf{0}(\mathbf{x})))| \, d\mathbf{x} = \int_{\Omega} |\ln 1| \, d\mathbf{x} = 0.$$

As the guiding function value is non-negative, any iterate  $\mathbf{U} \neq \mathbf{0}$  would have  $G(\mathbf{U}) \geq G(\mathbf{U}_0) = 0$ , with equality only if the volume change is  $J \equiv 1$ , which would mean that the iterate  $\mathbf{U}_1$  behaves like an incompressible material. This is very unlikely for the material models and parameters in this work. Hence we can expect that the guiding functional  $G$  increases with the first iterate  $\mathbf{U}_1$ . The damping now occurs by limiting this increase by the term  $\beta|\Omega|$ . This strategy does not impair the local



convergence behavior of Newton's method. If Newton's method converges, the norm of the search direction  $\Delta \mathbf{U}_k$  becomes smaller. As a consequence, the difference  $G(\mathbf{U}_{k+1}) - G(\mathbf{U}_k)$  decreases too and eventually is smaller than the offset  $\beta|\Omega|$ . We illustrate the behavior of this line search strategy with the following example.

**Example 3.2.7** (Compression of a Cube). Let  $\Omega$  be the cube  $(-1, 1)^3$  and  $\mathbf{g} = g_0 \mathbf{n}$ ,  $g_0 \in \mathbb{R}$  be an orthogonal boundary load on the whole boundary  $\Gamma^N = \partial\Gamma$ . As we do not impose any Dirichlet boundary conditions, the solution is not unique due to rigid body movements. However we will neglect these movements and consider a special (analytical) solution for this problem. First, we show that the compression

$$\hat{\mathbf{x}}(\mathbf{x}) = \gamma \mathbf{x}, \text{ or } \mathbf{U}_\gamma(\mathbf{x}) = (\gamma - 1)\mathbf{x}, \quad \gamma > 0, \quad (3.6)$$

with a certain constant  $\gamma = \gamma(g_0) \in \mathbb{R}$ , solves the energy minimization, that is the first-order optimality condition  $W_{,\mathbf{U}}(\mathbf{U}_\gamma) = 0$ .

*Proof.* The deformation gradient for (3.6) is simply  $F = \gamma I$ . Due to the isotropy of the elastic material behavior and frame indifference (2.19), the stress tensor is  $W_{,\mathbf{U}}(\mathbf{U}_\gamma) = c_\gamma I$ , with the constant  $c_\gamma = 2a\gamma + 8b\gamma^3 + 2c\gamma^5 - d\gamma^{-1}$  for the energy density (2.18). Inserting this into the first-order optimality condition (2.46) yields

$$\int_{\Omega} c_\gamma I : (\nabla \delta \mathbf{U}(\mathbf{x})) \, d\mathbf{x} - \int_{\partial\Omega} g_0 \delta \mathbf{U}(\mathbf{x})^\top \mathbf{n}(\mathbf{x}) \, dS = 0, \quad \forall \mathbf{U} \in \mathcal{U}.$$

The term  $I : \nabla \delta \mathbf{U} = \text{tr}(\nabla \delta \mathbf{U})$  can be rewritten as  $\text{div } \delta \mathbf{U}$ . We can now choose the constant  $\gamma$  such that  $c_\gamma = g_0$  because

$$c_\gamma = 2a\gamma + 8b\gamma^3 + 2c\gamma^5 - d\gamma^{-1}$$

is a monotone increasing and thus bijective function  $c_\gamma : (0, \infty) \rightarrow \mathbb{R}$  for  $a, b, c, d > 0$ . Our first-order optimality condition then reads

$$g_0 \int_{\Omega} \text{div } \delta \mathbf{U}(\mathbf{x}) \, d\mathbf{x} - g_0 \int_{\partial\Omega} \delta \mathbf{U}(\mathbf{x})^\top \mathbf{n}(\mathbf{x}) \, dS = 0, \quad \forall \mathbf{U} \in \mathcal{U},$$

which is Green's identity.

Let us consider the more important case with  $g_0 < 0$ , e. g. the cube is compressed, that is  $J = \gamma^3 < 1$ . Negative volume changes might occur in the linearized Newton steps if the load  $g_0 < 0$  is small enough. Starting at  $\mathbf{U}_0 \equiv 0$ , the first Newton step would be  $\Delta \mathbf{U}_0(\mathbf{x}) = (\gamma_0 - 1)\mathbf{x}$  (the linearized system inherits the isotropy and reference invariance of the nonlinear model and hence the step  $\delta \mathbf{U}_0$  differs only in the constant  $\gamma_0$ .) Unlike the constant  $\gamma > 0$ , the constant  $\gamma_0$  depends (affine-) linearly on  $g_0$  and thus might be negative, which is ruled out now by the guiding criterion. Following Newton's method will give a sequence of iterates  $\mathbf{U}_k$ ,  $\mathbf{U}_k(\mathbf{x}) = (\gamma_k - 1)\mathbf{x}$ .



Inserting  $\mathbf{U}_k$  into the guiding function (3.5) yields

$$G(\mathbf{U}_k) = \int_{\Omega} |\ln \gamma_k| \, d\mathbf{x}.$$

As the body is compressed, that is  $\gamma_k < 1$ , the guiding criterion (3.4) reads

$$\begin{aligned} G(\mathbf{U}_{k+1}) - G(\mathbf{U}_k) &\leq \beta |\Omega| \\ \int_{\Omega} -\ln(\gamma_{k+1}) \, d\mathbf{x} - \int_{\Omega} -\ln(\gamma_k) \, d\mathbf{x} &\leq \beta |\Omega| \\ \Leftrightarrow \quad \ln \frac{\gamma_k}{\gamma_{k+1}} &\leq \beta \quad \Leftrightarrow \quad \gamma_{k+1} \geq \gamma_k e^{-\beta}. \end{aligned}$$

This means that the constant  $\gamma_{k+1}$  of the new iterate  $\mathbf{U}_{k+1}$  is bounded below by the old constant multiplied by a factor  $e^{-\beta} < 1$ . In the case of an expanding body, instead of compressing, the factor would be  $e^{\beta}$ . Hence the volume change is bounded below and above. Of course, the uniform compression is a very simple example and does not account for richer deformations. Still, the guiding function (3.5) can be used for more complex problems, as it is a quantity to measure an “average” logarithmic volume change.

A numerical experiment on the different globalization techniques can be found in 6.1.1.

### 3.3 Krylov Subspace Methods

Before we discuss the discretization of the problem in the next section, we briefly present two Krylov subspace methods, namely the *conjugate gradient method* (CG method), introduced by Hestenes and Stiefel (1952), and the *minimal residual method* (MINRES-method), introduced by Paige and Saunders (1975) in the Hilbert space  $\mathcal{U}$ .

We seek to solve the Newton system (3.2), that is the variational equality

$$W_{,\mathbf{U}\mathbf{U}}(\mathbf{U}_k, \mathbf{C})[\delta\mathbf{U}, \Delta\mathbf{U}_k] = -W_{,\mathbf{U}}(\mathbf{U}_k, \mathbf{C})[\delta\mathbf{U}], \quad \forall \delta\mathbf{U} \in \mathcal{U}.$$

The Hilbert space  $\mathcal{U}$  has the inner product  $(\cdot, \cdot)_{\mathcal{U}}$  and its dual space is  $\mathcal{U}^*$ , while  $[\cdot, \cdot]_{\mathcal{U}^*, \mathcal{U}}$  denotes the duality pairing. We introduce the linear bounded operator  $A \in \mathcal{L}(\mathcal{U}, \mathcal{U}^*)$ , which is defined by

$$[Ax, y]_{\mathcal{U}^*, \mathcal{U}} := W_{,\mathbf{U}\mathbf{U}}(\mathbf{U}_k, \mathbf{C})[x, y], \quad x, y \in \mathcal{U}. \quad (3.7)$$

Since the second derivatives of  $W$  are symmetric, the operator  $A$  is self-adjoint, that is

$$[Ax, y]_{\mathcal{U}^*, \mathcal{U}} = [Ay, x]_{\mathcal{U}^*, \mathcal{U}}.$$



The right-hand side  $b \in \mathcal{U}^*$  is defined by

$$[b, x]_{\mathcal{U}^*, \mathcal{U}} := -W_{\mathcal{U}}(\mathbf{U}_k, \mathbf{C})[x], \quad x \in \mathcal{U}. \quad (3.8)$$

Both,  $A$  and  $b$ , depend on the current iterate  $\mathbf{U}_k$  of Newton's method. We now rewrite the Newton system (3.2) as the linear operator equation

$$Ax = b \in \mathcal{U}^*, \quad (3.9)$$

with the solution  $x := \Delta \mathbf{U} \in \mathcal{U}$ . In order to (approximately) solve (3.9), we apply a Krylov subspace method. As the operator  $A$  is self-adjoint, a MINRES-method can be applied, or a CG method can be used to solve the problem if the operator is positive definite (coercive), that is there exists a constant  $\underline{c} > 0$  such that

$$[Ax, x] \geq \underline{c} \|x\|_{\mathcal{U}}^2, \quad \forall x \in \mathcal{U}. \quad (3.10)$$

The operator  $A$  cannot be expected to be positive definite for all iterates  $\mathbf{U}_k \in \mathcal{U}$ , but we can still employ a *truncated* CG method inside Newton's method. More details will be given later, let us first start with the standard CG method.

### CG Method

We assume the operator  $A \in \mathcal{L}(\mathcal{U}, \mathcal{U}^*)$  to be self-adjoint and positive definite. The CG method can be derived in a Hilbert space in the same way as in  $\mathbb{R}^n$  and also yields the same algorithm, with one small, but important observation. Günnel et al. (2014) pointed out that the Riesz operator  $R \in \mathcal{L}(\mathcal{U}^*, \mathcal{U})$ , defined by

$$(Rr, x)_{\mathcal{U}} = [r, x]_{\mathcal{U}^*, \mathcal{U}}, \quad \forall r \in \mathcal{U}^*, x \in \mathcal{U}, \quad (3.11)$$

takes the same place as the inverse of a preconditioner in a so-called *preconditioned* CG method. Algorithm 3.3.1 shows the CG method in Hilbert space taken from Günnel et al. (2014).

**Algorithm 3.3.1** (CG method for (3.9) in Hilbert space).

```

1: set  $r_0 := b - Ax_0 \in \mathcal{U}^*$ 
2: set  $p_0 := Rr_0 \in \mathcal{U}$ 
3: set  $k := 0$ 
4: while (conv. crit. not satisfied) do
5:   set  $\alpha_k := \frac{[r_k, Rr_k]}{[Ap_k, p_k]}$ 
6:   set  $x_{k+1} := x_k + \alpha_k p_k$ 
7:   set  $r_{k+1} := r_k - \alpha_k Ap_k$ 
8:   set  $\beta_{k+1} := \frac{[r_{k+1}, Rr_{k+1}]}{[r_k, Rr_k]}$ 
9:   set  $p_{k+1} := Rr_{k+1} + \beta_{k+1} p_k$ 
10:  set  $k := k + 1$ 
11: end while

```



The choice of the preconditioner will be explained in section 3.5. The convergence criteria is the standard decrease of the residual in the  $R$ -norm (or  $\mathcal{U}^*$ -norm), e. g.

$$\frac{\|r_k\|_R}{\|b\|_R} \leq \text{rTol} \quad \text{or} \quad \|r_k\|_R \leq \text{aTol}, \quad (3.12)$$

with a user defined relative tolerance  $\text{rTol} > 0$  and absolute tolerance  $\text{aTol} > 0$ . We choose the norm of the right-hand side  $b$  in the denominator instead of the norm of  $r_0$  because we would like to have a stopping criterion which is independent of the initial guess  $x_0$ . A good initial guess would make it harder to satisfy the relative stopping criterion since the denominator would be smaller. For example, if we have an initial guess  $x_0$  which is very close to the solution, the relative stopping criterion might not even be satisfiable due to numerical rounding errors. We will still use the phrase *initial residual norm* for the norm  $\|b\|_R$ , even though the actual initial residual is  $r_0 = b - Ax_0$ .

### Truncated CG Method

The CG method requires the linear operator  $A = W_{\mathcal{U}\mathcal{U}}(\mathbf{U}_k, \mathbf{C})$  to be positive definite (coercive). This cannot be guaranteed for all iterates, however Ball's existence theorem implies that there is a local minimum thus the linear operator  $A$  is at least positive semi-definite. Therefore the assumption that  $A$  is positive definite near a solution is reasonable. In a case like this, a truncated CG method can be used inside of Newton's method, see "Line search Newton-CG method" in (Nocedal and Wright, 2006, ch. 7.1). Weiser et al. (2007) have used this approach to solve problems with elasticity, though they used a different technique for the globalization to ensure convergence of Newton's method.

The basic idea of a truncated CG method is simply to stop the CG method if a search direction  $p_k$  with negative curvature is detected. A negative curvature means that the term  $[Ap_k, p_k]$  is negative, in which case the iterate  $x_k$  is not updated in the direction  $p_k$ . This is a very small modification of the CG method and can be easily implemented, see algorithm 3.3.2, lines 5-7. As we assume  $A$  to be positive definite near the solution, a break due to negative curvatures will not happen near the solution. Therefore we can expect that this modification will not worsen the local convergence behavior of Newton's method as we eventually have a standard CG method.

### MinRes Method

Another Krylov subspace method to solve  $Ax = b$  for a self-adjoint operator  $A$  is the MINRES method. Like in the CG method, Günnel et al. (2014) have shown that the Riesz operator takes the role of the (inverse) preconditioner. Thus the choice of the inner product in the Hilbert space  $\mathcal{U}$  also defines the preconditioner, which is an explanation why a preconditioner for the MINRES method has to be positive



definite. Algorithm 3.3.3 shows details on the MINRES method, taken from Günnel et al. (2014).

**Algorithm 3.3.2** (Truncated CG method for (3.9) in Hilbert space).

```

1: set  $r_0 := b - Ax_0 \in \mathcal{U}^*$ 
2: set  $p_0 := Rr_0 \in \mathcal{U}$ 
3: set  $k := 0$ 
4: while (conv. crit. not satisfied) do
5:   if  $[Ap_k, p_k] < 0$  then
6:     return current  $x_k$  if  $k > 0$ , otherwise  $p_0$ 
7:   end if
8:   set  $\alpha_k := \frac{[r_k, Rr_k]}{[Ap_k, p_k]}$ 
9:   set  $x_{k+1} := x_k + \alpha_k p_k$ 
10:  set  $r_{k+1} := r_k - \alpha_k Ap_k$ 
11:  set  $\beta_{k+1} := \frac{[r_{k+1}, Rr_{k+1}]}{[r_k, Rr_k]}$ 
12:  set  $p_{k+1} := Rr_{k+1} + \beta_{k+1} p_k$ 
13:  set  $k := k + 1$ 
14: end while

```

**Algorithm 3.3.3** (MINRES method for (3.9) in Hilbert space).

```

1: set  $v_0 := 0 \in \mathcal{U}^*$  and  $w_0 := w_1 := 0 \in \mathcal{U}$ 
2: set  $v_1 := b - Ax_0 \in \mathcal{U}^*$ 
3: set  $z_1 := Rv_1$ 
4: set  $\gamma_1 := [v_1, z_1]^{1/2}$ 
5: set  $z_1 := z_1/\gamma_1$  and  $v_1 := v_1/\gamma_1$ 
6: set  $\eta_0 := \gamma_1$ ,  $s_0 := s_1 := 0$ ,  $c_0 := c_1 := 1$ 
7: set  $k := 1$ 
8: while (conv. crit. not satisfied) do
9:   set  $\delta_k := [Az_k, z_k]$ 
10:  set  $v_{k+1} := Az_k - \delta_k v_k - \gamma_k v_{k-1}$ 
11:  set  $z_{k+1} := Rv_{k+1}$ 
12:  set  $\gamma_{k+1} := [v_{k+1}, z_{k+1}]^{1/2}$ 
13:  set  $z_{k+1} := z_{k+1}/\gamma_{k+1}$  and  $v_{k+1} := v_{k+1}/\gamma_{k+1}$ 
14:  set  $\alpha_0 := c_k \delta_k - c_{k-1} s_k \gamma_k$  and  $\alpha_1 := (\alpha_0^2 + \gamma_{k+1}^2)^{1/2}$ 
15:  set  $\alpha_2 := s_k \delta_k + c_{k-1} c_k \gamma_k$  and  $\alpha_3 := s_{k-1} \gamma_k$ 
16:  set  $c_{k+1} := \alpha_0/\alpha_1$  and  $s_{k+1} := \gamma_{k+1}/\alpha_1$ 
17:  set  $w_{k+1} := (1/\alpha_1)[z_k - \alpha_3 w_{k-1} - \alpha_2 w_k]$ 
18:  set  $x_k := x_{k-1} + c_{k+1} \eta_{k-1} w_{k+1}$ 
19:  set  $\eta_k := -s_{k+1} \eta_{k-1}$ 

```



```

20:   set  $k := k + 1$ 
21: end while

```

Similarly to the CG method, we choose the convergence criterion

$$\frac{\|r_k\|_R}{\|b\|_R} = \frac{|\eta_k|}{\|b\|_R} \leq \text{rTol} \quad \text{or} \quad \|r_k\|_R = |\eta_k| \leq \text{aTol}. \quad (3.13)$$

The current residual norm  $\|r_k\|_R$  can be easily computed by  $\|r_k\|_R = |\eta_k|$ .

### Inexact Newton's Method

One reason for using Newton's method is its fast local convergence. To retain this convergence rate, one has to be careful how to incorporate iterative linear solvers like CG or MINRES.

**Remark 3.3.4** (Local convergence of Newton's method). Setting the step length  $\alpha = 1$  in the update (3.1) of Newton's method would yield a standard Newton's method. This method has a local quadratic convergence rate if it converges and certain assumptions are satisfied, see (Deuffhard, 2004, ch. 8) or (Kelley, 1995, Th. 5.1.2). This means that the iterates  $\mathbf{U}_k$  converge towards the solution  $\mathbf{U}_*$  and there exists a constant  $C > 0$  such that

$$\|\mathbf{U}_{k+1} - \mathbf{U}_*\| \leq C \|\mathbf{U}_k - \mathbf{U}_*\|^2 \quad \forall k \in \mathbb{N}.$$

This only holds true if the Newton step (3.1) is solved to sufficient accuracy. As we will use iterative solvers for the Newton system, we have to choose the stopping criteria of the linear solvers carefully so we will not lose the fast local convergence of Newton's method.

A Newton's method, where the Newton system is not solved accurately in each step, is called an *inexact Newton's method*. Here, we solve the first Newton steps with a lower accuracy by setting the relative tolerance of the iterative linear solver quite large, e. g.  $\text{rTol}_{\text{safe}} = 10^{-4}$ . This tolerance will be lowered when Newton's method converges, that means the norm  $\|W_{,\mathbf{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*}$  from the stopping criterion (3.3) of Newton's method decreases. The choice how we set the relative tolerance  $\text{rTol}$  of the linear solver influences the local convergence behavior of Newton's method. The following two strategies depend on the norm of the current right-hand side in the Newton step (3.1), that is  $\|W_{,\mathbf{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*}$ . We can choose the relative tolerances

$$\begin{aligned} \text{rTol} &= \min \{ \text{rTol}_{\text{safe}}, \|W_{,\mathbf{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*} \} \\ \text{or} \quad \text{rTol} &= \min \left\{ \text{rTol}_{\text{safe}}, \|W_{,\mathbf{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*}^{\frac{1}{2}} \right\}, \end{aligned} \quad (3.14)$$

with a minimal relative tolerance  $\text{rTol}_{\text{safe}} > 0$ . The first strategy will keep the  $q$ -quadratic convergence rate, while the second strategy will give a  $q$ -superlinear



convergence rate (of order  $\frac{3}{2}$ ), that means there exists a zero sequence  $c_k$  with

$$\|\mathbf{U}_{k+1} - \mathbf{U}_*\| \leq c_k \|\mathbf{U}_k - \mathbf{U}_*\| \quad \forall k \in \mathbb{N},$$

see (Nocedal and Wright, 2006, Th. 7.2) or (Kelley, 1995, Th. 6.1.2). Next to this relative stopping criterion, we use a stopping criterion with a very small absolute tolerance  $\text{aTol}$ . This can be understood as a safety action: If we do not use  $\text{aTol}$  as well, we risk that rounding errors might prevent the iterative linear solvers from satisfying the relative stopping criterion.

Another point is the expectation on the performance of the algorithm with respect to the computational costs. We emphasize that the local quadratic convergence rate is with respect to the number of iterations, not to the computational costs themselves. The overall algorithm, including the iterative solvers for the (linear) Newton system, might not be as fast if the iterative solvers take a lot of computational time due to their possibly slower convergence rate. Their number of iterations might also grow as we solve the Newton system more and more accurately. Still, Newton's method shows the local quadratic convergence, even though the computational costs to achieve it might grow quite fast.

### 3.4 Discretization

In order to approach the energy minimization problem (2.45) with the solution  $\mathbf{U} \in \mathcal{U}$  numerically, we need to reduce the infinite dimensional function space  $\mathcal{U}$  to a finite dimensional space  $\mathcal{U}_h$  and compute an approximation  $\mathbf{U}_h$  of the solution  $\mathbf{U}$ . This procedure is called *discretization* and one of the most common methods is the finite element method (FEM). We briefly introduce the FEM to clarify the notation. For a comprehensive introduction and details we refer to Braess (2007), Babuška et al. (2011), Brenner and Scott (2002), Elman et al. (2005) and references therein.

We start with the base of the FEM, the triangulation or mesh. Having a two- or three-dimensional domain  $\Omega$ , we decompose it into a finite number of simplices, that are triangles or tetrahedra, in this work.

**Definition 3.4.1** (Triangulation). Let  $\Omega \subset \mathbb{R}^d$ ,  $d = 2, 3$ , be a given open two- or three-dimensional domain. We call a decomposition

$$\mathcal{T}_h = \{T_1, T_2, T_3, \dots, T_{N_T}\},$$

with simplices  $T_i$  a *triangulation* or mesh if the following criteria are satisfied:

- (1)  $\overline{\Omega} = \bigcup_{i=1}^{N_T} T_i$
- (2) two simplices share, that is  $T_i \cap T_j$ ,  $i \neq j$ , either
  - a point that is a corner node of both simplices



- a line that is an edge of both simplices
- a triangle that is a facet of both simplices (only for  $d = 3$ )

The task of decomposing the domain  $\overline{\Omega}$  into a triangulation  $\mathcal{T}_h$  can be achieved by either using a structured mesh in case  $\overline{\Omega}$  is a rather simple geometry that allows a straightforward definition of the simplices, or by a mesh generator for more arbitrary domains.

Having a mesh  $\mathcal{T}_h$ , we can define ansatz functions on the simplices in order to define basis functions to span a finite-dimensional function space  $\mathcal{U}_h$ . Table 3.1 shows the kinds of Lagrange finite elements which we use in this thesis.

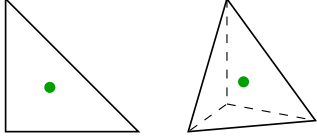
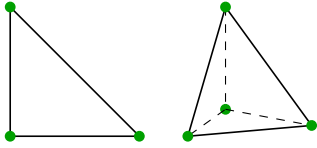
Piecewise <b>constant</b> elements $\mathcal{P}_0$	
$v : \overline{\Omega} \rightarrow \mathbb{R}$ with $v _{\text{int } T_i} \equiv \text{const}, \forall T_i \in \mathcal{T}_h$	
Continuous, piecewise <b>linear</b> elements $\mathcal{P}_1$	
$v \in C^0(\overline{\Omega})$ with $v _{T_i}$ is affine linear w. r. t. $\mathbf{x}$	

TABLE 3.1. The two types of finite elements which are used in this work. The elements defined on a triangle are used for the plate model, while the tetrahedron is used for the full three-dimensional problem. The green dots are the nodes representing the degrees of freedom of a Lagrange finite element.

With these ansatz functions, the Lagrange basis functions  $\varphi_i$ ,  $i = 1, \dots, N_{\mathcal{U}}$ , are defined such that  $\varphi_i(\mathbf{x}_j^{\text{glob}}) = \delta_{ij}$  with the global nodes  $\mathbf{x}_j^{\text{glob}}$  and  $\dim \mathcal{U}_h = N_{\mathcal{U}}$ . These functions form a base of the discretized function space  $\mathcal{U}_h$ , e. g.

$$\mathcal{U}_h = \text{span} \{ \varphi_1, \varphi_2, \dots, \varphi_{N_{\mathcal{U}}} \}.$$

Hence, a function  $\mathbf{U}_h \in \mathcal{U}_h$  can be represented by the coefficient vector

$$\vec{u} = (u_1, u_2, \dots, u_{N_{\mathcal{U}}})^\top \in \mathbb{R}^{N_{\mathcal{U}}},$$

$$\text{with } \mathbf{U}_h(\mathbf{x}) = \sum_{i=1}^{N_{\mathcal{U}}} u_i \varphi_i(\mathbf{x}), \quad \mathbf{x} \in \Omega.$$



We use this discretization to solve our energy minimization numerically. As said earlier, we first *optimize* (that is applying algorithm 3.1.1) and then *discretize* (that is reducing the Newton system from  $\mathcal{U}$  to  $\mathcal{U}_h$ ).

**Definition 3.4.2** (Discretized State Space). Let  $\mathcal{T}_h$  be the triangulation and  $\mathcal{U}$  the state space of the domain  $\Omega$ . We choose continuous piecewise linear elements  $\mathcal{P}_1$  to discretize the state space, that is

$$\mathcal{U}_h = \left\{ U \in \mathcal{U} : U|_{T_i} \text{ is affine linear} \right\}. \quad (3.15)$$

**Definition 3.4.3** (Discretized Control Space). Let  $\mathcal{T}_h$  be the triangulation and  $\mathcal{C}$  the control space of the domain  $\Omega$ . We choose continuous piecewise constant elements  $\mathcal{P}_0$  to discretize the control space, that is

$$\mathcal{C}_h = \left\{ C \in \mathcal{C} : C|_{T_i} \text{ is constant} \right\}. \quad (3.16)$$

If the load/control lives on the boundary, e. g.  $\mathcal{C} = L_2(\Gamma^N)$ , the discretized control space lives on  $\Gamma^N$  as well.

Having defined the discretized function spaces  $\mathcal{U}_h$  and  $\mathcal{C}_h$ , we can have another look at the algorithms 3.1.1 (Newton's method), 3.3.1 (CG method) and 3.3.3 (MINRES method). Besides discretizing all functions in these algorithms, we need representations of the operator  $A$  and the Riesz operator  $R$ . These will be the stiffness matrix  $A_h$  and the discretized Riesz  $R_h$  which will be the application of the preconditioner  $P$ . First, we start with the stiffness matrix while section 3.5 will present the preconditioner.

We consider the Newton system (3.2), that is

$$W_{,UU}(U_h, C_h)[\delta U_h, \Delta U_h] = -W_{,U}(U_h, C_h)[\delta U_h], \quad \forall \delta U_h \in \mathcal{U}_h,$$

and replace the function space  $\mathcal{U}$  by  $\mathcal{U}_h$ , e. g.

$$W_{,UU}(U_h, C_h)[\delta U_h, \Delta U_h] = -W_{,U}(U_h, C_h)[\delta U_h], \quad \forall \delta U_h \in \mathcal{U}_h, \quad (3.17)$$

for a given load  $C_h \in \mathcal{C}_h$ , which can be obtained by an approximation  $C_h \approx C \in \mathcal{C}$ . Using the coefficient vectors

$$\begin{aligned} \vec{u} &= (u_1, u_2, \dots, u_{N_{\mathcal{U}}})^T \in \mathbb{R}^{N_{\mathcal{U}}}, & \Delta U_h(\mathbf{x}) &= \sum_{j=1}^{N_{\mathcal{U}}} u_j \varphi_j(\mathbf{x}), & \mathbf{x} \in \Omega, \\ \vec{v} &= (v_1, v_2, \dots, v_{N_{\mathcal{U}}})^T \in \mathbb{R}^{N_{\mathcal{U}}}, & \delta U_h(\mathbf{x}) &= \sum_{i=1}^{N_{\mathcal{U}}} v_i \varphi_i(\mathbf{x}), & \mathbf{x} \in \Omega, \end{aligned} \quad (3.18)$$



and inserting this into (3.17) yields

$$\begin{aligned} W_{,UU}(\mathbf{U}_h, \mathbf{C}_h)[\Sigma v_i \varphi_j, \Sigma u_j \varphi_j] &= -W_{,U}(\mathbf{U}_h, \mathbf{C}_h)[\Sigma v_i \varphi_i], \quad \forall \vec{v} \in \mathbb{R}^{N_U} \\ \Leftrightarrow \sum_{i,j=1}^{N_U} u_j v_i W_{,UU}(\mathbf{U}_h, \mathbf{C}_h)[\varphi_i, \varphi_j] &= -\sum_{j=1}^{N_U} v_i W_{,U}(\mathbf{U}_h, \mathbf{C}_h)[\varphi_i], \quad \forall \vec{v} \in \mathbb{R}^{N_U}. \end{aligned}$$

By choosing the unit vectors  $\vec{e}_i$  in  $\mathbb{R}^{N_U}$  for the vectors  $\vec{v}$ , we finally obtain the system of linear equations

$$A_h \vec{u} = \vec{b}, \quad (3.19)$$

with the symmetric stiffness matrix

$$[A_h]_{ij} = W_{,UU}(\mathbf{U}_h, \mathbf{C}_h)[\varphi_i, \varphi_j], \quad (3.20)$$

and the right-hand side

$$[\vec{b}]_i = W_{,U}(\mathbf{U}_h, \mathbf{C}_h)[\varphi_i]. \quad (3.21)$$

We like to emphasize that both, stiffness matrix  $A_h \in \mathbb{R}^{N_U \times N_U}$  and right-hand-side  $b_h \in \mathbb{R}^{N_U}$  depend on the current iterate  $\mathbf{U}_k$  in Newton's method. The assembly of  $A_h$  and  $\vec{b}$  is normally done element-wise and the integrals are computed by Gauss quadrature rules for triangles and tetrahedra.

The stiffness matrix  $A_h$  is symmetric, what follows from the symmetry of second derivatives. The symmetry is very important property for the choice of linear solvers and could be destroyed if the Dirichlet boundary conditions are applied to the linear system (3.19) in the wrong way.

**Remark 3.4.4** (Ways to apply Dirichlet boundary conditions).

**Elimination:** The nodes belonging the Dirichlet BCs are eliminated, that is their already known values are integrated into the linear system and their entries are then deleted from the matrix  $A_h$ , coefficient vector  $\vec{u}$  and right-hand side  $f_h$ . Hence, the dimension of the problem is reduced since the degrees of freedom from the Dirichlet nodes are eliminated. In the implementation, this approach requires a rearrangement of memory allocated for the matrix and vectors, which might be costly.

**Nonsymmetric row replacement:** Let the degree of freedom  $j$  be part of the Dirichlet BC  $u_j = 0$ , e.g. the  $\mathbf{x}_2$ -component in a certain global node. The  $j$ -th row from the matrix  $A_h$  and right-hand side  $f_h$  are replaced by unit vectors  $\vec{e}_j^T$  and  $[f_h]_j = 0$ , such that the equation  $u_j = 0$  is used instead of the PDE in this node. (In the case of inhomogeneous Dirichlet BC  $u_j = u_j^D$  and that the current iterate does not fulfill them yet, the difference between given value and current value in this node is written in the right-hand side). This can be easily implemented but the symmetry of the matrix is destroyed.



**Symmetric row and column replacement:** Again, let the degree of freedom  $j$  be part of the Dirichlet BCs. The  $j$ -th row from the matrix  $A_h$  and right-hand side  $f_h$  are replaced by unit vectors  $\vec{e}_i^\top$  and  $[f_h]_j = 0$ , such that the equation  $u_j = 0$  is used instead of the PDE in this node. In order to preserve the symmetry of  $A_h$ , the  $j$ -th column is replaced by  $\vec{e}_j$ . In case of homogeneous Dirichlet BCs, this does not require a change of the right-hand side  $f_h$ . (In case that the inhomogeneous Dirichlet BCs are not fulfilled yet, certain terms have to be added to  $f_h$  too, hence applying the Dirichlet BCs has to be done on  $A_h$  and  $f_h$  simultaneously.)

**Projection:** The projection method can be used with iterative solvers like Krylov subspace methods. The matrix  $A_h$  and the right-hand side  $f_h$  are not modified, but after multiplying with the matrix  $A_h$ , e.g.  $\vec{r} = A_h \vec{u} - f_h$ , the Dirichlet BCs are applied to the result, e.g.  $\vec{r}$ . This ensures that the iterates of the Krylov method stay in the subspace where the homogeneous Dirichlet BCs are fulfilled.

As we will use a Krylov subspace method, we choose to keep the symmetry of  $A_h$  by a symmetric row and column replacement. The projection method would also be possible, but it might cause difficulties with the smoother inside the preconditioner, see the next section 3.5.

### 3.5 Preconditioner and Multigrid Method

The choice of the preconditioner  $P$  has a huge impact on the convergence behavior of the Krylov subspace method for the linear system  $Ax = b$ , with a linear operator  $A \in \mathcal{L}(\mathcal{U}, \mathcal{U}^*)$  and a Hilbert space  $\mathcal{U}$ . As mentioned earlier, this choice is equivalent to the choice of the inner product  $(\cdot, \cdot)_{\mathcal{U}}$  of the Hilbert space  $\mathcal{U}$ . This also means that for an unfavorable choice for preconditioner/inner product, even properties like boundedness or coercivity of the linear operator  $A$  might be lost.

Many practical realizations of a preconditioner cannot be formulated for the infinite dimensional function space  $\mathcal{U}$ , but rather depend on the discretized space  $\mathcal{U}_h$ . For example, methods like incomplete-LU-factorization (ILU) or algebraic multigrid (AMG) depend on the structure of the matrix  $A_h$ , while other methods like Bramble-Pasciak-Xu (BPX) or geometrical multigrid also need information about the hierarchy of a family of meshes. Therefore we understand the preconditioner as a symmetric and positive definite matrix  $P$  which is applied on the discretized linear system  $A_h \vec{x} = \vec{b}$ .

We seek to find a “good” preconditioner, which is a compromise between the performance of the CG or MINRES method, e.g. how many iterations are needed to solve the problem to a satisfactory accuracy, and the computational costs of the application of the preconditioner. There are two extreme cases: The identity matrix  $P = I$  has virtually no costs for the application but a very poor performance for the



CG or MINRES method for fine meshes. On the other hand, choosing the matrix  $P = A_h$  implies that we need only one Krylov iteration, but the costs to apply the preconditioner is a linear system solve itself. Various methods for the preconditioner  $P$  has been proposed so far in order to find a compromise between costs and performance of the Krylov subspace method. It is important to note that there is no “best” preconditioner and that the choice should always take the problem behind it into account.

The performance, or convergence behavior, of the CG and MINRES method is much more complex than the classical convergence proof suggests. For example, let us restate the result for the CG method from (Braess, 2007, Th. 4.4).

**Theorem 3.5.1** (Classical linear bound for the CG method). Let the sequence  $\vec{x}_0, \vec{x}_1, \vec{x}_2, \dots$  be iterates generated by the CG method (alg. 3.3.1) with the preconditioner  $P$  applied on the linear system  $A_h \vec{x} = \vec{b}$  with the solution  $\vec{x}_*$  and let  $\kappa$  be the condition number of the symmetric positive definite matrix  $PA_h$ . Then the iterates satisfy the linear bound

$$\|\vec{x}_k - \vec{x}_*\|_{A_h} \leq 2 \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \|\vec{x}_0 - \vec{x}_*\|_{A_h} \right)^k \quad (3.22)$$

with the  $A_h$ -norm  $\|\vec{x}\|_{A_h} := \vec{x}^\top A_h \vec{x}$ .

**Remark 3.5.2.** This result can be extended to infinite dimensional Hilbert spaces where  $\kappa$  is the quotient of the lower and upper bound of the spectrum of the linear operator  $RA \in \mathcal{L}(\mathcal{U}, \mathcal{U})$ , see (Daniel, 1967, Section 1.2) and (Elman et al., 2005, Section 2.2), where a similar linear bound for the MINRES method is shown.

According to this linear bound based on the condition number  $\kappa$ , we can expect a better convergence rate if the condition number is small. However, as Gergelits and Strakoš (2013) recently noted, this a-priori convergence rate analysis is based on simplifications that eventually cannot be expected to describe the actual convergence behavior of a CG method running on a computer. For example, the classical convergence rate analysis neglects rounding errors and simplifies the spectrum of  $A$  to its bounds or outlying eigenvalues. We conclude that a small condition number  $\kappa$  is desirable for our purpose and might even offer a better convergence in the CG or MINRES method, though we do not expect the convergence behavior to behave as the linear bound based on the condition number  $\kappa$  might suggest.

Another desirable property is the spectral equivalence of the preconditioner  $P$  and the stiffness matrix  $A_h$  for different refinement levels. It implies that with subsequently refined meshes, the number of Krylov iterations will be bounded from above, which is also called mesh independent behavior. Otherwise, the running time of the



overall algorithm to solve the problem would not scale linearly with the problem size, that is the number of degrees of freedom.

**Definition 3.5.3** (Spectrally Equivalent Preconditioner). Let  $A \in \mathcal{L}(\mathcal{U}, \mathcal{U}^*)$  be a linear operator and let  $\mathcal{U}_0 \subset \mathcal{U}_1 \subset \mathcal{U}_2 \subset \dots \subset \mathcal{U}$  be a family of discretized function spaces. The discretizations of  $A$  are  $A_i \in \mathcal{L}(\mathcal{U}_i, \mathcal{U}_i^*)$ . The family of preconditioners  $P_i \in \mathcal{L}(\mathcal{U}_i, \mathcal{U}_i^*)$  is called *spectrally equivalent* if there exist constants  $\underline{c}, \bar{c} > 0$  such that for all  $i = 0, 1, 2, \dots$ , we have

$$\underline{c} [P_i x, x]_{\mathcal{U}_i^*, \mathcal{U}_i} \leq [A_i x, x]_{\mathcal{U}_i^*, \mathcal{U}_i} \leq \bar{c} [P_i x, x]_{\mathcal{U}_i^*, \mathcal{U}_i}, \quad \forall x \in \mathcal{U}_i. \quad (3.23)$$

We emphasize that the spectral equivalence of  $P$  and  $A_h$  does not guarantee a fast convergence as the quotient  $\kappa := \bar{c}/\underline{c}$  takes the role of the condition number in the linear bound and  $\kappa$  might be very large. For example, the stiffness matrices belonging to the standard inner product of the Sobolev space  $H_0^1(\Omega)^3$  might be a bad choice even though they are spectrally equivalent to the matrices  $A_i$ . A better choice would be the inner product (2.28) which belongs to the linear model of elasticity. Of course, we will not solve this problem exactly, but rather approximately by a V-cycle of a geometrical multigrid method. Another choice would be a V-cycle of the stiffness matrix  $A_h$  itself, though it is not guaranteed that this preconditioner is positive definite, see section 5.4. Let us start by briefly introducing the multigrid method itself.

### Multigrid Method

The core idea is a hierarchy of meshes, so-called *levels*. While solving a problem on the finest mesh, we can switch to other meshes where it might be easier to solve the problem and low frequent errors can be easily diminished. See Hackbusch (1980), Wesseling (1992) or Trottenberg et al. (2001) for an introduction.

We start with a given mesh  $\mathcal{T}_0$ , the *coarse* mesh, and subsequent refinements yield the new meshes  $\mathcal{T}_1 \subset \mathcal{T}_2 \subset \dots \subset \mathcal{T}_L$ , where  $L$  is the number of total refinements. We assume that the refinement only adds new degrees of freedom, therefore the discretized function spaces  $\mathcal{U}_0 \subset \mathcal{U}_1 \subset \dots \subset \mathcal{U}_L$  are nested. We now define a prolongation which transfers a function  $u_h \in \mathcal{U}_\ell$  on level  $\ell$  to the next finer mesh on level  $\ell + 1$ , or to be more precise, the coefficient vector is prolonged. Since the discretized function spaces are nested, the prolonged function is the same function. Of course, this identity in the function space does not mean an identity for the coefficient vectors since the basis functions in  $\mathcal{U}_\ell$  and  $\mathcal{U}_{\ell+1}$  are different. This is where the prolongation operator  $p_{\ell \rightarrow \ell+1}$  takes its role.

Let  $\{\varphi_j\}_{j=1}^L$  and  $\{\psi_i\}_{i=1}^{N_{\ell+1}}$  be the finite element basis of  $\mathcal{U}_\ell$  respectively  $\mathcal{U}_{\ell+1}$ . A function  $u \in \mathcal{U}_\ell$  can be represented by a coefficient vector on each level, e. g.  $\vec{x} \in \mathbb{R}^L$



on level  $\ell$  and  $\vec{y} \in \mathbb{R}^{N_{\ell+1}}$  on level  $\ell + 1$ , with the identity

$$x_\ell = \sum_{j=1}^{N_\ell} x_j^\ell \varphi_j = \sum_{i=1}^{N_{\ell+1}} y_i \psi_i.$$

We evaluate this equation in the global nodes  $\mathbf{x}_k^{\text{glob}}$ ,  $k = 1, \dots, N_{\ell+1}$  of the mesh on level  $\ell + 1$  yields and, as we use Lagrange elements, that implies  $\psi_i(\mathbf{x}_k^{\text{glob}}) = \delta_{ik}$ , we get a simple representation for the vector  $\vec{y}$ , that is

$$y_k = \sum_{j=1}^{N_\ell} x_j \varphi_j(\mathbf{x}_k^{\text{glob}}).$$

This sum is actually very short because the Lagrange finite elements have a local support.

**Definition 3.5.4** (Prolongation and restriction matrix). The *prolongation matrix*  $p_{\ell \rightarrow \ell+1} \in \mathbb{R}^{N_{\ell+1} \times L}$  (such that  $\vec{y} = p_{\ell \rightarrow \ell+1} \vec{x}$ ) is defined as

$$[p_{\ell \rightarrow \ell+1}]_{ij} = \varphi_j(\mathbf{x}_i^{\text{glob}}). \quad (3.24)$$

The canonical *restriction matrix*  $r$  is defined as  $r_{\ell+1 \rightarrow \ell} := p_{\ell \rightarrow \ell+1}^\top$ .

With these prolongation and restriction operators, we can move from one level to another. This allows the following idea of multigrid: A decomposition of the error of an intermediate solution into the eigenfunctions of the operator  $A$  yields parts of higher and parts of lower “frequencies”<sup>2</sup>. The intermediate solution on the finest mesh has higher and lower frequency error components, while those with higher frequency tend to be “smoothed out” faster than those with lower frequencies. On the other hand, the errors with high frequencies are not resolved on a very coarse mesh and here the former low frequency errors have a high frequency, and hence can be “smoothed out” on the coarse mesh. The conclusion is the multigrid idea: Smooth out the higher frequency error on the finer meshes and the lower frequency errors on the coarser meshes. However, we will not employ a full multigrid method to solve our linear problems but rather use it as a preconditioner inside of a Krylov subspace method, see Braess (2007) or the numerical experiments in Ospald (2012).

### V-Cycle of a Multigrid Method

A multigrid V-cycle starts on the finest mesh level  $L$  and computes the current residual on this level. The residual is then restricted and passed to the next coarser mesh, where a smoothing is applied. This is repeated until the coarsest mesh level  $\ell = 0$  is reached. Here, the problem is solved by a direct solver. This is rather

<sup>2</sup>The word “frequency” is very descriptive in the case of the Poisson equation in 1D where the eigenfunctions are sin- and cos-functions whose frequencies varies the eigenvalues. In the case of linear elasticity, it even has the physical meaning of natural oscillation of an elastic body.



inexpensive because the number of degrees of freedom on the level  $\ell = 0$  is small compared to the other levels. The correction is prolonged and passed to the next finer mesh level  $\ell = 1$ , where a smoothing is applied. This procedure is repeated until the finest mesh is reached. Braess and Hackbusch (1983) proved that a multigrid with a V-cycle converges mesh independently. Therefore, our preconditioner, a single multigrid V-cycle with a matrix  $K_{\text{lin,elast}}$ , is spectral equivalent to the inverse  $K_{\text{lin,elast}}^{-1}$ , which would be an exact application of the preconditioner. Furthermore, Schöberl et al. (2011) have shown that this also applies for the linear Reissner-Mindlin plate model, which is a special case of the hierarchical plate model for small deformations. Hence we choose the multigrid V-cycle also in the case of the hierarchical plate model to test it.

A recursive definition of a multigrid V-cycle can be found in algorithm 3.5.5. The matrices  $A_\ell$  on the various levels can be obtained by a Galerkin projection, see (Hackbusch, 1980, ch. 3.7). We only assemble the matrix  $A_L$  on the finest mesh and restrict it to the other levels by recursively computing

$$A_{\ell-1} = r_{\ell+1 \rightarrow \ell} A_\ell p_{\ell \rightarrow \ell+1}. \quad (3.25)$$

**Algorithm 3.5.5** (Multigrid V-cycle  $MGV(\ell, \vec{x}, \vec{b})$  for the linear system  $Ax = b$ ).

**Input:** level  $\ell$ , iterate  $\vec{x} \in \mathbb{R}^L$ , right-hand side  $\vec{b} \in \mathbb{R}^L$

**Output:** new iterate  $\vec{x} \in \mathbb{R}^L$

```

1: if  $\ell = 0$  then
2:   return  $\vec{x} = A_0^{-1} \vec{b}$ 
3: else
4:    $\vec{x} := \mathcal{S}_\ell^{\nu_1}(A_\ell, \vec{x}, \vec{b})$ 
5:    $\vec{d} := r_{\ell+1 \rightarrow \ell}(\vec{b} - A_\ell \vec{x})$ 
6:    $\vec{e} := MGV(\ell - 1, \vec{e}, \vec{d})$ 
7:    $\vec{x} := \vec{x} + p_{\ell \rightarrow \ell+1} \vec{e}$ 
8:    $\vec{x} := \mathcal{S}_\ell^{\nu_2}(A_\ell, \vec{x}, \vec{b})$ 
9:   return  $\vec{x}$ 
10: end if
```

The choice of the smoother  $\mathcal{S}$  and its number of repetitions  $\nu_1, \nu_2$  have an influence on the preconditioner in several aspects: As we want to use a V-cycle as a preconditioner, the application of a V-cycle has to be that of a symmetric, positive definite, linear operator. On the other hand, the choice of the smoother has an impact on how “good” the preconditioner is, this means the performance of the Krylov method as well as the computational costs of applying the smoother. Numerical experiments in Ospald (2012) on linear elasticity suggest the use of a *successive over-relaxation* (SOR) and  $\nu_1, \nu_2 = 1$ .

### Nested Iterations



Since we have a hierarchy of nested meshes from the multigrid method, we can employ the idea of so-called *nested iterations* to solve the forward problem. Let  $L + 1$  be the number of nested function spaces, e. g.

$$\mathcal{U}_0 \subset \mathcal{U}_1 \subset \dots \subset \mathcal{U}_L.$$

We seek to solve the optimal control problem (4.2) on the finest mesh, that means in  $\mathcal{U}_L$ . Starting on the coarsest mesh and solving the forward problem in  $\mathcal{U}_0$ , we get the solution

$$\mathbf{U}^{(0)} \in \mathcal{U}_0.$$

By interpolating  $\mathbf{U}^{(0)} \in \mathcal{U}_0$  onto  $\mathcal{U}_1$  by the prolongation operator  $p_{0 \rightarrow 1}$  (see definition 3.5.4), we can use this interpolation as an initial guess for the forward problem in  $\mathcal{U}_1$ , e. g.

$$\mathbf{U}_0^{(0)} = p_{0 \rightarrow 1} \mathbf{U}^{(1)} \in \mathcal{U}_1.$$

This might a better initial guess than the generic choice  $\mathbf{U}_0^{(1)} = 0$  and thus fewer iterations of Newton's method on the level 1 are needed. We recursively employ this idea until we have solved the finest mesh, see algorithm 3.5.6. Working on the finest mesh is expensive and by having a better initial guess  $\mathbf{U}_0^{(L)}$ , we hope to save iterations and thus computational work load.

**Algorithm 3.5.6** (Newton's method with nested iterations).

**Input:** initial  $\mathbf{U}_0^{(0)}$ , nested spaces  $\mathcal{U}_0 \subset \mathcal{U}_1 \subset \dots \subset \mathcal{U}_L$

**Output:**  $\mathbf{U}^{(L)}$

```

1: for  $\ell := 0, 1, \dots, L$  do
2:   compute  $\mathbf{U}^\ell$  by solving the forward prob. (2.46) with alg. 3.1.1 starting with
      $\mathbf{U}_0^{(\ell)}$ 
3:   if  $\ell < L$  then
4:     set  $\mathbf{U}_0^{(\ell+1)} := p_{\ell \rightarrow \ell+1} \mathbf{U}^{(\ell)} \in \mathcal{U}_{(\ell+1)}$ 
5:   else
6:     return displacement  $\mathbf{U}^{(L)}$  on the finest mesh level
7:   end if
8: end for
```

### Summary of the Forward Solver

We conclude this chapter by summarizing the core points of the forward problem, that is the computation of the displacement  $\mathbf{U}$  for a given load  $\mathbf{C}$ .

- Newton's method is used to solve the system  $0 = W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})$ , see alg. 3.1.1
- A line search method is used to avoid inadmissible deformations with local negative volume change, see section 3.2
- Krylov methods like truncated CG or MINRES are employed to solve the Newton system (3.2).



- The finite element method (FEM) is used to discretize the Hilbert space  $\mathcal{U}$ , with  $\mathcal{P}_0$ ,  $\mathcal{P}_1$  and  $\mathcal{P}_2$  as finite elements.
- A multigrid hierarchy is built and a V-cycle of a multigrid method serves as a preconditioner for the truncated CG or MINRES method.



# 4 Optimal Control Problems in Elasticity

## Contents

---

4.1	Setting of the Optimal Control Problem	73
4.1.1.	Quality functionals	74
4.1.2.	Cost or Penalty Functionals	84
4.2	Lagrange-Newton: An All-at-once Approach	85
4.2.1.	Solving the Lagrange Equation	86
4.2.2.	Discretization	88
4.3	Quasi-Newton: A Reduced Formulation	89
4.3.1.	Quasi-Newton Method	93
4.3.2.	Broyden-Fletcher-Goldfarb-Shanno-Update	94
4.3.3.	Simple Wolfe-Powell Line Search	97
4.3.4.	Discretization	100

---

The main aspect of this chapter is the optimal control of the elasticity model from chapter 2. That means we want to achieve a desirable state  $\mathbf{U}$  by applying a certain load  $\mathbf{C}$ . We start with the general setting of the optimal control problem in section 4.1. We discuss the objective functional  $I$  and possible ways to quantify a “desirable” state. In some cases for large deformations, the standard approach with a tracking type functional might not work because we simply cannot provide a reasonable desired state  $\mathbf{U}^{\text{des}}$ . We give a couple of examples for alternative objectives. The first is based on a regional penalization of the space occupied by the deformed body. For example, this can be used so that the deformed body avoids certain regions of the space. The second example is motivated by a flower moving towards the sun, where the deformed body in our model tries to point into a given direction. A third functional measures the volume that is gained by the deformation of a plate.

In order to solve the optimal control problem, we present two methods. The first method in section 4.2 is an all-at-once approach that is based on the Lagrange functional and finding a root of its first derivative. Setting the first derivative to zero is a nonlinear equation which we solve with Newton’s method, called the Lagrange-Newton method in optimal control. The Lagrange-Newton system will be solved by a MINRES method with a block preconditioner.

The second method is the quasi-Newton method in section 4.3. Here we eliminate the state  $\mathbf{U} = S(\mathbf{C})$  by the solution operator  $S$  which returns the displacement  $\mathbf{U}$



for a given load  $\mathbf{C}$ . We then have an unconstrained minimization problem with the reduced objective functional  $I^{\text{red}}$ . We continue by showing how the first derivative of the reduced objective functional  $I^{\text{red}}$  can be computed, which includes solving the adjoint equation. Next, we show how a Broyden-Fletcher-Goldfarb-Shanno (BFGS) update formula can be formulated in function space so that we can approximate the second derivative  $I_{\mathbf{C}\mathbf{C}}^{\text{red}}$ . To ensure that this methods works, we use a Wolfe-Powell line search. We conclude section 4.3 by giving some remarks on the discretization for the quasi-Newton method.

## 4.1 Setting of the Optimal Control Problem

The object of interest is an elastic body  $\Omega$  that can be deformed by the load  $\mathbf{C}$ , which can be an external load, for example a volume load  $\mathbf{f}$  or a boundary load  $\mathbf{g}$ , or an internal load, for example an inner pressure  $t$  or a fiber tension  $m$ . Using this load, we seek to deform the body in a desirable way. In the following, we call the load  $\mathbf{C}$  the *control* and the displacement  $\mathbf{U}$  the *state*<sup>1</sup>.

Both, state and control, are linked by the elastic material behavior discussed in chapter 2, that eventually led to the variational equality

$$0 = W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\delta\mathbf{U}], \quad \forall \delta\mathbf{U} \in \mathcal{U} \quad (4.1)$$

with  $W(\mathbf{U}, \mathbf{C}) = W^{(T)}(\mathbf{U}) + W^{(\mathbf{C})}(\mathbf{U}, \mathbf{C})$ ,

where  $W^{\mathbf{C}}$  is one of the energy functionals of the load  $\mathbf{C}$ , compare table 2.1. The control belongs to a control space  $\mathcal{C}$ , depending on which type of load is applied. We assume that we have only one of the four loads as a control for an easier notation, though it is straight-forward how to extend the problem to multiple controls.

Our problem now is to find a certain control  $\mathbf{C}$  to achieve a state  $\mathbf{U}$  which is desirable. We formulate this as the optimization problem

$$\begin{aligned} & \min_{\mathbf{U} \in \mathcal{U}, \mathbf{C} \in \mathcal{C}} I(\mathbf{U}, \mathbf{C}) \\ \text{s.t. } & 0 = W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\delta\mathbf{U}], \quad \forall \delta\mathbf{U} \in \mathcal{U}. \end{aligned} \quad (4.2)$$

In order to illustrate examples for the objective functional  $I : \mathcal{U} \times \mathcal{C} \rightarrow [0, \infty)$ , we assume the splitting

$$I(\mathbf{U}, \mathbf{C}) = Q(\mathbf{U}) + P(\mathbf{C}), \quad (4.3)$$

consisting of a *quality* functional  $Q : \mathcal{U} \rightarrow [0, \infty)$  and a *penalty/cost* functional  $P : \mathcal{C} \rightarrow [0, \infty)$ . Lubkoll et al. (2012) have shown the existence of a solution of the optimal control problem for the case of boundary loads  $\mathbf{g}$  under certain assumptions.

---

<sup>1</sup>Please note that a wide part of mathematical literature on optimal control uses a different notation, where the control is denoted by  $y$  and the state by  $u$ .



### 4.1.1. Quality functionals

We consider a function  $Q : \mathcal{U} \rightarrow [0, \infty)$  that assigns a certain objective value  $Q(\mathbf{U})$  to a given displacement  $\mathbf{U}$ , for example depending on how "well" the deformed body  $\hat{\Omega}$  matches a given desired shape  $\hat{\Gamma}^{\text{des}}$ , which also defines  $\hat{\Omega}^{\text{des}}$  and vice versa. Of course, there are various ways to describe this quality and the practical problem behind it should be taken into account as well. Most importantly, the way how the desired shape is described plays a huge role for the implementation. Among others, we could have

- a parametrization of the surface  $\hat{\Gamma}^{\text{des}}$ ,
- an implicit definition of the surface  $\hat{\Gamma}^{\text{des}}$ ,
- B-splines or non-uniform rational B-splines (NURBS) describing  $\hat{\Gamma}^{\text{des}}$
- a two-dimensional mesh defined by given vertices.

The following presented quality functionals might be more or less applicable to one of those descriptions. First, we give an introducing example and later show how it can be modeled with different quality functionals.

**Example 4.1.1** (Elevation of a Bar). The undeformed body is the bar  $\Omega = [0, 2] \times [0, \frac{1}{5}] \times [0, \frac{1}{10}]$ . The goal is to elevate the right quarter  $\Omega^{\text{right}} = [\frac{3}{2}, 2] \times [0, \frac{1}{5}] \times [0, \frac{1}{10}]$  to the height  $h = 0.2$ , which affects the third spatial coordinate  $x_3$ . The other coordinates  $x_1$  and  $x_2$  are free to change. A load may be applied only in the part  $\Omega^{\text{left}} = [0, \frac{3}{2}] \times [0, \frac{1}{5}] \times [0, \frac{1}{10}]$ .



FIGURE 4.1. The bar in blue and red is the undeformed body  $\Omega$  and the bar in gray is the unknown deformed bar  $\hat{\Omega}$ . The red box illustrates the desired height  $h = 0.2$ , where the red right part  $\Omega^{\text{right}}$  has to be elevated to. The blue region indicates the part  $\Omega^{\text{left}}$  where the load may be applied.

### Desired State

We start with a very common functional which is often found in literature on optimal control.



**Definition 4.1.2** (Standard tracking type). Let a desired state  $\mathbf{U}^{\text{des}} \in \mathcal{U}$  be given, that is we want the state  $\mathbf{U}$  to match this desired state as much as possible. We penalize differences between the two states with the weighted squared error

$$Q^{\text{track}}(\mathbf{U}) := \frac{1}{2} \int_{\Omega} \gamma(\mathbf{x}) \|\mathbf{U}(\mathbf{x}) - \mathbf{U}^{\text{des}}(\mathbf{x})\|_2^2 d\mathbf{x}, \quad (4.4)$$

with a weight function  $\gamma : \Omega \rightarrow [0, \infty)$  and the Euclidean norm  $\|\cdot\|_2$ . The derivative with respect to the state  $\mathbf{U}$  is

$$Q_{,\mathbf{U}}^{\text{track}}(\mathbf{U})[\delta\mathbf{U}] := \int_{\Omega} \gamma(\mathbf{x}) (\mathbf{U}(\mathbf{x}) - \mathbf{U}^{\text{des}}(\mathbf{x}))^\top \delta\mathbf{U}(\mathbf{x}) d\mathbf{x}. \quad (4.5)$$

The model can be easily extended by allowing a weight function  $\gamma_i$  for each spatial coordinate  $\mathbf{x}_i$ .

We give the modeling of the introduction example 4.1.1 with a tracking-type quality functional.

**Example 4.1.3** (Bar: Standard tracking type). This simple example allows us to give a desired state for the right quarter, e. g.  $\mathbf{U}^{\text{des}}(\mathbf{x}) = (0, 0, h)^\top$  and  $\gamma(\mathbf{x}) = \gamma > 0$  for  $\mathbf{x} \in \Omega^{\text{right}}$ . It is not necessary to define  $\mathbf{U}^{\text{des}}$  on the remaining left quarter, as we set  $\gamma(\mathbf{x}) = 0$  for the remaining part anyway. We then have

$$Q^{\text{track}}(\mathbf{U}) := \frac{\gamma}{2} \int_{\Omega^{\text{right}}} \|\mathbf{U}(\mathbf{x}) - (0, 0, h)^\top\|_2^2 d\mathbf{x},$$

and its derivative

$$Q_{,\mathbf{U}}^{\text{track}}(\mathbf{U})[\delta\mathbf{U}] := \gamma \int_{\Omega^{\text{right}}} (\mathbf{U}(\mathbf{x}) - (0, 0, h)^\top)^\top \delta\mathbf{U}(\mathbf{x}) d\mathbf{x}.$$

This quality functional is very practical in various optimal control problems. However, in elasticity with large deformations, it has a drawback because the desired state  $\mathbf{U}^{\text{des}}$  is not easy to determine in some problems. For example, we want to deform the mechanical body in such a way that it matches a certain shape. It is not clear how to extend the desired shape to a unique  $\mathbf{U}^{\text{des}}$ . Even material points on the boundary  $\Gamma$  are not uniquely determined along the desired shape, see figure 4.2. A remedy to this would be to consider the desired state only on a subdomain.

### Desired shape

Let a desired shape  $\widehat{\Gamma}^{\text{des}}$  be given, that means we want to deform the body  $\Omega$  in such a way that its boundary is as “close” as possible to the desired shape  $\widehat{\Gamma}^{\text{des}}$ . The



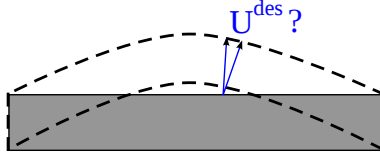


FIGURE 4.2. In this example the shape is given but we cannot uniquely define the displacement for each point. This is true not only for the boundary points, but also for the interior points.

essential question now is how to qualify what is meant that  $\Gamma$  and  $\hat{\Gamma}^{\text{des}}$  are “close”. A possible choice could be the distance between both surfaces, e. g.

$$Q^{\text{shape}_1}(\mathbf{U}) := \int_{\Gamma} \gamma(\mathbf{x}) d(\mathbf{x} + \mathbf{U}(\mathbf{x}), \hat{\Gamma}^{\text{des}}) \, dS,$$

with a weight function  $\gamma : \Gamma \rightarrow [0, \infty)$  and a distance measure  $d$  between a point  $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{U}$  and the desired shape  $\hat{\Gamma}^{\text{des}}$ . This might be easily computable for certain simple shapes  $\hat{\Gamma}^{\text{des}}$ , but it is an enormous effort to determine the distance  $d(\mathbf{x} + \mathbf{U}(\mathbf{x}), \hat{\Gamma}^{\text{des}})$  or the set  $\hat{\Omega} \setminus \hat{\Omega}^{\text{des}}$  for most shapes, let alone computing a derivative which is needed in the optimization later.

Another choice could be the volume difference between the boundary  $\Gamma$  and  $\hat{\Gamma}^{\text{des}}$ , e. g.

$$Q^{\text{shape}_2}(\mathbf{U}) := \int_{\hat{\Omega} \setminus \hat{\Omega}^{\text{des}}} d\mathbf{x} + \int_{\hat{\Omega}^{\text{des}} \setminus \hat{\Omega}} d\mathbf{x},$$

the desired domain  $\hat{\Omega}^{\text{des}}$  defined by the boundary  $\hat{\Gamma}^{\text{des}}$ . The problem here is the computation of the domains  $\hat{\Omega} \setminus \hat{\Omega}^{\text{des}}$  and  $\hat{\Omega}^{\text{des}} \setminus \hat{\Omega}$ , as this turns out to be very complex.

Another measure to compare two shapes was suggested by Rumpf and Wirth (2009). They considered the physical work which is needed to transform the current shape into the desired shape. This problem alone is very challenging because there is no mapping between the two shapes which could tell us where the points on the boundary have to be moved to. To use this for a quality functional seems too expensive.

**Regional Penalization** Another example is the penalization of certain regions which should be avoided by the deformed body  $\Omega$ . These could be areas where the deformed body is not allowed or regions away from a desired shape.



**Definition 4.1.4** (Regional Penalization). Let  $q : \mathbb{R}^3 \rightarrow [0, \infty)$  be a continuous function that allocates a certain penalty density  $q(\hat{\mathbf{x}})$  to each spatial point  $\hat{\mathbf{x}} \in \mathbb{R}^3$ . The total penalty by the deformed body then is

$$Q^{\text{pen}}(\mathbf{U}) = \int_{\hat{\Omega}} q(\hat{\mathbf{x}}) d\hat{\mathbf{x}} = \int_{\Omega} q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \det F(\mathbf{x}) d\mathbf{x}. \quad (4.6)$$

This quality functional can also be interpreted as a penalty method for a state-constrained optimal control problem with the constrain that the deformed body must not occupy a certain region.

Computing the derivative with respect to the state  $\mathbf{U}$  directly yields

$$\begin{aligned} Q_{,\mathbf{U}}^{\text{pen}}(\mathbf{U})[\delta\mathbf{U}] &= \int_{\Omega} \widehat{\nabla} q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \det F(\mathbf{x}) \\ &\quad + q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \operatorname{cof} F(\mathbf{x})^{-\top} : \nabla \delta\mathbf{U} d\mathbf{x}. \end{aligned} \quad (4.7)$$

However, this formula does not reveal an important feature of this quality functional: The change of the value of  $Q^{\text{pen}}$  depends only on how the boundary  $\hat{\Gamma}$  changes. For example, if the boundary does not change, that is  $\delta\mathbf{U}|_{\Gamma} = 0$ , the change  $Q_{,\mathbf{U}}^{\text{pen}}(\mathbf{U})[\delta\mathbf{U}]$  is zero as well. Hence, only the change of the exterior shape  $\Gamma$  has an influence on the optimization, not the deformation inside the body. This is proven in the next theorem.

**Theorem 4.1.5** (Change of regional penalization). Let  $q : \mathbb{R}^3 \rightarrow [0, \infty)$  be a continuous function and  $Q^{\text{pen}} : \mathcal{U} \rightarrow [0, \infty)$  with  $Q^{\text{pen}}$  from (4.6). Then the formal directional derivative of  $Q^{\text{pen}}$  is

$$Q_{,\mathbf{U}}^{\text{pen}}(\mathbf{U})[\delta\mathbf{U}] = \int_{\Gamma} q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \delta\mathbf{U}(\mathbf{x})^{\top} \operatorname{cof} F(\mathbf{x}) \mathbf{n}(\mathbf{x}) dS. \quad (4.8)$$

*Proof.* We start by stating the Hadamard representation of shape derivatives to describe a derivative with respect to a perturbation of  $\mathbf{U}$ . This follows the ideas presented in Sokołowski and Zolésio (1992) or (Eppler, 2010, Sec. 2.11.). We consider the quality functional (4.6) formulated of the deformed domain  $\hat{\Omega}$ , that is

$$Q^{\text{pen}}(\mathbf{U}) = \int_{\hat{\Omega}} q(\hat{\mathbf{x}}) d\hat{\mathbf{x}}.$$



One approach for the shape calculus is a *perturbation of identity* on the domain  $\widehat{\Omega}$ : We regard  $\delta\widehat{\mathbf{U}} : \widehat{\Omega} \rightarrow \mathbb{R}^3$  as a perturbation of the deformed domain  $\widehat{\Omega}$ , that is

$$\widehat{\Omega}_\delta := \left\{ \widehat{\mathbf{x}}_\delta = \widehat{\mathbf{x}} + \delta\widehat{\mathbf{U}}(\widehat{\mathbf{x}}), \quad \widehat{\mathbf{x}} \in \widehat{\Omega} \right\}.$$

Rewriting this expression on the undeformed domain and using  $\delta\widehat{\mathbf{U}}(\widehat{\mathbf{x}}(\mathbf{x})) = \delta\mathbf{U}(\mathbf{x})$  yields

$$\widehat{\Omega}_\delta = \{ \widehat{\mathbf{x}}_\delta = \mathbf{x} + \mathbf{U}(\mathbf{x}) + \delta\mathbf{U}(\mathbf{x}), \quad \mathbf{x} \in \Omega \}.$$

One can see that a perturbation  $\delta\widehat{\mathbf{U}}$  of the deformed domain  $\widehat{\Omega}$  can be viewed as a perturbation  $\delta\mathbf{U}$  of the displacement  $\mathbf{U}$ . The shape gradient representation of the Eulerian derivative in direction  $\delta\widehat{\mathbf{U}}$  is

$$Q_{,\mathbf{U}}^{\text{pen}}(\mathbf{U})[\delta\widehat{\mathbf{U}}] = \int_{\widehat{\Gamma}} q(\widehat{\mathbf{x}}) \delta\widehat{\mathbf{U}}(\widehat{\mathbf{x}})^\top \widehat{\mathbf{n}}(\widehat{\mathbf{x}}) d\widehat{S}.$$

This expression is transformed to the undeformed domain, that is

$$\begin{aligned} Q_{,\mathbf{U}}^{\text{pen}}(\mathbf{U})[\delta\mathbf{U}] &= \int_{\Gamma} q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \delta\mathbf{U}(\mathbf{x})^\top \frac{F(\mathbf{x})^{-\top} \mathbf{n}(\mathbf{x})}{\|F(\mathbf{x})^{-\top} \mathbf{n}(\mathbf{x})\|} \det F(\mathbf{x}) \|F(\mathbf{x})^{-\top} \mathbf{n}(\mathbf{x})\| dS \\ &= \int_{\Gamma} q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \delta\mathbf{U}(\mathbf{x})^\top (F(\mathbf{x})^{-\top} \mathbf{n}(\mathbf{x})) \det F(\mathbf{x}) dS. \end{aligned}$$

**Example 4.1.6** (Bar: Regional Penalization). We consider the distance to the desired box in the 1-norm, e. g.

$$q(\widehat{\mathbf{x}}) = d(\widehat{\mathbf{x}}_3, [h, c + h]),$$

where  $d$  is the distance of a point to an interval. We restrict the integral such that only the right quarter is considered, e. g.

$$Q^{\text{pen}}(\mathbf{U}) = \int_{\Omega^{\text{right}}} q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \det F(\mathbf{x}) d\mathbf{x}.$$

**Remark 4.1.7** (Difficulties for the numerical integration). The numerical integration, which we will eventually use to compute the quality functional  $Q^{\text{pen}}$  and its derivative  $Q_{,\mathbf{U}}^{\text{pen}}$ , might face difficulties for certain penalization functions  $q$ . We illustrate this with figure 4.3. In the left sketch the element lies in the region where  $q$  is zero, which is also the result of the numerical integration. The center sketch shows the case where the penalty function  $q$  is still zero in the integration point but the element is partly in the region where  $q$  is positive. Even more, the gradient  $\widehat{\nabla}q$



in the derivative  $Q_{,U}^{\text{pen}}$  from (4.7) is undefined. Here, the numerical and analytical integration yield different results for  $Q^{\text{pen}}$ . The right sketch shows the case if  $Q_{,U}^{\text{pen}}$  is computed by the formula from theorem 4.1.5. Here, the numerical integration over the boundary returns the correct result. Of course, these errors in the computation of  $Q^{\text{pen}}$  become smaller with a finer discretization or by using quadrature formulas that are adapted to this kind of integrals. On the other hand, we expect the deformed body to be very close to the region where  $q > 0$ .

As a consequence, the numerical integration to evaluate  $Q^{\text{pen}}$  might have errors if the deformed body is close to the region where  $q > 0$ . These errors might cause problems in an optimization algorithm which assumes an exact evaluation of the objective function. A remedy to this problem could be a smoothing of  $q$ . This is observed in our numerical experiment in section 6.2.2.

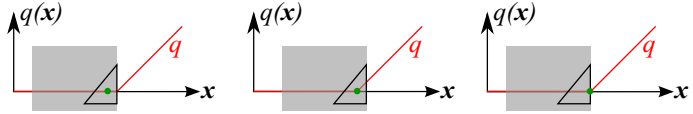


FIGURE 4.3. Illustration of the difficulties for the numerical integration of  $Q^{\text{pen}}$  from (4.6). The black triangle is a finite element with its green integration point and the red curve indicates the value of the penalization function  $q$ .

### Desired Direction

**Example 4.1.8** (Heliotropism). The next example is motivated by biology. In certain flowers, called *heliotropic* flowers, a phenomenon called *heliotropism* causes them to align their flower heads to the sun during the day, that means they move themselves in order to follow the movement of the sun from east to west, see Raven et al. (2001). The movement of the flower is caused by *motor cells* in a flexible part of the stem below the flower. The motor cells control the water balance of nearby cells, for example by pumping potassium ions into these cells and changing the osmotic water flow and thereby the turgor pressure.

The probably most popular example for this flower movement is the sun flower (*Helianthus annuus*), though there is an old misconception: The fully formed head of a sunflower does not follow the sun movement, see the left picture of figure 4.4. The uniform alignment of a blooming field of sunflowers is actually static to the east, making them “living compasses”. The alignment to the east is caused by the fact that only the buds (earlier development stage before the head) track the sun



across the sky. The last alignment of the buds is normally towards east. For more details and experiments we refer to Lang and Begg (1979).

In general, there have been proposed several hypotheses what causes floral heliotropism, e.g. pollinator attraction and warmth management by Kevan (1975), though this might not apply to all flowers, see the work of Totland (1996). Other examples for heliotropism are the arctic poppy (*Papaver radicatum*), the sensitive plant (*Mimosa pudica*) and the meadow buttercup (*Ranunculus acris*).



FIGURE 4.4. Left picture: blooming sunflower at sunset with the sun in its back (photo: Heiko Günnel). Right picture: arctic poppies bending towards the sun (photo: Phil Wickens/Quark Expeditions)

Let us consider an undeformed body  $\Omega$  which points into the direction  $\mathbf{s}$ . We seek to find a deformation such that the body points into a desired direction  $\mathbf{s}^{\text{des}} \in \mathbb{R}^3$ , see figure 4.5.

In order to measure the direction of the body, we choose a surface  $\Gamma^{\mathbf{s}}$  and the tracking-type functional

$$Q^{\mathbf{s}}(\mathbf{U}) = \frac{1}{2} \int_{\Gamma^{\mathbf{s}}} \|\widehat{\mathbf{s}}(\widehat{\mathbf{x}}(\mathbf{x})) - \mathbf{s}^{\text{des}}\|_2^2 \, dS. \quad (4.9)$$

In this case, there are two ways to formulate the direction  $\widehat{\mathbf{s}}$  of the deformed body.

- (1) Let  $\mathbf{s}$  be a direction like a fiber. During the deformation, this direction becomes

$$\widehat{\mathbf{s}}(\widehat{\mathbf{x}}(\mathbf{x})) = \frac{F(\mathbf{x})\mathbf{s}}{\|F(\mathbf{x})\mathbf{s}\|_2}. \quad (4.10)$$



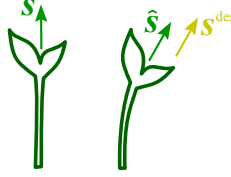


FIGURE 4.5. Sketch of a plant that points towards the direction  $\mathbf{s}$  (left). The direction of the sun is the arrow  $\mathbf{s}^{\text{des}}$  in yellow. Heliotropism causes the plant to bend such that the current direction  $\hat{\mathbf{s}}$  matches the desired direction  $\mathbf{s}^{\text{des}}$  (right).

- (2) The normal  $\mathbf{n}$  on  $\Gamma^{\mathbf{s}}$  is the direction  $\mathbf{s}$  of the undeformed body. The current normal  $\hat{\mathbf{n}}$  of the deformed body

$$\hat{\mathbf{n}}(\hat{\mathbf{x}}(\mathbf{x})) = \frac{F(\mathbf{x})^{-\top} \mathbf{n}(\mathbf{x})}{\|F(\mathbf{x})^{-\top} \mathbf{n}(\mathbf{x})\|_2}, \quad (4.11)$$

then gives the direction  $\hat{\mathbf{s}} = \hat{\mathbf{n}}$  of the deformed body.

Comparing the equations (4.10) and (4.11), they yield the same results for orthogonal matrices  $F$ , that is the deformation gradient is only a rotational matrix on the surface  $\Gamma^{\mathbf{s}}$ . This assumption might be reasonable for certain problems. In our later numerical test in section 6.2.3, we will assume  $F$  to be orthogonal. This is mainly due to the fact that the expected solution changes the shape of the surface  $\Gamma^{\mathbf{s}}$  negligibly. We can then simplify the quality functional to

$$Q^{\mathbf{s}}(\mathbf{U}) = \frac{1}{2} \int_{\Gamma^{\mathbf{s}}} \|F(\mathbf{x})\mathbf{s}(\mathbf{x}) - \mathbf{s}^{\text{des}}\|_2^2 \, dS. \quad (4.12)$$

The first derivative of  $Q^{\mathbf{s}}$  is

$$Q_{,\mathbf{U}}^{\mathbf{s}}(\mathbf{U})[\delta\mathbf{U}] = \int_{\Gamma^{\mathbf{s}}} \mathbf{s}(\mathbf{x})^\top \nabla \delta\mathbf{U}(\mathbf{x})^\top (F(\mathbf{x})\mathbf{s}(\mathbf{x}) - \mathbf{s}^{\text{des}}) \, dS.$$

This simplification has no considerable influence on the solution of the optimal control problem in section (4.10), as comparative computations have shown. Of course, it is also possible to choose a part  $\Omega^{\mathbf{s}} \subset \Omega$  instead of the surface  $\Gamma^{\mathbf{s}}$ , e. g.

$$Q^{\mathbf{s}}(\mathbf{U}) = \frac{1}{2} \int_{\Omega^{\mathbf{s}}} \|\hat{\mathbf{s}}(\hat{\mathbf{x}}(\mathbf{x})) - \mathbf{s}^{\text{des}}\|_2^2 \, d\mathbf{x}.$$

In this case, we take the first formulation of  $\hat{\mathbf{s}}$  because the surface normal  $\mathbf{n}$  is not available for the interior of  $\Omega^{\mathbf{s}}$ .

### Enclosed Volume



Let us consider a thin plate  $\Omega = \Omega^{2D} \times (-d, d)$  of thickness  $2d$  which is clamped at its rim  $\Gamma^D = \partial\Omega^{2D} \times [-d, d]$ . Due to its deformation, a certain volume is gained because the deformed plate makes room for it. A two-dimensional illustration can be found in figure 4.6. We seek to maximize this volume.

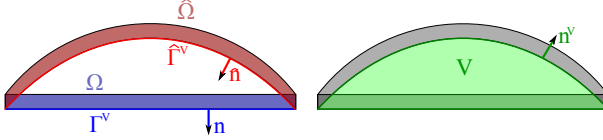


FIGURE 4.6. Two-dimensional sketch of the enclosed volume: the undeformed plate  $\Omega$  in blue, with the surface  $\Gamma^V$  and the normal  $\mathbf{n}$ , and the deformed plate  $\hat{\Omega}$  in red, with the surface  $\hat{\Gamma}^V$  and the normal  $\hat{\mathbf{n}}$ . The volume  $V$  in green is enclosed by the two surfaces  $\Gamma^V$  and  $\hat{\Gamma}^V$ .

To model this, let  $\Gamma^V := \Omega^{2D} \times \{-d\}$  be the lower surface of the undeformed plate and  $\hat{\Gamma}^V := \hat{\mathbf{x}}(\Omega)$  the lower surface of the deformed plate. As the plate is clamped at its rim  $\Gamma^D$ , both surfaces enclose a domain  $V$  with the boundary  $\partial V = \Gamma^V \cup \hat{\Gamma}^V$ . We are interested in the volume of  $V$ , that is

$$|V| = \int_V 1 \, d\mathbf{x}.$$

As the domain  $V$  depends on  $\hat{\Gamma}^V$ , which itself depends on the displacement  $\mathbf{U}$ , we can use the volume  $|V|$  for a quality functional  $Q^V(\mathbf{U})$ . However, we need a representation of the volume  $|V|$  in terms of the displacement  $\mathbf{U}$ .

**Theorem 4.1.9** (Volume enclosed by a surface). Let  $V$  be a domain enclosed by the surfaces  $\Gamma^V$  and  $\hat{\Gamma}^V$ . The volume of  $V$  is

$$|V| = \int_{\Gamma^V} \mathbf{x}_3 [\mathbf{n}(\mathbf{x})]_3 \, dS - \int_{\hat{\Gamma}^V} \hat{\mathbf{x}}_3 [\hat{\mathbf{n}}(\hat{\mathbf{x}})]_3 \, d\hat{S}, \quad (4.13)$$

with the outer normals  $\mathbf{n}$  and  $\hat{\mathbf{n}}$  of the surfaces  $\Gamma^V$  respectively  $\hat{\Gamma}^V$ .

*Proof.* Green's identity from lemma (A.1) with the function  $(0, 0, \mathbf{x}_3)^\top$  over the domain  $V$  reads

$$|V| = \int_V 1 \, d\mathbf{x} = \int_V \operatorname{div}(0, 0, \mathbf{x}_3)^\top \, d\mathbf{x} = \int_{\partial V} (0, 0, \mathbf{x}_3) \mathbf{n}^V(\mathbf{x}) \, dS = \int_{\partial V} \mathbf{x}_3 [\mathbf{n}^V(\mathbf{x})]_3 \, dS,$$



where  $\mathbf{n}^V$  is the outer normal of  $V$ , which is  $\mathbf{n}^V = \mathbf{n}$  on  $\Gamma^V$  and  $\mathbf{n}^V = -\hat{\mathbf{n}}$  on  $\hat{\Gamma}^V$ , see figure 4.6. Finally, we insert the splitting  $\partial V = \Gamma^V \cup \hat{\Gamma}^V$  and obtain

$$|V| = \int_{\Gamma^V} \mathbf{x}_3[\mathbf{n}(\mathbf{x})]_3 \, dS - \int_{\hat{\Gamma}^V} \hat{\mathbf{x}}_3[\hat{\mathbf{n}}(\hat{\mathbf{x}})]_3 \, d\hat{S},$$

as claimed.

The first term of the sum in (4.13) can be simplified to

$$\int_{\Gamma^V} \mathbf{x}_3[\mathbf{n}(\mathbf{x})]_3 \, dS = \int_{\Omega^{2D}} d \, d\mathbf{x}_1 d\mathbf{x}_2, \quad (4.14)$$

because  $\mathbf{x}_3 = -d$  and  $\mathbf{n} = (0, 0, -1)^\top$  on  $\Gamma^V = \Omega^{2D} \times \{-d\}$ . The second term can be transformed back to the current configuration and simplified to

$$\begin{aligned} \int_{\hat{\Gamma}^V} \hat{\mathbf{x}}_3[\hat{\mathbf{n}}(\hat{\mathbf{x}})]_3 \, d\hat{S} &= \int_{\Gamma^V} (\mathbf{x}_3 + [\mathbf{U}(\mathbf{x})]_3) [\operatorname{cof} F(\mathbf{x})\mathbf{n}(\mathbf{x})]_3 \, dS \\ &= \int_{\Omega^{2D}} (-d + [\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, -d)]_3) [\operatorname{cof} F(\mathbf{x}_1, \mathbf{x}_2, -d)(0, 0, -1)^\top]_3 \, d\mathbf{x}_1 d\mathbf{x}_2 \\ &= - \int_{\Omega^{2D}} (-d + [\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, -d)]_3) [\operatorname{cof} F(\mathbf{x}_1, \mathbf{x}_2, -d)]_{33} \, d\mathbf{x}_1 d\mathbf{x}_2. \end{aligned}$$

Finally, the quality functional to describe the volume that is enclosed by  $\Gamma^V$  and  $\hat{\Gamma}^V$  with respect to the deformation  $\mathbf{U}$  reads

$$Q^V(\mathbf{U}) := \int_{\Omega^{2D}} d + (-d + [\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, -d)]_3) [\operatorname{cof} F(\mathbf{x}_1, \mathbf{x}_2, -d)]_{33} \, d\mathbf{x}_1 d\mathbf{x}_2. \quad (4.15)$$

This formula could be directly differentiated with respect to  $\mathbf{U}$ , however, there is an easier way to obtain the first derivative  $Q_{,\mathbf{U}}^V$ .

**Theorem 4.1.10** (First derivative of the enclosed volume). Let  $Q^V$  be given from (4.15). Its formal first derivative with respect to the displacement  $\mathbf{U}$  is

$$Q_{,\mathbf{U}}^V(\mathbf{U})[\delta\mathbf{U}] := \int_{\Omega^{2D}} \delta\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, -d)^\top \operatorname{cof} F(\mathbf{x}_1, \mathbf{x}_2, -d)(0, 0, -1)^\top \, d\mathbf{x}_1 d\mathbf{x}_2. \quad (4.16)$$

*Proof.* Again, we use the Hadamard representation as in the proof of theorem 4.1.5. The perturbation  $\delta\hat{\mathbf{U}} : \hat{\Omega} \rightarrow \mathbb{R}^3$  restricted on the boundary  $\hat{\Gamma}^V$  will be the *perturbation of identity* of the domain  $V$ . That means a point  $\hat{\mathbf{x}} \in \hat{\Gamma}^V$  on the



boundary  $\widehat{\Gamma}^V$  is moved to  $\widehat{\mathbf{x}} + \delta\widehat{\mathbf{U}}(\widehat{\mathbf{x}})$ . This is consistent with  $\delta\mathbf{U}$  being a perturbation of the displacement  $\mathbf{U}$ , because

$$\widehat{\mathbf{x}}(\mathbf{x}) + \delta\widehat{\mathbf{U}}(\widehat{\mathbf{x}}(\mathbf{x})) = \mathbf{x} + \mathbf{U}(\mathbf{x}) + \delta\mathbf{U}(\mathbf{x}).$$

The quality functional  $Q^V$  is the volume of the domain  $V$ , so we have

$$Q^V = |V| = \int_V 1 \, d\mathbf{x}.$$

Sokołowski and Zolésio (1992) or (Eppler, 2010, sec. 2.11.) state the Hadamard representation of the shape gradient of the Eulerian derivative of  $|V|$  in direction  $\delta\widehat{\mathbf{U}}$ , that is

$$(|V|)'[\delta\widehat{\mathbf{U}}] = \int_{\widehat{\Gamma}^V} \delta\widehat{\mathbf{U}}(\widehat{\mathbf{x}})^\top \widehat{\mathbf{n}}(\widehat{\mathbf{x}}) \, d\widehat{S},$$

because the perturbation only acts on the boundary  $\widehat{\Gamma}^V$  and thus the part of the integral over  $\Gamma^V$  vanishes. We transform the integral back to the current configuration and obtain the first derivative

$$Q_{,U}^V(\mathbf{U})[\delta\mathbf{U}] := \int_{\Gamma^V} \delta\mathbf{U}(\mathbf{x})^\top \operatorname{cof} F(\mathbf{x}) \mathbf{n}(\mathbf{x}) \, dS.$$

Inserting  $\Gamma^V = \Omega^{2D} \times \{-d\}$  yields the claim

$$Q_{,U}^V(\mathbf{U})[\delta\mathbf{U}] := \int_{\Omega^{2D}} \delta\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, -d)^\top \operatorname{cof} F(\mathbf{x}_1, \mathbf{x}_2, -d) (0, 0, -1)^\top \, d\mathbf{x}_1 d\mathbf{x}_2.$$

#### 4.1.2. Cost or Penalty Functionals

The cost or penalty functional has to be chosen for the control  $\mathbf{C}$ , otherwise the optimal control problem might be unbounded. We will consider the standard squared  $L_2$ -norm of the control, that is

$$\begin{aligned} P^{(f)}(\mathbf{f}) &:= \int_{\Omega} \gamma^{(f)}(\mathbf{x}) \mathbf{f}(\mathbf{x})^\top \mathbf{f}(\mathbf{x}) \, d\mathbf{x}, & \text{with } \gamma^{(f)} : \Omega &\rightarrow (0, \infty), \\ P^{(g)}(\mathbf{g}) &:= \int_{\Gamma} \gamma^{(g)}(\mathbf{x}) \mathbf{g}(\mathbf{x})^\top \mathbf{g}(\mathbf{x}) \, dS, & \text{with } \gamma^{(g)} : \Gamma &\rightarrow (0, \infty), \\ P^{(t)}(t) &:= \int_{\Omega} \gamma^{(t)}(\mathbf{x}) (t(\mathbf{x}))^2 \, d\mathbf{x}, & \text{with } \gamma^{(t)} : \Omega &\rightarrow (0, \infty), \\ P^{(m)}(m) &:= \int_{\Omega} \gamma^{(m)}(\mathbf{x}) (m(\mathbf{x}))^2 \, d\mathbf{x}, & \text{with } \gamma^{(m)} : \Omega &\rightarrow (0, \infty). \end{aligned} \tag{4.17}$$



To unify the notation for the various controls, we simply write  $P^{(\mathcal{C})}$  and  $\gamma^{(\mathcal{C})}$ .

**Remark 4.1.11.** In some cases, the control might not be defined on the whole domain  $\Omega$  respectively  $\Gamma^N$ . In such cases the control  $\mathcal{C}$  can be restricted to  $\Omega^{\text{ctrl}} \subset \Omega$  respectively  $\Gamma^{\text{ctrl}} \subset \Gamma^N$  by restricting the integral in the energy functional  $W^{(\mathcal{C})}$ , e. g.

$$W^{(\mathcal{f})}(\mathbf{U}, \mathbf{f}) = \int_{\Omega^{\text{ctrl}}} \mathbf{f}^\top \mathbf{U} \, d\mathbf{x}.$$

## 4.2 Lagrange-Newton: An All-at-once Approach

Our first method to solve the optimal control problem (4.2) is a Lagrange-Newton method, see Alt (1990) and Alt and Malanowski (1993) for a reference. The method is based on the Lagrange functional and seeks to find a root of the first derivative.

**Definition 4.2.1** (Lagrange functional). Let the optimal control problem (4.2) be given. The corresponding *Lagrange function*  $L : \mathcal{U} \times \mathcal{C} \times \mathcal{U} \rightarrow \mathbb{R}$  to this constrained optimization problem is

$$L(\mathbf{U}, \mathbf{C}, \mathbf{Z}) := I(\mathbf{U}, \mathbf{C}) + W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\mathbf{Z}], \quad (4.18)$$

with the state  $\mathbf{U} \in \mathcal{U}$ , the state  $\mathbf{C} \in \mathcal{C}$  and the multiplier  $\mathbf{Z} \in \mathcal{U}$ . We abbreviate the triple  $X := (\mathbf{U}, \mathbf{C}, \mathbf{Z})$  and  $X \in \mathcal{X} := \mathcal{U} \times \mathcal{C} \times \mathcal{U}$ .

Under certain regularity assumptions, see Alt (1990), the Lagrange principle states that a minimum  $X^* = (\mathbf{U}^*, \mathbf{C}^*, \mathbf{Z}^*)$  of the optimal control problem satisfies

$$L_{,X}(X^*)[\delta X] = 0 \quad \forall \delta X \in \mathcal{X}. \quad (4.19)$$

The formal first derivatives  $L_{,X}$  are

$$\begin{aligned} L_{,\mathbf{U}}(\mathbf{U}, \mathbf{C}, \mathbf{Z})[\delta \mathbf{U}] &= I_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\delta \mathbf{U}] + W_{,\mathbf{U}\mathbf{U}}(\mathbf{U}, \mathbf{C})[\mathbf{Z}, \delta \mathbf{U}] \\ L_{,\mathbf{C}}(\mathbf{U}, \mathbf{C}, \mathbf{Z})[\delta \mathbf{C}] &= I_{,\mathbf{C}}(\mathbf{U}, \mathbf{C})[\delta \mathbf{C}] + W_{,\mathbf{U}\mathbf{C}}(\mathbf{U}, \mathbf{C})[\mathbf{Z}, \delta \mathbf{C}] \\ L_{,\mathbf{Z}}(\mathbf{U}, \mathbf{C}, \mathbf{Z})[\delta \mathbf{Z}] &= W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\delta \mathbf{Z}]. \end{aligned} \quad (4.20)$$

The behavior of the Lagrange-Newton method is mesh-independent under certain assumptions, see Alt (2001). This implies that the convergence behavior will not worsen on sufficiently refined meshes. For a general discussion on local convergence we refer to Alt (1994). As we already mentioned in remark 3.3.4, the local quadratic convergence of Newton's method is with respect to the iteration number assuming that the Newton system is solved accurately. If we use an inexact Newton's method, the computation costs tend to grow for the last iterations because the Newton system has to be solved more and more accurately.



### 4.2.1. Solving the Lagrange Equation

The nonlinear *Lagrange system* (4.19) can be solved by Newton's method, that is

$$X_{k+1} = X_k + \alpha_k \Delta X_k, \quad (4.21)$$

with a step length  $\alpha_k$  and the search direction  $\Delta X$  satisfying

$$L_{,XX}(X_k)[\delta X, \Delta X_k] = -L_{,X}(X_k)[\delta X] \quad \forall \delta X \in \mathcal{X}. \quad (4.22)$$

This leads to algorithm 4.2.2.

**Algorithm 4.2.2** (Lagrange-Newton method to solve  $L_{,X}(X^*) = 0$ ).

**Input:** initial  $X_0$ , Lagrange functional  $L$ ,  $k = 0$

**Output:**  $X := (U, C, Z)$

- 1: **while** ( $\|L_{,X}(X_k)\|_{\mathcal{X}^*}$  is too large) **do**
- 2:   get  $\Delta X_k$  by solving  $L_{,XX}(X_k)[\delta X, \Delta X_k] = -L_{,X}(X_k)[\delta X]$
- 3:   choose step length  $\alpha_k$  by a line search
- 4:   set  $X_{k+1} := X_k + \alpha \Delta X_k$
- 5:   set  $k := k + 1$
- 6: **end while**
- 7: **return**  $X_k := (U_k, C_k, Z_k)$

As a stopping criterion, we choose a relative or absolute tolerance for the right-hand side  $L_{,X}(X_k)$ , e. g.

$$\frac{\|L_{,X}(X_k)\|_{\mathcal{X}^*}}{\|L_{,X}(X_0)\|_{\mathcal{X}^*}} \leq \text{rTol} \quad \text{or} \quad \|L_{,X}(X_k)\|_{\mathcal{X}^*} \leq \text{aTol}, \quad (4.23)$$

with a user defined relative tolerance  $\text{rTol} > 0$  and absolute tolerance  $\text{aTol} > 0$ , which do not have to be the same as the tolerances for the MINRES method.

Since all derivatives are evaluated at the same point  $X_k = (U_k, C_k, Z_k)$ , we drop the references to the point  $X_k$  and to the perturbation  $\delta X = (\delta U, \delta C, \delta Z)$  for better readability. The Lagrange-Newton system then reads

$$\begin{bmatrix} L_{,UU} & \mathcal{L}_{,UC} & \mathcal{L}_{,UZ} \\ L_{,CU} & \mathcal{L}_{,CC} & \mathcal{L}_{,CZ} \\ L_{,ZU} & \mathcal{L}_{,ZC} & 0 \end{bmatrix} \begin{bmatrix} \Delta U_k \\ \Delta C_k \\ \Delta Z_k \end{bmatrix} = - \begin{bmatrix} L_{,U} \\ L_{,C} \\ L_{,Z} \end{bmatrix}. \quad (4.24)$$



The single entries of the Newton matrix from (4.24), evaluated at the point  $X_k = (U_k, C_k, Z_k)$ , are

$$\begin{aligned}
 L_{,UU}[\delta U, \Delta U_k] &= I_{,UU}[\delta U, \Delta U_k] + W_{,UUU}[Z_k, \delta U, \Delta U_k] \\
 L_{,UC}[\delta U, \Delta C_k] &= I_{,UC}[\delta U, \Delta C_k] + W_{,UUC}[Z_k, \delta U, \Delta C_k] \\
 L_{,UZ}[\delta U, \Delta Z_k] &= W_{,UU}[\Delta Z_k, \delta U] \\
 \\ 
 L_{,CU}[\delta C, \Delta U_k] &= I_{,CU}[\delta C, \Delta U_k] + W_{,UCU}[Z_k, \delta C, \Delta U_k] \\
 L_{,CC}[\delta C, \Delta C_k] &= I_{,CC}[\delta C, \Delta C_k] + W_{,UCC}[Z_k, \delta C, \Delta C_k] \\
 L_{,CZ}[\delta C, \Delta Z_k] &= W_{,UC}[\Delta Z_k, \delta C] \\
 \\ 
 L_{,ZU}[\delta Z, \Delta U_k] &= W_{,UU}[\delta Z, \Delta U_k] \\
 L_{,ZC}[\delta Z, \Delta C_k] &= W_{,UC}[\delta Z, \Delta C_k].
 \end{aligned} \tag{4.25}$$

Please note that the mixed derivative  $I_{,CU}$  are zero for the objective functional  $I(U, C) = Q(U) + P(C)$  and that for the presented controls from table 2.1 the second derivatives  $W_{,CC}$  are zero as well. This simplifies the system to

$$\begin{bmatrix} I_{,UU} + W_{,UUU} & W_{,UUC} & W_{,UU} \\ W_{,UCU} & 0 & W_{,UC} \\ W_{,UU} & W_{,UC} & 0 \end{bmatrix} \begin{bmatrix} \Delta U_k \\ \Delta C_k \\ \Delta Z_k \end{bmatrix} = - \begin{bmatrix} I_{,U} + W_{,UU} \\ I_{,C} + W_{,UC} \\ W_{,U} \end{bmatrix}.$$

In order to solve the Lagrange-Newton system (4.24), it is important to know that the corresponding linear operator  $L_{,XX} \in \mathcal{L}(\mathcal{X}, \mathcal{X}^*)$  is symmetric and indefinite. Therefore, we apply the MINRES method from section 3.3 to solve the system (4.24). As the MINRES method depends on the inner product of  $\mathcal{X} = \mathcal{U} \times \mathcal{C} \times \mathcal{U}$ , we choose the canonical product induced by the inner products of the state space  $\mathcal{U}$  and the control space  $\mathcal{C}$ , e. g.

$$(X_1, X_2)_{\mathcal{X}} := s_U (U_1, U_2)_{\mathcal{U}} + s_C (C_1, C_2)_{\mathcal{C}} + s_Z (Z_1, Z_2)_{\mathcal{U}}, \tag{4.26}$$

with  $X_i = (U_i, C_i, Z_i)$ ,  $i = 1, 2$ , the inner products  $(\cdot, \cdot)_{\mathcal{U}}$  from (2.28) and  $(\cdot, \cdot)_{\mathcal{C}}$  from table 2.2, and scaling factors  $s_C, s_U, s_Z > 0$ . This choice is equivalent to the block preconditioner

$$P = \begin{bmatrix} s_U K_{\text{lin.elast}} & & \\ & s_C M_C & \\ & & s_Z K_{\text{lin.elast}} \end{bmatrix}, \tag{4.27}$$

with the stiffness matrix  $K_{\text{lin.elast}}$  from the linear elasticity model and the mass matrix  $M_C$  induced by the inner product  $(\cdot, \cdot)_{\mathcal{C}}$ . The scaling factors  $s_C, s_U, s_Z$  are user-defined such that the block preconditioner  $P$  is “similar” to the Lagrange-Newton matrix from the operator  $L_{,XX}$ . Instead of an analysis of the lower and



upper bound in the spectral equivalence between these two matrices, we run numerical tests on the convergence behavior of the overall algorithm for a collection of scalings  $s_C, s_U, s_Z$  and choose the scaling triple with the smallest number of Krylov iterations. This testing procedure is reduced by setting  $s_U = 1$  and varying  $s_C, s_Z$  as the overall scaling of the preconditioner  $P$  has no influence on the convergence behavior. However, one has to be aware that as the preconditioner  $P$  defines the dual norm, e. g.  $\|L_{,X}(X_k)\|_{P^{-1}}$ , the stopping criteria changes too. More details will be given in section ??.

### Line Search

The Lagrange-Newton step (4.21) has a step length  $\alpha_k$  which has not been discussed yet. Fixing  $\alpha_k = 1$  might be a bad idea as we risk failure of the Lagrange-Newton algorithm 4.2.2 due to the nonlinearity of the Lagrange-Newton function itself or due to possible negative local volume change  $J < 0$  (which renders the elasticity model physically meaningless). There are some methods to deal with these problems:

- The problem with a negative local volume change  $J < 0$  can be redeemed by using the guiding criterion from (3.4).
- As discussed in (Nocedal and Wright, 2006, ch. 15.4-15.6), we could use a merit function and add a penalty functional for the equation  $0 = W_U(\mathbf{U}, \mathbf{C})$  to the objective functional and employ a strategy to control the factor in front of the penalty functional.
- Another method is an augmented Lagrangian-SQP (sequential quadratic programming) suggested by Ito and Kunisch (1996a), Ito and Kunisch (1996b) and others. Here we add a penalty functional on the multiplier  $\mathbf{Z}$  to the original Lagrange function, for more details we refer to the work cited before.

In our later numerical experiments in section 6.2, the guiding criterion was sufficient for the convergence of the Lagrange-Newton method. However, if it turns out that a more sophisticated method is necessary, a merit function like in (Nocedal and Wright, 2006, ch. 15.4-15.6) could be implemented.

#### 4.2.2. Discretization

To solve the first order optimality  $L_{,X} = 0$  from (4.19) numerically, we use a finite element method to discretize it, like we did for the forward problem in section 3.4. We replace the space  $\mathcal{X}$  by the discretizations of state and control space, that is  $\mathcal{X}_h := \mathcal{U}_h \times \mathcal{C}_h \times \mathcal{U}_h$ . The finite element base of  $\mathcal{X}_h$  is denoted by  $\{\varphi_i\}_{i=1}^{N_{\mathcal{X}}}$ , with its dimension  $N_{\mathcal{X}} = \dim \mathcal{X}_h = N_{\mathcal{U}} + N_{\mathcal{C}} + N_{\mathcal{U}}$ . The first  $N_{\mathcal{U}}$  basis functions are the ones for  $\mathbf{U}$ , the next  $N_{\mathcal{C}}$  basis functions for  $\mathbf{U}$  and the last  $N_{\mathcal{U}}$  basis functions for  $\mathbf{Z}$ . The Lagrange system (4.22) in  $\mathcal{X}_h$  becomes the linear system of equations

$$A\vec{x} = \vec{b}, \quad (4.28)$$



with the coefficient matrix  $A \in \mathbb{R}^{N_{\mathcal{X}} \times N_{\mathcal{X}}}$ , the solution vector  $\vec{x} \in \mathbb{R}^{N_{\mathcal{X}}}$  and the right-hand side  $\vec{b} \in \mathbb{R}^{N_{\mathcal{X}}}$ , that is

$$\begin{aligned} [A]_{ij} &= L_{,XX}[\varphi_i, \varphi_j] & i, j = 1, \dots, N_{\mathcal{X}} \\ [\vec{b}]_i &= L_{,X}[\varphi_i] & i = 1, \dots, N_{\mathcal{X}}. \end{aligned} \quad (4.29)$$

The discretized solution then is

$$X_h = \sum_{i=1}^{N_{\mathcal{X}}} [\vec{x}]_i \varphi_i \in \mathcal{X}. \quad (4.30)$$

The block preconditioner from (4.27) is applied separately for the three blocks. Solving with the stiffness matrix  $K_{\text{lin,elast}}$  is approximated by a V-cycle of a multigrid method, see 3.5. The mass matrix  $M_C$  is a diagonal matrix for  $\mathcal{P}_0$ -elements, hence we can directly solve this block.

### Nested Iterations

We can apply the idea of nested iterations (alg. 3.5.6) for the forward solver for the Lagrange-Newton again, see algorithm 4.2.3.

**Algorithm 4.2.3** (Lagrange-Newton method with nested iterations).

**Input:** initial  $X_0^{(0)}$ , nested spaces  $\mathcal{X}_0 \subset \mathcal{X}_1 \subset \dots \subset \mathcal{X}_L$

**Output:**  $X^{(L)}$

```

1: for  $\ell := 0, 1, \dots, L$  do
2:   compute  $X^\ell$  by solving prob. (4.2) with alg. 4.2.2 starting with  $X_0^{(\ell)}$ 
3:   if  $\ell < L$  then
4:     set  $X_0^{(\ell+1)} := p_{\ell \rightarrow \ell+1} X^{(\ell)} \in \mathcal{X}_{(\ell+1)}$ 
5:   else
6:     return  $X^{(L)}$ 
7:   end if
8: end for
```

## 4.3 Quasi-Newton: A Reduced Formulation

The idea of a reduced formulation is to reformulate the constrained optimization problem (4.2) into a unconstrained optimization problem, for example see Hinze and Kunisch (2004) and Herzog and Kunisch (2010). This is done by introducing a solution operator to eliminate the constraint  $0 = W_{,\mathcal{U}}$  from the optimization problem.



**Definition 4.3.1** (Solution operator). Let the state equation  $0 = W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})$  be given. We assume the existence and (local) uniqueness of the state  $\mathbf{U} \in \mathcal{U}$  for every given  $\mathbf{C} \in \mathcal{C}$ . Then we can define a *solution operator*  $S : \mathcal{C} \rightarrow \mathcal{U}$  by

$$S(\mathbf{C}) = \mathbf{U} \quad \Leftrightarrow \quad 0 = W_{,\mathbf{U}}(\mathbf{U}, \mathbf{C})[\delta \mathbf{U}] \quad \forall \delta \mathbf{U} \in \mathcal{U}, \quad (4.31)$$

that means  $\mathbf{U}$  solves the state equation for a given  $\mathbf{C}$ .

**Remark 4.3.2.** The assumption of existence of a solution  $\mathbf{U} \in \mathcal{U}$  for a given  $\mathbf{C} \in \mathcal{C}$  is ensured by the Ball's existence theorem 2.2.9. Still we assume the local uniqueness of the solution by the following reasoning. In order to solve the state equation, we use Newton's method (algorithm 3.1.1) which gives us a single solution  $\mathbf{U}$  for a given  $\mathbf{C}$ . See remark 2.2.11 for a brief discussion.

With the solution operator  $S$ , we can eliminate the constraint  $0 = W_{,\mathbf{U}}$  by replacing the state  $\mathbf{U}$  with  $S(\mathbf{C})$ , which results in a free optimization problem.

**Definition 4.3.3** (Reduced Optimal Control Problem). Let the optimal control problem (4.2) be given. The *reduced optimal control problem* is

$$\min_{\mathbf{C} \in \mathcal{C}} I^{\text{red}}(\mathbf{C}) := I(S(\mathbf{C}), \mathbf{C}), \quad (4.32)$$

with the corresponding solution operator  $S : \mathcal{C} \rightarrow \mathcal{U}$ .

Later, we want to apply derivative based optimization methods, which would require the computation of the derivative  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C})$  or the gradient  $\nabla_{\mathcal{C}} I^{\text{red}}(\mathbf{C})$  of the reduced objective function. Before we continue we would like to point out the difference between derivative and gradient and their relationship. The derivative of the reduced objective functional  $I^{\text{red}}$  with respect to  $\mathbf{C}$  is a linear, bounded operator  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}) \in \mathcal{C}^*$ , while the gradient  $\nabla_{\mathcal{C}} I^{\text{red}}(\mathbf{C}) \in \mathcal{C}$  is an element of the primal control space  $\mathcal{C}$ . They both are linked by the linear equation

$$[I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}), \delta \mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} = \left( \nabla_{\mathcal{C}} I^{\text{red}}(\mathbf{C}), \delta \mathbf{C} \right)_{\mathcal{C}} \quad \forall \delta \mathbf{C} \in \mathcal{C}. \quad (4.33)$$

Equation (4.33) basically states that the gradient is the Riesz representative of the derivative in the Hilbert space  $\mathcal{C}$ . For a discretized control space  $\mathcal{C}_h$ , solving this linear equation is basically solving with the mass matrix of  $\mathcal{C}_h$ , which is very cheap in the case of  $\mathcal{P}_0$ -elements because the mass matrix is a diagonal matrix.

The question remains, whether we use the derivative or the gradient in the derivative based optimization algorithms. Although the term “derivative based“ optimization might suggest the use of the derivative, many algorithms actually work with the gradient. For example, the steepest descent method chooses the search direction to be  $-\nabla_{\mathcal{C}} I^{\text{red}}(\mathbf{C}) \in \mathcal{C}$ . It cannot use the derivative  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}) \in \mathcal{C}^*$  directly as a search direction because the search direction has to be an element of the control space  $\mathcal{C}$ ,



so  $I_{\mathcal{C}}^{\text{red}}(\mathbf{C}) \in \mathcal{C}^*$  cannot be used in the update on the current solution. However, this incompatibility of the derivative as a search direction is often overlooked in the space  $\mathbb{R}^n$  with the Euclidean scalar product. Here, the mass matrix is an identity matrix and therefore the derivative and the gradient are identified as the same. On the other hand, there are methods that use the derivative instead of the gradient because the derivative is not directly used textitas the search direction but rather used *to compute* the search direction. An example is Newton's method in optimization. Here, the derivative is the right-hand side of a Newton system with the search direction as the solution.

FENICS, the program that we use to implement our algorithms, offers the possibility of automatic differentiation (AD, also called algorithmic differentiation) which makes the computation of the derivatives very easy for the user. Since we can directly compute the derivatives and we use a quasi-Newton method later, it seems more natural to work with derivatives instead of the gradients. Later, this choice will have consequences for the adjoint operator (otherwise it would be the Hilbert space adjoint) and for the formulation of the BFGS update.

In order to compute the derivative  $I_{\mathcal{C}}^{\text{red}}$ , we apply the chain rule, that is

$$\begin{aligned} I_{\mathcal{C}}^{\text{red}}(\mathbf{C})[\delta\mathbf{C}] &= \frac{\partial}{\partial\mathbf{C}} (I(S(\mathbf{C}), \mathbf{C}))[\delta\mathbf{C}] \\ &= I_{,U}(S(\mathbf{C}), \mathbf{C})[S_{,C}(\mathbf{C})[\delta\mathbf{C}]] + I_{,C}(S(\mathbf{C}), \mathbf{C})[\delta\mathbf{C}] \\ &= [I_{,U}(S(\mathbf{C}), \mathbf{C}), S_{,C}(\mathbf{C})[\delta\mathbf{C}]]_{\mathcal{U}^*, \mathcal{U}} + [I_{,C}(S(\mathbf{C}), \mathbf{C}), \delta\mathbf{C}]_{\mathcal{C}^*, \mathcal{C}}, \end{aligned} \quad (4.34)$$

with a linearization  $S_{,C}(\mathbf{C})$  of the solution operator, that is a linear (bounded) operator  $S_{,C}(\mathbf{C}) \in \mathcal{L}(\mathcal{C}, \mathcal{U})$  at the current control  $\mathbf{C}$ . Before we continue with the derivation of a representation of  $I_{\mathcal{C}}^{\text{red}}(\mathbf{C})$ , we show how the linear operator  $S_{,C}(\mathbf{C})$  can be computed. In the following, we abbreviate the current point by  $(\sim) := (S(\mathbf{C}), \mathbf{C})$  for easier reading.

**Theorem 4.3.4** (The linearized solution operator  $S_{,C}(\mathbf{C})$ ). The (formal) linearization  $S_{,C}(\mathbf{C}) \in \mathcal{L}(\mathcal{C}, \mathcal{U})$  of the solution operator  $S : \mathcal{C} \rightarrow \mathcal{U}$  in (4.31) at the point  $\mathbf{C}$  is

$$\begin{aligned} S_{,C}(\mathbf{C})[\delta\mathbf{C}] = \delta\mathbf{U} \iff 0 &= W_{,UU}(S(\mathbf{C}), \mathbf{C})[\mathbf{V}, \delta\mathbf{U}] \\ &\quad + W_{,UC}(S(\mathbf{C}), \mathbf{C})[\mathbf{V}, \delta\mathbf{C}] \quad \forall \mathbf{V} \in \mathcal{U}. \end{aligned} \quad (4.35)$$

*Proof.* We regard  $\delta\mathbf{C}$  as a small perturbation of the control  $\mathbf{C}$  and seek to determine the change  $\delta\mathbf{U}$  for the state  $\mathbf{U} = S(\mathbf{C})$ , that is

$$0 = W_{,UU}(\sim)[\mathbf{V}, S_{,C}(\mathbf{C})[\delta\mathbf{C}]] + W_{,UC}(\sim)[\mathbf{V}, \delta\mathbf{C}] \quad \forall \mathbf{V} \in \mathcal{U}.$$



Setting  $\delta \mathbf{U} := S_{,\mathcal{C}}(\mathbf{C})[\delta \mathbf{C}]$  yields (4.35). Under the assumption of the Fréchet-differentiability of  $W_{,\mathcal{U}}$ , this can also be viewed as the application of the implicit function theorem, which is stated for example in Zeidler (1986).

In order to get a practical representation of  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C})$ , we need to shift the linearized solution operator  $S_{,\mathcal{C}}(\mathbf{C})[\delta \mathbf{C}]$  in equation (4.34) onto  $I_{,\mathcal{U}}$ . This can be done by the adjoint operator.

**Definition 4.3.5** (Adjoint Operator). Let  $\mathcal{U}$  and  $\mathcal{C}$  be Hilbert spaces with the duality pairings  $[\cdot, \cdot]_{\mathcal{U}^*, \mathcal{U}}$ , respectively  $[\cdot, \cdot]_{\mathcal{C}^*, \mathcal{C}}$ . Furthermore, let  $S_{,\mathcal{C}}(\mathbf{C}) \in \mathcal{L}(\mathcal{C}, \mathcal{U})$  be a linear operator. The *adjoint operator*  $S_{,\mathcal{C}}(\mathbf{C})^* \in \mathcal{L}(\mathcal{U}^*, \mathcal{C}^*)$  is defined by

$$[\mathbf{V}, S_{,\mathcal{C}}(\mathbf{C})[\delta \mathbf{C}]]_{\mathcal{U}^*, \mathcal{U}} = [S_{,\mathcal{C}}(\mathbf{C})^* \mathbf{V}, \delta \mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} \quad \forall \delta \mathbf{C} \in \mathcal{C}, \mathbf{V} \in \mathcal{U}^*. \quad (4.36)$$

The next theorem states an equation to represent the adjoint  $S_{,\mathcal{C}}(\mathbf{C})^*$ .

**Theorem 4.3.6** (Adjoint of the linearized solution operator  $S_{,\mathcal{C}}(\mathbf{C})$ ). Let the linearized solution operator  $S_{,\mathcal{C}}(\mathbf{C}) \in \mathcal{L}(\mathcal{C}, \mathcal{U})$  from (4.35) be given. The adjoint operator  $S_{,\mathcal{C}}(\mathbf{C})^* \in \mathcal{L}(\mathcal{U}^*, \mathcal{C}^*)$  is defined by

$$\begin{aligned} S_{,\mathcal{C}}(\mathbf{C})^*[I_{,\mathcal{U}}(\sim)] &:= W_{,\mathcal{U}\mathcal{C}}(\sim)[\mathbf{Z}, \cdot] \\ \text{with} \quad W_{,\mathcal{U}\mathcal{U}}(\sim)[\mathbf{V}, \mathbf{Z}] + I_{,\mathcal{U}}(\sim)[\mathbf{V}] &= 0 \quad \forall \mathbf{V} \in \mathcal{U}. \end{aligned} \quad (4.37)$$

*Proof.* Inserting  $\mathbf{V} = \mathbf{Z}$  into the linearized solution operator (4.35) and inserting  $\mathbf{V} = \delta \mathbf{U}$  into the adjoint equation (4.37) yields

$$\begin{aligned} W_{,\mathcal{U}\mathcal{U}}(\sim)[\mathbf{Z}, \delta \mathbf{U}] + W_{,\mathcal{U}\mathcal{C}}(\sim)[\mathbf{Z}, \delta \mathbf{C}] &= 0 \\ \text{and} \quad W_{,\mathcal{U}\mathcal{U}}(\sim)[\delta \mathbf{U}, \mathbf{Z}] + I_{,\mathcal{U}}(\sim)[\delta \mathbf{U}] &= 0. \end{aligned}$$

Due to symmetry of the second derivatives the first terms of both equation are identical, leaving the equation

$$W_{,\mathcal{U}\mathcal{C}}(\sim)[\mathbf{Z}, \delta \mathbf{C}] = I_{,\mathcal{U}}(\sim)[\delta \mathbf{U}].$$

Inserting  $[S_{,\mathcal{C}}(\mathbf{C})^*[I_{,\mathcal{U}}(\sim)], \delta \mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} = W_{,\mathcal{U}\mathcal{C}}(\sim)[\mathbf{Z}, \delta \mathbf{C}]$  from (4.37) and  $\delta \mathbf{U} = S_{,\mathcal{C}}(\mathbf{C})[\delta \mathbf{C}]$  from (4.35) yields

$$[S_{,\mathcal{C}}(\mathbf{C})^*[I_{,\mathcal{U}}(\sim)], \delta \mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} = [I_{,\mathcal{U}}(\sim), S_{,\mathcal{C}}(\delta \mathbf{C})]_{\mathcal{U}^*, \mathcal{U}},$$

which is the definition of the adjoint operator.

The linear equation in (4.37) is also called *adjoint equation*. Having a computable representation of  $S'^*$ , we can continue with the derivation of the reduced derivative



in (4.34), that is

$$\begin{aligned} [I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}), \delta\mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} &= [I_{,\mathcal{U}}(\sim), S_{,\mathcal{C}}(\mathbf{C})[\delta\mathbf{C}]]_{\mathcal{U}^*, \mathcal{U}} + [I_{,\mathcal{C}}(\sim), \delta\mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} \\ &= [S_{,\mathcal{C}}(\mathbf{C})^* [I_{,\mathcal{U}}(\sim)], \delta\mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} + [I_{,\mathcal{C}}(\sim), \delta\mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} \\ &= [S_{,\mathcal{C}}(\mathbf{C})^* [I_{,\mathcal{U}}(\sim)] + I_{,\mathcal{C}}(\sim), \delta\mathbf{C}]_{\mathcal{C}^*, \mathcal{C}}. \end{aligned}$$

Finally we get the reduced derivative

$$I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}) = S_{,\mathcal{C}}(\mathbf{C})^* [I_{,\mathcal{U}}(S(\mathbf{C}), \mathbf{C})] + I_{,\mathcal{C}}(S(\mathbf{C}), \mathbf{C}) \in \mathcal{C}^*. \quad (4.38)$$

This implies the following steps in order to compute the reduced derivative:

- (1) apply the **solution operator**  $\mathbf{U} = S(\mathbf{C})$  from (4.31), that is solving the (nonlinear) state equation  $0 = W_{,\mathcal{U}}(\mathbf{U}, \mathbf{C})$  for a given  $\mathbf{C} \in \mathcal{C}$
- (2) solve the (linear) **adjoint equation** (4.37) to get the adjoint state  $\mathbf{Z} \in \mathcal{U}$
- (3) finalize the derivative  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}) = W_{,\mathcal{U}\mathcal{C}}(S(\mathbf{C}), \mathbf{C})[\mathbf{Z}, \cdot] + I_{,\mathcal{C}}(S(\mathbf{C}), \mathbf{C})$  from (4.38).

We are now able to compute the reduced objective functional  $I^{\text{red}}$  and its derivative  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C})$ . As we like to avoid a high number of function and gradient evaluations, we do not choose methods like steepest descent or nonlinear CG methods. We also try to avoid the very expensive computation of the second derivative of the reduced objective function. Hence a quasi-Newton method with an approximation for the second derivative could be a compromise.

#### 4.3.1. Quasi-Newton Method

We start by presenting a quasi-Newton method for the reduced objective functional  $I^{\text{red}}$  for the control space  $\mathcal{C}$ . Again, this *optimize-then-discretize*-approach will offer insight into the choice of scalar products as well as norms for the convergence criteria.

In a quasi-Newton method in the  $k$ -th iteration with the current control  $\mathbf{C}_k$ , the search direction  $\Delta\mathbf{C}_k$  is computed by minimization of a quadratic model  $M : \mathcal{C} \rightarrow \mathbb{R}$  at  $\mathbf{C}_k$ , e. g.

$$\min_{\Delta\mathbf{C}} M(\Delta\mathbf{C}) = I^{\text{red}}(\mathbf{C}_k) + I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)[\Delta\mathbf{C}] + [B_k \Delta\mathbf{C}, \Delta\mathbf{C}]_{\mathcal{C}^*, \mathcal{C}}, \quad (4.39)$$

with a self-adjoint and positive definite (coercive) linear operator  $B_k \in \mathcal{L}(\mathcal{C}, \mathcal{C}^*)$ , the control space  $\mathcal{C}$  and its dual space  $\mathcal{C}^*$ . The first order optimality condition is

$$\begin{aligned} B_k \Delta\mathbf{C}_k &= -I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k) \in \mathcal{C}^*, \\ \text{or} \quad \Delta\mathbf{C}_k &= -B_k^{-1} I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k) \in \mathcal{C}. \end{aligned} \quad (4.40)$$



The positive definiteness of  $B_k \in \mathcal{L}(\mathcal{C}, \mathcal{C}^*)$ , and thus  $B_k^{-1} \in \mathcal{L}(\mathcal{C}^*, \mathcal{C})$  as well, ensures that  $\Delta \mathbf{U}_k$  is a descent direction, that is for  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k) \neq 0$  we have

$$\begin{aligned} I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)[\Delta \mathbf{C}_k] &= \left[ I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k), \Delta \mathbf{C}_k \right]_{\mathcal{C}^*, \mathcal{C}} \\ &= - \left[ I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k), B_k^{-1} I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k) \right]_{\mathcal{C}^*, \mathcal{C}} \leq -c \|I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)\|_{\mathcal{C}^*}^2 < 0, \end{aligned}$$

where  $c$  is the constant from the coercivity inequality for  $B_k \in \mathcal{L}(\mathcal{C}, \mathcal{C}^*)$ . After computing the search direction  $\Delta \mathbf{C}_k$  we perform a line search to get a step length  $\alpha > 0$  for the update

$$\mathbf{C}_{k+1} = \mathbf{C}_k + \alpha \Delta \mathbf{C}_k.$$

If the operator  $B_k$  is chosen to be the Hessian  $I_{,\mathcal{C}\mathcal{C}}^{\text{red}}(\mathbf{C}_k)$ , one would get a standard Newton's method. However, since computing the exact Hessian is very expensive, we choose  $B_k$  to be an approximation by a BFGS-update formula. Here, we also have the possibility to directly use the inverse

$$H_k := B_k^{-1}, \quad (4.41)$$

$H_k : \mathcal{C}^* \rightarrow \mathcal{C}$ , which allows us to avoid solving the linear system. Before we formulate the BFGS-update for the control space  $\mathcal{C}$ , we introduce the standard version in  $\mathbb{R}^n$ .

#### 4.3.2. Broyden-Fletcher-Goldfarb-Shanno-Update

The idea for the *Broyden-Fletcher-Goldfarb-Shanno* update formula was independently discovered by Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970). Since then, the BFGS method became very popular and is used in a wide range of applications. For a derivation of the method, we refer to Nocedal and Wright (2006), Geiger and Kanzow (1999) and Kelley (1999). First we present the method in  $\mathbb{R}^n$  as it is used in most of the literature. We then translate it to the function space  $\mathcal{C}$  and will see that, if the spaces are chosen well, the algorithm after the discretization resembles the algorithm in  $\mathbb{R}^n$ .

**Example 4.3.7** (BFGS-update formula). We consider  $\mathbb{R}^n$  with the Euclidean scalar product and the free optimization problem

$$\min_{x \in \mathbb{R}^n} f(x),$$

and apply a quasi-Newton method with a BFGS-update formula for the matrix  $B_k \in \mathbb{R}^{n \times n}$ . As  $B_k$  will be symmetric positive definite (more details on this will follow), the search direction  $\Delta x_k$  can be computed by

$$B_k \Delta x_k = -\nabla f(x_k).$$

We avoid solving this system of linear equation with the inverse  $H_k := B_k^{-1}$ , that means

$$\Delta x_k = -H_k \nabla f(x_k),$$



by using the so-called *inverse BFGS*-update formula

$$\begin{aligned}
 H_{k+1} &= (I - \rho_k s_k y_k^\top) H_k (I - \rho_k y_k s_k^\top) + \rho_k s_k s_k^\top, \\
 \text{with } s_k &:= x_{k+1} - x_k \\
 y_k &:= \nabla f(x_{k+1}) - \nabla f(x_k) \\
 \rho_k &:= 1/(y_k^\top s_k).
 \end{aligned}$$

As mentioned before, the matrix  $B_k$  is required to be symmetric positive definite, thus  $H_k = B_k^{-1}$  as well. The symmetry follows directly from the BFGS-update formula, while the positive definiteness can be ensured by the condition

$$y_k^\top s_k > 0 \quad \Longleftrightarrow \quad \nabla f(x_{k+1})^\top \Delta x_k > f(x_k)^\top \Delta x_k,$$

for a proof, see (Nocedal and Wright, 2006, pg. 141). This curvature condition can be satisfied by a Wolfe-Powell line search. Still, this formula is not satisfactory for our problem since we have a large scale problem and the matrices  $H_k$  will be dense, meaning that the memory usage and the computational time for the matrix-vector multiplication will scale quadratically with the dimension of the discretized control space  $\mathcal{C}_h$ . Both effects being undesirable, we use a matrix-free version of the update formula to compute the product  $H_k \nabla f(x_k)$ , which uses the history of  $s_k$  and  $y_k$ . Since this history increases with the quasi-Newton iteration, we store only the last  $m$  of  $s_k$  and  $y_k$ , also called *limited-memory* BFGS (LM-BFGS). Algorithm 4.3.8 shows the matrix-free (recursive) and limited-memory version of the inverse BFGS-update formula, proposed by Nocedal (1980). If the input is the gradient  $\nabla f(x_k)$ , the output  $p$  is the search direction  $\Delta x_k$ .

**Algorithm 4.3.8** (Limited-Memory-BFGS for  $H_k v \in \mathbb{R}^n$ ).

**Input:**  $\{s_i\}_{i=k-m}^{k-1}$ ,  $\{y_i\}_{i=k-m}^{k-1}$ , initial  $H_0$ ,  $v \in \mathbb{R}^n$

**Output:**  $H_k v$

```

1: set  $q := v$ 
2: for  $i := k-1, k-2, \dots, k-m$  do
3:   set  $\alpha_i := \rho_i q^\top s_i$ 
4:   set  $q := q - \alpha_i y_i$ 
5: end for
6: set  $p := H_0 q$ 
7: for  $i := k-m, \dots, k-1$  do
8:   set  $\beta := \rho_i y_i^\top p$ 
9:   set  $p := p + (\alpha_i - \beta) s_i$ 
10: end for
11: return  $p$ 

```



Broyden et al. (1973) have shown that the quasi-Newton method with BFGS-update has local q-superlinear convergence under certain conditions. Even though the approximations  $H_k$  usually does not converge towards the inverse of the Hessian at the solution, the approximation in the direction of  $\Delta \mathbf{C}_k$  converges and is enough for the q-superlinear convergence. However, this proof does not apply for the limited-memory BFGS-update.

Algorithm 4.3.8 for the BFGS update has to be adapted for the quasi-Newton method in the Hilbert space  $\mathcal{C}$  instead of  $\mathbb{R}^n$ . Work on the formulation of the quasi-Newton method in Hilbert space was done by Turner and Huntley (1976), Turner and Huntley (1977), Kelley and Sachs (1989), Kelley et al. (1991) and references therein. However, they applied the quasi-Newton method to optimal control problems with ordinary differential equations and not PDEs like in our case.

The iterates  $x_k$  correspond to  $\mathbf{C}_k \in \mathcal{C}$  and the search directions  $\Delta x_k = p$  to  $\Delta \mathbf{C}_k \in \mathcal{C}$ . Hence  $\{s_i\}_{i=k-m}^{k-1}$  belongs to the control space  $\mathcal{C}$ , too. However, the vector  $v$  (the vector to which  $H_K$  is multiplied) and the vectors  $\{y_i\}_{i=k-m}^{k-1}$  can be chosen to belong either to the control space  $\mathcal{C}$  or its dual  $\mathcal{C}^*$ , yielding to slightly different versions of the LM-BFGS algorithm 4.3.8. Since we decided earlier to use derivatives rather than gradients, the  $\{y_i\}_{i=k-m}^{k-1}$  will be differences of the derivatives and the vector  $v$  corresponds to the derivative  $I_{\mathcal{C}}^{\text{red}}(\mathbf{C}_k)$ . We summarize these thoughts in the following.

**Definition 4.3.9** (BFGS-auxiliary functions  $s$  and  $y$ ). In a quasi-Newton method with a BFGS-update, we define  $s \in \mathcal{C}$  to be the change in the iterates  $\mathbf{C}_k \in \mathcal{C}$  and  $y \in \mathcal{C}^*$  to be the change in the derivatives  $I_{\mathcal{C}}^{\text{red}}(\mathbf{C}_k) \in \mathcal{C}^*$ , that is

$$\begin{aligned} s_k &:= \mathbf{C}_{k+1} - \mathbf{C}_k && \in \mathcal{C} \\ y_k &:= I_{\mathcal{C}}^{\text{red}}(\mathbf{C}_{k+1}) - I_{\mathcal{C}}^{\text{red}}(\mathbf{C}_k) && \in \mathcal{C}^* \\ \rho_k &:= 1 / [y_k, s_k]_{\mathcal{C}^*, \mathcal{C}}. \end{aligned} \tag{4.42}$$

Algorithm 4.3.10 shows the LM-BFGS-update for  $H_k : \mathcal{C}^* \rightarrow \mathcal{C}$ . First, we can see that the algorithm works in the dual control space  $\mathcal{C}^*$  in the first half, then it switches to the control space  $\mathcal{C}$  by the initial operator  $H_0 : \mathcal{C}^* \rightarrow \mathcal{C}$  and then stays in  $\mathcal{C}$ . Here it becomes clear, that  $H_0$  plays an essential role, more than the algorithm in  $\mathbb{R}^n$  would suggest<sup>2</sup>.

---

<sup>2</sup>Most authors, e.g. (Nocedal and Wright, 2006, pg. 178), Geiger and Kanzow (1999) and (Kelley, 1999, ch. 4.2), suggest computing an initial Hessian matrix, which could serve the purpose of the transition from  $\mathcal{C}^*$  to  $\mathcal{C}$  but is far too expensive (and dense!). Some also suggest to take a scaled identity matrix, but this choice could not ensure mesh independence of our quasi-Newton method.



**Algorithm 4.3.10** (Limited-Memory-BFGS for  $H_k v \in \mathcal{C}$ ).

**Input:**  $\{s_i\}_{i=k-m}^{k-1} \subset \mathcal{C}$ ,  $\{y_i\}_{i=k-m}^{k-1} \subset \mathcal{C}^*$ , initial  $H_0 : \mathcal{C}^* \rightarrow \mathcal{C}$ ,  $v \in \mathcal{C}^*$

**Output:**  $H_k v \in \mathcal{C}$

```

1: set  $q := v \in \mathcal{C}^*$ 
2: for  $i := k-1, k-2, \dots, k-m$  do
3:   set  $\alpha_i := \rho_i [q, s_i]_{\mathcal{C}^*, \mathcal{C}}$ 
4:   set  $q := q - \alpha_i y_i \in \mathcal{C}^*$ 
5: end for
6: set  $p := H_0 q \in \mathcal{C}$ 
7: for  $i := k-m, 1, \dots, k-1$  do
8:   set  $\beta := \rho_i [y_i, p]_{\mathcal{C}^*, \mathcal{C}}$ 
9:   set  $p := p + (\alpha_i - \beta) s_i \in \mathcal{C}$ 
10: end for
11: return  $p \in \mathcal{C}$ 

```

A simple idea for the initial  $H_0 : \mathcal{C}^* \rightarrow \mathcal{C}$  is a scaled Riesz operator for  $\mathcal{C}$ . For this choice,  $p := H_0 q$  means solving the linear system

$$s_{\mathcal{C}}(r, \delta \mathbf{C})_{\mathcal{C}} = [q, \delta \mathbf{C}]_{\mathcal{C}^*, \mathcal{C}} \quad \forall \delta \mathbf{C} \in \mathcal{C}. \quad (4.43)$$

If the discretized control space  $\mathcal{C}_h$  is build on  $\mathcal{P}_0$ -elements, solving the system (4.43) simply means solving with a diagonal mass matrix. The scaling  $s_{\mathcal{C}} > 0$  is given by the user and tries to compensate the magnitude of order of the actual Hessian  $I_{\mathcal{C}\mathcal{C}}^{\text{red}}(\mathbf{C}_0)$ . A guess could be the penalty parameter  $\gamma^{(\mathbf{C})}$  from (4.17).

#### 4.3.3. Simple Wolfe-Powell Line Search

In the  $k$ -th iteration of the quasi-Newton method, we have the current iterate  $\mathbf{C}_k \in \mathcal{C}$  and the search direction  $\Delta \mathbf{C}_k \in \mathcal{C}$ . The line search tries to determine a step length  $\alpha \geq 0$  such that the update

$$\mathbf{C}_{k+1} = \mathbf{C}_k + \alpha \Delta \mathbf{C}_k,$$

satisfies certain conditions, which are

- (1) having a sufficient decrease in the reduced objective functional  $I^{\text{red}}$  (otherwise the quasi-Newton might not converge) and
- (2) ensuring a curvature condition so that the operator  $H_k$  remains positive definite.

Both demands can be taken care of by the Wolfe-Powell conditions. For easier reading, we define the reduced objective functional  $\varphi : [0, \infty) \rightarrow \mathbb{R}$  along  $\Delta \mathbf{C}_k$ , that is

$$\begin{aligned} \varphi(\alpha) &:= I^{\text{red}}(\mathbf{C}_k + \alpha \Delta \mathbf{C}_k), \\ \varphi'(\alpha) &:= I_{\mathcal{C}}^{\text{red}}(\mathbf{C}_k + \alpha \Delta \mathbf{C}_k)[\Delta \mathbf{C}_k]. \end{aligned} \quad (4.44)$$



**Definition 4.3.11** (Wolfe-Powell Conditions). Let the function  $\varphi$  from (4.44) be given. The *Wolfe-Powell conditions* are

$$\begin{aligned}\varphi(\alpha) &\leq \varphi(0) + c_1 \alpha \varphi'(0) && \text{(sufficient decrease)} \\ \varphi'(\alpha) &\geq c_2 \varphi'(0) && \text{(curvature condition)},\end{aligned}\tag{4.45}$$

with constants  $0 < c_1 < c_2 < 1$ . The sufficient decrease condition is also known as the Armijo condition.

Figure 4.7 illustrates an example for the Wolfe-Powell conditions.

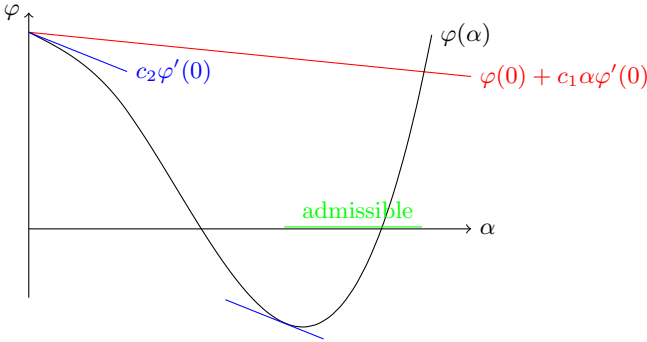


FIGURE 4.7. The Wolfe-Powell conditions from definition 4.3.11 with the parameters  $c_1 = 0.2$  and  $c_2 = 0.8$ . The reduced objective functional  $\varphi(\alpha)$  along the search direction is shown as the black curve. The Armijo condition is illustrated by the red line and the curvature condition by the blue slope. The green region is the set of admissible step lengths that satisfy both conditions, that is points of the curve are below the red line and have a steeper slope than the blue line.

There exists a step length  $\alpha$  which satisfies the Wolfe-Powell conditions with constants  $0 < c_1 < c_2 < 1$  given by the user and can be determined by algorithm 4.3.12, a bisection version based on (Geiger and Kanzow, 1999, Alg. 6.2) and (Nocedal and Wright, 2006, Alg. 3.5 and 3.6). For easier reading, we use the abbreviations  $\varphi(\alpha)$  and  $\varphi'(\alpha)$  from (4.44).

**Algorithm 4.3.12** (Simple Wolfe-Powell line search).

**Input:** obj. function  $\varphi : [0, \infty) \rightarrow \mathbb{R}$ ,  $0 < c_1 < c_2 < 1$ , expansion factor  $c_\alpha > 1$

**Output:** step length  $\alpha$  that satisfies the Wolfe-Powell cond. (4.45)



```

1: set  $\alpha := 1$ ,  $\text{done}_{(A)} := \text{FALSE}$ ,  $\text{done}_{(A)} := \text{FALSE}$ 
2: repeat
3:   if  $\varphi(\alpha) \geq \varphi(0) + c_1 \alpha \varphi'(0)$  then
4:     set  $a := 0$ ,  $b := \alpha$  and  $\text{done}_{(A)} := \text{TRUE}$ 
5:   else
6:     if  $\varphi'(\alpha) \geq c_2 \varphi(0)$  then
7:       set  $\text{done}_{(A)} := \text{TRUE}$  and  $\text{done}_{(A)} := \text{TRUE}$ 
8:     else
9:       set  $\alpha := c_\alpha \alpha$ 
10:    end if
11:  end if
12: until  $\text{done}_{(A)}$ 
13: while not  $\text{done}_{(B)}$  do
14:   Choose  $\alpha := \frac{1}{2}(a + b)$ 
15:   if  $\varphi(\alpha) \geq \varphi + c_1 \alpha \varphi'(0)$  then
16:     set  $b := \alpha$ 
17:   else
18:     if  $\varphi'(\alpha) \geq c_2 \varphi(0)$  then
19:       set  $\text{done}_{(B)} := \text{TRUE}$ 
20:     else
21:       set  $a := \alpha$ 
22:     end if
23:   end if
24: end while
25: return  $\alpha$ 

```

**Remark 4.3.13** (Simple Wolfe-Powell line search in algorithm 4.3.12). The first phase (A), line 2-12, aims to extend the initial search interval  $[0, 1]$  to  $[0, b]$  in order to guarantee that at least one admissible step length  $\alpha$  exists. If an admissible step length  $\alpha$  is found, the algorithm immediately stops and returns  $\alpha$ . The second phase (B), line 13-24, tries to find an admissible step length by shrinking the search interval subsequently while ensuring that the search interval retains an admissible step length.

**Remark 4.3.14** (Evaluating  $\varphi(\alpha)$  and  $\varphi'(\alpha)$ ). Computing  $\varphi(\alpha)$  for a given step length  $\alpha$  includes the update  $\mathbf{C}_{\text{temp}} = \mathbf{C}_k + \alpha \Delta \mathbf{C}_k$ , solving the forward problem  $\mathbf{U}_{\text{temp}} = S(\mathbf{C}_{\text{temp}})$  by the forward solver (algorithm 3.1.1), and finally evaluating  $I(\mathbf{U}_{\text{temp}}, \mathbf{C}_{\text{temp}})$ . As the derivative  $\varphi'(\alpha)$  always follows the evaluation of  $\varphi(\alpha)$ , the state  $\mathbf{U}_{\text{temp}}$  already matches the control  $\mathbf{C}_{\text{temp}}$  and we only need to solve the adjoint equation (4.37) and compute  $I_{,\mathbf{C}}^{\text{red}}(\mathbf{C}_{\text{temp}})$  by (4.38).



Finally, we summarize the function space quasi-Newton method presented in this section in algorithm (4.3.15). As a convergence criterion we choose a relative decrease of the norm of  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)$ , e. g.

$$\frac{\|I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)\|_{\mathcal{C}^*}}{\|I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_0)\|_{\mathcal{C}^*}} \leq \text{rTol} \quad \text{or} \quad \|I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)\|_{\mathcal{C}^*} \leq \text{aTol}, \quad (4.46)$$

for user defined relative tolerance  $\text{rTol} > 0$  and absolute tolerance  $\text{aTol} > 0$ . These do not have to be the same as in the CG or MINRES method.

**Algorithm 4.3.15** (Quasi-Newton method).

**Input:** red. obj. func.  $I^{\text{red}}$ , initial  $\mathbf{C}_0$ , various parameters needed in the sub routines

**Output:** solution  $\mathbf{C}$

- 1: set  $k := 0$
- 2: solve  $\mathbf{U}_0 := S(\mathbf{C}_0)$  with alg. 3.1.1
- 3: solve adjoint equation (4.37) and compute  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_0)$  from (4.38)
- 4: **while** ( $\|I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)\|_{\mathcal{C}^*}$  is too large) **do**
- 5:   set  $\Delta \mathbf{C}_k := -H_k I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)$  by the BFGS-update in alg. 4.3.10
- 6:   determine  $\alpha$  by the Wolfe-Powell line search in alg. 4.3.12
- 7:   set  $\mathbf{C}_{k+1} := \mathbf{C}_k + \alpha \mathbf{C}_k$
- 8:   store auxiliary functions  $s_k$  and  $y_k$  from (4.42)
- 9:   set  $k := k + 1$
- 10: **end while**

#### 4.3.4. Discretization

As described in section 3.4, we replace the infinite dimensional state space  $\mathcal{U}$  and control space  $\mathcal{C}$  by their discretizations  $\mathcal{U}_h$  respectively  $\mathcal{C}_h$ . All functions of  $\mathcal{U}_h$  and  $\mathcal{C}_h$  are represented by their coefficient vectors from  $\mathbb{R}^{N_{\mathcal{U}}}$  respectively  $\mathbb{R}^{N_{\mathcal{C}}}$ , where  $N_{\mathcal{U}} = \dim \mathcal{U}_h$  and  $N_{\mathcal{C}} = \dim \mathcal{C}_h$ .

##### The Forward Problem.

The numerical solver for the solution operator (4.31) directly follows the algorithm 3.1.1, for more details see section 3.4.

##### Adjoint Equation.

The adjoint equation (4.37) uses the same stiffness matrix as the Newton solver for the forward problem, because  $W_{,UU}(S(\mathbf{C}_k), \mathbf{C}_k)$  had to be computed during the algorithm 3.1.1. As  $W_{,UU}(S(\mathbf{C}_k), \mathbf{C}_k)$  is at least positive semi-definite, a CG solver can be used to compute the adjoint  $\mathbf{Z}$  which then is used to assemble the reduced derivative  $I_{,\mathcal{C}}^{\text{red}}(\mathbf{C}_k)$ .

##### BFGS-Update.



Since we decided to work with the derivatives instead of the gradients, we give a brief overview on the discretized dual control space  $\mathcal{C}_h^* := (\mathcal{C}_h)^*$ . The base  $\{\psi_j\}_{j=1}^{N_C}$  of  $\mathcal{C}_h^*$  is the dual of the base  $\{\varphi_i\}_{i=1}^{N_C}$  of  $\mathcal{C}_h$ , that means

$$\psi_j(\varphi_i) = \delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{else} \end{cases}.$$

In the following, we drop the range of the summation and only state the name of the index, e.g.  $\sum_i$  means  $\sum_{i=1}^{N_C}$ .

**Lemma 4.3.16** (Duality pairing and the Euclidean scalar product). Let  $p \in \mathcal{C}_h^*$  and  $\delta\mathbf{C} \in \mathcal{C}_h$  be discretized by

$$p = \sum_j p_j \psi_j, \quad \vec{p} = (p_j)_{j=1}^{N_C}, \quad \delta\mathbf{C} = \sum_i c_i \varphi_i, \quad \vec{c} = (c_i)_{i=1}^{N_C}.$$

Then the duality pairing can be evaluated by the Euclidean scalar product between the two coefficient vectors,

$$[p, \delta\mathbf{C}]_{\mathcal{C}_h^*, \mathcal{C}_h} = \vec{p}^\top \vec{c}.$$

*Proof.* A direct calculation using the linearity of the duality pairing gives

$$\begin{aligned} [p, \delta\mathbf{C}]_{\mathcal{C}_h^*, \mathcal{C}_h} &= \left[ \sum_j p_j \psi_j, \sum_i c_i \varphi_i \right]_{\mathcal{C}_h^*, \mathcal{C}_h} \\ &= \sum_j \left( p_j \sum_i c_i [\psi_j, \varphi_i]_{\mathcal{C}_h^*, \mathcal{C}_h} \right) = \sum_j \left( p_j \sum_i c_i \delta_{ij} \right) = \sum_j p_j c_j = \vec{p}^\top \vec{c}. \end{aligned}$$

Next, we illustrate the fact that automatic (or algorithmic) differentiation (AD) yields objects which belong to the discretized dual space  $\mathcal{C}_h^*$  and not the primal space  $\mathcal{C}_h$ . Let  $P : \mathcal{C} \rightarrow \mathbb{R}$  be a directional differentiable function. With the discretization of  $\mathcal{C}$ , the discretized function  $P_h : \mathbb{R}^{N_C} \rightarrow \mathbb{R}$  can be defined such that

$$P_h(\vec{c}) = P \left( \sum_i c_i \varphi_i \right) = P(\mathbf{C}).$$

The AD, see Griewank (2000) for an general introduction and Logg et al. (2012a) for details on FENICS, computes the partial derivatives of  $P_h$  with respect to the



coefficient vector  $\vec{c}$  belonging to  $\delta\mathbf{C}$ , that is

$$\begin{aligned} \frac{\partial P_h}{\partial c_j}(\vec{c}) &= \frac{\partial P_h}{\partial \vec{c}}(\vec{c})[\vec{e}_j] = \lim_{h \rightarrow 0} \frac{P_h(\vec{c} + h \vec{e}_j) - P_h(\vec{c})}{h} \\ &= \lim_{h \rightarrow 0} \frac{P(\sum_i c_i \varphi_i + h \varphi_j) - P(\sum_i c_i \varphi_i)}{h} \\ &= \lim_{h \rightarrow 0} \frac{P(\mathbf{C} + h \varphi_j) - P(\mathbf{C})}{h} = P_{,\mathbf{C}}(\mathbf{C})[\varphi_j]. \end{aligned}$$

One can already see that testing  $P_{,\mathbf{C}} \in \mathcal{C}^*$  with the base functions  $\varphi_j$  produces an object belonging to the dual space, like the right-hand side (3.21) in the forward problem. To make it more clear, we define the values of directional derivatives

$$p_j := P_{,\mathbf{C}}(\mathbf{C})[\varphi_j], \quad \vec{p} = (p_j)_{j=1}^{N_C},$$

and can decompose the derivative

$$P_{,\mathbf{C}}(\mathbf{C}) = \sum_j p_j \psi_j.$$

To see this, we insert  $p_j = P_{,\mathbf{C}}(\mathbf{C})[\varphi_j]$  into the above equation and test it with  $\varphi_i$ ,  $i = 1, \dots, N_C$ , i. e.

$$\left[ \sum_j P_{,\mathbf{C}}(\mathbf{C})[\varphi_j] \psi_j, \varphi_i \right]_{\mathcal{C}_h^*, \mathcal{C}_h} = \sum_j P_{,\mathbf{C}}(\mathbf{C})[\varphi_j] [\psi_j, \varphi_i]_{\mathcal{C}_h^*, \mathcal{C}_h} = P_{,\mathbf{C}}(\mathbf{C})[\varphi_i]$$

We conclude by stating that automatic differentiation computes coefficient vectors of the derivative which belong to the dual space  $\mathcal{C}^*$ . Hence, the duality pairing is evaluated by the Euclidean scalar product between the coefficient vectors  $\vec{y}$  for  $y \in \mathcal{C}_h^*$  and  $\vec{s}$  for  $s \in \mathcal{C}_h$ , that is

$$[y, s]_{\mathcal{C}_h^*, \mathcal{C}_h} = \vec{y}^\top \vec{s}. \quad (4.47)$$

This follows from the decision to work with the derivative rather than the gradient of the reduced objective functional  $I^{\text{red}}$ . Otherwise, the duality pairing would be replaced by the inner product in  $\mathcal{C}_h$ , which would mean multiplying with the mass matrix  $M_C$ , e. g.  $\vec{y}^\top M_C \vec{s}$ . Of course, the BFGS-update from algorithm 4.3.10 with a diagonal mass matrix  $M_C$  is not computationally expensive and the choice of the version has hardly an influence on the overall computational time. However, this might change if  $M_C$  is not diagonal. In this case of discontinuous Lagrange elements, solving with the mass matrix can still be done by a direct solver because  $M_C$  consists of several small blocks which can be solved separately. Otherwise, in the case of continuous elements, the mass matrix could be approximated by a diagonal matrix of its diagonal entries  $[M_C]_{ii}$ .

**Wolfe-Powell Line Search.**



The Wolfe-Powell line search (algorithm 4.3.12) assumes an exact evaluation of the reduced objective functional  $I^{\text{red}}$  and its derivative  $I_{\mathbf{C}}^{\text{red}}$ , otherwise the existence of admissible step lengths satisfying the Wolfe-Powell conditions (4.45) cannot be guaranteed and the line search might fail, see also (Kelley, 2003, pg. 14). For example, if the error in  $\varphi(0)$  is too large, there might be no step length satisfying the Armijo condition. The same holds true for the error in  $\varphi'(0)$  and the curvature condition, and especially if we seek a step length satisfying both conditions. Relaxing these conditions is difficult as we use the curvature condition to ensure the positive definiteness of  $H_k$ , which we might risk if the curvature condition does not hold.

Since the *exact* solution cannot be obtained by an iterative solver, we assume that we can overcome this problem by setting the tolerance in the forward solver (alg. 3.1.1) and the iterative solvers sufficiently small. Moreover, we force the forward solver and the linear solvers to do at least one iteration even though the absolute stopping criterion might already be met. This ensures that a new state  $\mathbf{U}$  and a new derivative  $I_{\mathbf{C}}^{\text{red}}$  is used for the evaluation of the objective function, otherwise the line search might fail. For example, let  $\mathbf{C}_k$  be very close to the solution, hence the search direction  $\Delta\mathbf{C}_k$  might be very small, that is  $\mathbf{C}_{k+1} \approx \mathbf{C}_k$ . This could cause the forward solver to assume that the state  $\mathbf{U}_k$  already solves the problem if the absolute stopping criterion is satisfied before the first iteration. The same applies to the linear solvers. Let us assume that the state is not updated, hence the value of quality functional remains  $Q(\mathbf{U}_{k+1}) = Q(\mathbf{U}_k)$ , and the change  $\Delta\mathbf{C}_k$  causes the penalty functional  $P$  to increase, e. g.  $P(\mathbf{C}_{k+1}) > P(\mathbf{C}_k)$ , then the line search is stuck as the Armijo-condition in (4.45) cannot be satisfied.



# 5 Implementation in FEniCS

## Contents

---

5.1	Introduction to FEniCS	105
5.2	Weak Formulations and UFL	109
5.2.1.	Mutual Definitions	110
5.2.2.	Objective Functions	113
5.2.3.	Forward Problem	115
5.2.4.	Lagrange Problem	118
5.2.5.	Reduced Optimal Control Problem	121
5.2.6.	Automatic Differentiation vs. Differentiation by Hand	124
5.2.7.	Parametrization	126
5.2.8.	Hierarchical Plate Model	131
5.3	Solver Routines	137
5.3.1.	Multigrid Method	137
5.3.2.	CG and MINRES Method	140
5.3.3.	Forward Problem	141
5.3.4.	Lagrange-Newton Problem	142
5.3.5.	Reduced Problem	144
5.4	An Experimental Preconditioner	147

---

This chapter gives an overview on the implementation of the presented algorithms into the finite element library FEniCS. The FEniCS Project is a collection of open-source software for the automation to solve mathematical problems based on differential equations, see Alnæs et al. (2009), Logg and Wells (2010), Logg (2007), Alnæs and Mardal (2010) and Kirby (2004). The most important aspect for us is the *Unified Form Language* (UFL, see Alnæs et al. (2013)) which allows to generate assembly routines by providing only the variational equation. This is an enormous saving of human resources which would be spent if one implements and tests the assembly routines for bilinear forms like  $W_{,UU}$  by hand. Another big advantage of the UFL is the possible use of automatic differentiation (AD) in order to avoid the implementation of long formulas which are associated with errors in calculation and implementation. Besides that, FEniCS offers a wide range of high-end routines that allow the user to solve simple problems quite easily in a few lines of code. However, we still have to implement the presented algorithms anew, mainly to their special



usage of the multigrid package FMG. The FMG package was developed by Ospald (2012) and offers the computational structures for a geometrical multigrid method in FEniCS, as well as high-end solver routines for linear problems.

In this work, we use the FEniCS version 1.2.0. A newer version 1.3.0 was released recently (January 2014), when the implementation of all presented algorithms in this work was already done. Also, the geometrical multigrid package FMG was only supported up to version 1.2.0. Therefore we did not port our implementation to the new version.

We will start with a simple example in section 5.1 to illustrate the work flow in FEniCS with the example of a problem in linear elasticity. First, the user has to provide the variational equation in a UFL file which will be compiled to generate the assembly routines. Then, the solver routine is called in a C++ code. We will give details on the UFL files for our problems in section 5.2 and the implementation of the solver routine in section 5.3. Of course, we will concentrate on the functionalities of the methods rather than stating them in detail.

The last section 5.4 is about an experimental preconditioner. So far, we said that the preconditioner for the forward problem is based on the linear model of elasticity. However, as the linear and nonlinear model differ for large deformations, we cannot expect that this preconditioner works well for very large deformations. Here we suggest a strategy how to use the current stiffness matrix from the nonlinear model as an experimental preconditioner with a precautional criterion if the current stiffness matrix turns out to be indefinite.

## 5.1 Introduction to FEniCS

We consider an example from linear elasticity to illustrate how it can be solved with the FEM-toolbox FEniCS.

**Example 5.1.1** (Linear Elasticity: Steel Bar). We consider a steel bar of dimension  $1\text{m} \times 10\text{cm} \times 10\text{cm}$  which is clamped at  $\mathbf{x}_1 = 0$ , i. e.  $\Gamma^D = \{0\} \times [0, 10] \times [0, 10]$ . A dead load  $\mathbf{f} \equiv (0, 0, -g\rho)^\top$  acts on the whole body  $\Omega$ , with the gravity acceleration  $g$  and the material density  $\rho$ . The material behavior is assumed to be described by the linear model of elasticity from (2.50). We seek the displacement  $\mathbf{U} \in \mathcal{U} = H_0^1(\Omega)^3$ .

First, let us recall the variational formulation (2.50) of the linear elasticity model. We seek to solve the linear equation

$$\int_{\Omega} \lambda \operatorname{tr}(\nabla \mathbf{U}) \operatorname{tr}(\nabla \mathbf{V}) + 2\mu(\varepsilon(\mathbf{U}) : \varepsilon(\mathbf{V})) \, d\mathbf{x} = \int_{\Omega} \mathbf{f}^\top \mathbf{V} \, d\mathbf{x}, \quad \forall \mathbf{V} \in \mathcal{U}, \quad (5.1)$$



where  $\lambda$  and  $\mu$  are the Lamé constants and the linearized strain is  $\varepsilon(\mathbf{U}) := \frac{1}{2}(\nabla \mathbf{U} + \nabla \mathbf{U}^\top)$ . This implies the bilinear form

$$a(\mathbf{U}, \mathbf{V}) := \int_{\Omega} \lambda \operatorname{div} \mathbf{U} \operatorname{div} \mathbf{V} + 2\mu \varepsilon(\mathbf{U}) : \varepsilon(\mathbf{V}) \, d\mathbf{x}, \quad (5.2)$$

and the linear form

$$b(\mathbf{V}) := \int_{\Omega} \mathbf{f}^\top \mathbf{V} \, d\mathbf{x}. \quad (5.3)$$

Both forms can be written in a *Unified Form Language* (UFL) file named `Elasticity.ufl`.

```

1 # define the finite element
2 elem_U = VectorElement("Lagrange", tetrahedron, dim=3, degree=1)
3
4 # define variables
5 U = TrialFunction(elem_U)
6 V = TestFunction(elem_U)
7
8 # define the load and the Lamé constants
9 f      = Constant(tetrahedron)
10 lambda = Constant(tetrahedron)
11 mu     = Constant(tetrahedron)
12
13 # define a function which return eps(u)
14 def eps(U):
15     return sym(grad(U))
16
17 # define the bilinear and linear forms
18 a = lambda*div(U)*div(V)*dx + 2*mu*inner( eps(U), eps(V) )*dx
19 L = inner(V, f)*dx
20
21 # name the forms which have to be assembled
22 forms = [ a, L ]

```

The UFL uses the syntax of Python, that means comments start with a `#`. Let us first explain the lines in the UFL file `Elasticity.ufl`.

- 2: The finite elements `elem_U` for the state space  $\mathcal{U} = H_0^1(\Omega)^3$  are vector-valued, continuous and piecewise linear Lagrange elements defined on tetrahedra.
- 5-6: The solution `U` and the test function `V` are defined on the finite element `elem`.
- 9-11: The volume load `f` and the Lamé constants  $\lambda$  and  $\mu$  are defined as `Constant` on the elements. The name `lambda` is already used in the UFL and therefore avoided.



- 14-15: We define a function for the operator  $\varepsilon(\cdot)$  for easier use. The gradient of a function can be accessed by the command `grad` which returns the Jacobi matrix of the vector field  $\mathbf{U}$ .
- 18-19: The variational formulations of the forms  $a$  and  $L$  can be directly written in the UFL file. The command `div` returns the divergence of a vector field and `inner(,)` is the inner product between tensors, e.g. vectors or matrices. The integration over the domain  $\Omega$  is called by `*dx`, while `*ds` is used to integrate over the boundary  $\partial\Omega$ .
- 22: Finally, we name the forms which need to be assembled. This allows the declaration of multiple forms in one UFL file and the use of auxiliary quantities which might not be assembled themselves.

The UFL file `Elasticity.ufl` is compiled by the *FEniCS Form Compiler* (FFC, see Logg et al. (2012b)) to yield the assembly routines and function space definitions in form of the header file `Elasticity.h` that can be included into a C++ program. The FFC has various features, very noticeable, for example, is the tensor optimization which cuts down the number of arithmetic operations by rearranging the formulas of the numerical integration.

FEniCS has already several routines included, so solving the equation (2.50) can be done in a few lines of C++ code, here named `main.cpp`.

```

1  #include <dolfin.h>
2  #include "Elasticity.h"
3
4  using namespace dolfin;
5
6  // Define Gamma^D where the body is clamped
7  class GammaD : public SubDomain
8  { bool inside(const Array<double>& x, bool on_boundary) const
9    { return on_boundary && ( x[0] < 0.0 + DOLFIN_EPS ); }
10 };
11
12 int main()
13 {
14     // Define the mesh and the function space on it
15     BoxMesh mesh(0,0,0, 5,1,1, 20,4,4);
16     Elasticity::FunctionSpace V_U(mesh);
17
18     // Define variational forms
19     Elasticity::BilinearForm a(V_U, V_U);
20     Elasticity::LinearForm L(V_U);
21     Function U(V);
22
23     // Set the material constants and the load

```



```

24 double E = 2.1e5;
25 double nu = 0.3;
26 Constant lmbda(nu * E / ((1+nu)*(1-2*nu)));
27 Constant mu(E / (2*(1+nu)));
28 Constant f(0, 0, -9.81*7.85e-6 );
29
30 a.lmbda = lmbda; a.mu = mu;
31 L.f = f;
32
33 // Assign clamping boundary condition
34 GammaD gammaD;
35 Constant U0(0, 0, 0);
36 DirichletBC bc(V, U0, left);
37
38 // Compute solution
39 solve(a == L, U, bc);
40
41 // save the solution
42 File file_U("U.pvd");
43 displacement_file << u;
44
45 return 0;
46 }

```

Again, we give some brief comments on the code.

- 1-2: The FEniCS packages as well as the assembly routines and function space definitions from the previously generated `Elasticity.h` are included.
- 7-10: The definition of the Dirichlet boundary  $\Gamma^D$  is done by a boolean function inside the class `Left`, where  $\mathbf{x}$  is the coordinate of a point for which is checked whether it belongs to  $\Gamma^D$  or not.
- 15-16: The triangulation  $\mathcal{T}_h$  is a structured mesh generated by the built-in routine `BoxMesh`, which generates an axis-aligned box defined by the two points  $(0, 0, 0)$  and  $(1000, 100, 100)$  (unit of length is mm). There are 20 divisions in  $\mathbf{x}_1$ - and 4 in  $\mathbf{x}_2$ - and  $\mathbf{x}_3$ -direction, yielding a mesh with  $21 \times 5 \times 5$  vertices. The function space  $\mathcal{U}_h = \mathbf{V}_h \times \mathbf{U}_h$  is defined by the finite elements declared earlier in `Elasticity.h`.
- 19-21: Bilinear and linear forms defined in `Elasticity.h` need the discretized space  $\mathcal{U}_h$  to be accessed. Inside these classes are the assembly routines for the stiffness matrix  $K_{\text{lin,elast}}$  and the right-hand side vector. The discretized solution  $\mathbf{U}_h = \mathbf{U}$  is defined as a function on  $\mathcal{U}_h$ .
- 24-28: We consider a ferrite steel with a Young's modulus of  $E = 2.1 \cdot 10^5 \frac{\text{N}}{\text{mm}^2}$  and a Poisson's ratio of  $\nu = 0.3$ . The volume load  $\mathbf{f}$  is the dead load of the body with a gravity acceleration  $g = 9.81 \frac{\text{m}}{\text{s}^2}$  and a density of  $\rho = 7.85 \cdot 10^{-6} \frac{\text{kg}}{\text{mm}^3}$ .



- 30-31: We set the Lamé constants  $\lambda$  and  $\mu$  in the bilinear form and the load  $\mathbf{f}$  in the linear form.
- 34-36: The boundary  $\Gamma^D$  is defined by the subdomain `gammaD` which is a member of the class `GammaD` from line 7-10. The homogeneous values  $\mathbf{U}^D \equiv \mathbf{0}$  are defined by the constant `U0`. Finally, the Dirichlet boundary conditions are collected in the object `bc`.
- 39: The `solve` function internally calls the constructor of a linear variational problem and a direct solver to compute the solution  $\mathbf{U}_h$ . This also includes the assembly of the stiffness matrix and the right-hand side.
- 42-43: The solution is saved in the pvd format (ParaView data object) which is a pointer to possibly multiple vtu files<sup>1</sup>.

The plot of the solution  $\mathbf{U}_h$  can be seen in figure 5.1.

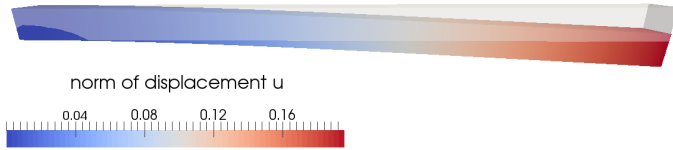


FIGURE 5.1. The numerical solution of the example 5.1.1. The transparent bar is the undeformed body. The displacement  $\mathbf{U}$  has been scaled by a factor 200 to visualize the deformation.

In this introduction, we used many high-end routines from FENICS which already have certain options that the user can adapt to the problem. However, we will see that the solver routines have to be modified or newly written to incorporate the algorithms with their ideas and techniques from the previous chapters.

## 5.2 Weak Formulations and UFL

In this section we give some details on the UFL files that are needed for the forward problem as well as the optimal control problem. The UFL file itself depends on some choices, namely

- which control/load  $\mathbf{C}$  is chosen (see table 2.1),
- which goal function is chosen (see section 4.1.1) and,
- whether a parametrization of the domain  $\Omega$  is used (see section 2.4),
- whether the hierarchical plate model is used or a full 3D model.

<sup>1</sup>vtu is an XML-based file format used by Visualization ToolKit (VTK), the u stands for unstructured mesh.



### 5.2.1. Mutual Definitions

Large parts of the code is mutual for all of the above cases and will be collected in a Python file named `common.py` which is imported in the other UFL files. We first start with the finite element definitions, as these are repeatedly used in all UFL files.

#### Finite Element Types

```

1 # 3D elements
2 cell3      = tetrahedron
3 elem3F0    = FiniteElement( "DG",          cell3, degree=0 )
4 elem3V0    = VectorElement( "DG",          cell3, degree=0, dim=3 )
5 elem3V1    = VectorElement( "Lagrange",    cell3, degree=1, dim=3 )

```

Elements defined by `FiniteElement` have one component while those with `VectorElement` have three components (see `dim`). The option "DG" marks discontinuous, while "Lagrange" is used for continuous elements. The "degree" refers to the polynomial order of the Lagrange ansatz function.

#### Energy Formulation

The material behavior is described by the energy density  $w$ , see section 2.2.2. We recall the density (2.18) and the material parameter choices (2.23) with respect to the Lamé constants  $\lambda$  and  $\mu$ , that is

$$w(F) = \underbrace{a}_{\frac{\mu}{2} - \frac{\lambda}{8}} \|F\|_F^2 + \underbrace{b}_{\frac{\lambda}{8}} \|\operatorname{cof} F\|_F^2 + \underbrace{c}_{\frac{\lambda}{8}} (\det F)^2 - \underbrace{d}_{\mu + \frac{\lambda}{2}} \ln(\det F) + \underbrace{e}_{-\frac{3\mu}{2} - \frac{\lambda}{8}}.$$

In the UFL file `common.py`, we write a function for the energy density that depends on the current deformation gradient  $F = I + \nabla \mathbf{U}$  and the Lamé constants  $\lambda$  and  $\mu$ . The backslash `\` is used to continue long statements over multiple lines in Python.

```

1 # energy density w
2 def EnergyDensity(F,mu,lmbda):
3     a = 0.5*mu-0.125*lmbda
4     b = 0.125*lmbda
5     c = 0.125*lmbda
6     d = 0.5*lmbda+mu
7     e = 1.5*mu+0.125*lmbda
8     return a*tr(F.T*F) + b*tr(cof(F.T*F)) \
9         + c*det(F)**2 - d*ln(det(F)) - e

```

**Remark 5.2.1** (Cofactor in FEniCS 1.2.0). The command `cofac` is implemented wrongly in FEniCS 1.2.0, as it computes  $\det FF^{-1}$  instead of  $\operatorname{cof} F := \det FF^{-\top}$ . We compensate this bug by a new function `cof` that calls `cofac(F.T)`.



```

1 # cofac in FEniCS 1.2.0 returns (det F)*F^-1 instead of (det F)*F
   ^-T
2 def cof(F):
3     return cofac(F.T)

```

We will later use the first derivative  $w_{,F}(F)[G]$  and second derivative  $w_{,FF}(F)[G, H]$  of the energy density  $w$ . There are two ways to gain them: automatic differentiation (AD) or differentiation by hand, see lemma A.2.1.

```

1 # 1st derivative of the energy density w in direction G
2 def EnergyDensity_1st(F,G,mu,lmbda):
3     a = 0.5*mu-0.125*lmbda
4     b = 0.125*lmbda
5     c = 0.125*lmbda
6     d = 0.5*lmbda+mu
7     W_F = (2*a + 2*b*tr(F.T*F))*F - 2*b*F*F.T*F \
8           + (2*c*det(F)-d/det(F))*cof(F)
9     return inner( W_F , G)

```

```

1 # 2nd derivative of the energy density w in directions G, H
2 def EnergyDensity_2nd(F,G,H,mu,lmbda):
3     a = 0.5*mu-0.125*lmbda
4     b = 0.125*lmbda
5     c = 0.125*lmbda
6     d = 0.5*lmbda+mu
7     W_FF = (2*a+2*b*inner(F,F))*G \
8            + 4*b*inner(F,G)*F \
9            - 2*b*(F*(F.T*G+G.T*F)+G*F.T*F) \
10            + 4*c*cof(F)*inner(cof(F),G) \
11            + (d/(det(F))**2-2*c )*cof(F)*G.T*cof(F)
12     return inner(W_FF,H)

```

We now do the same for the energy term of the loads. While the terms  $W^{(f)}$  for the volume load and  $W^{(g)}$  for the boundary load are very easy to implement in the UFL and do not require a new function, the terms  $W^{(t)}$  for the inner pressure and  $W^{(m)}$  for the fiber tension are part of `common.py`, as well as their first and second derivatives from the lemmas A.2.2 and A.2.3. The integration over the domain  $\Omega$  will be done in the UFL file for the problem itself. This applies to all forms that we write in `common.py`, in order to allow more flexibility with the integration. As both energy densities depend only on the deformation gradient, both expressions can be understood as densities similar to the energy density. The derivatives are then with respect to the deformation gradient  $F$ , thus the variables  $\mathbf{G}$  and  $\mathbf{H}$  are gradients, e.g.  $\mathbf{G} = \nabla \delta \mathbf{U}$  or  $\mathbf{H} = \nabla \Delta \mathbf{U}$ .



```

1 # inner pressure (turgor) and its derivatives
2 def Turgor(F):
3     return -det(F)
4 def Turgor_1st(F,G):
5     return -inner( cof(F), G )
6 def Turgor_2nd(F,G,H):
7     return -( inner(cof(F),G)*inner(cof(F),H) \
8               -inner(cof(F.T)*G,H.T*cof(F)) )/(det(F))

```

```

1 # fiber tension and its derivatives
2 def Fiber(F,a):
3     Fa = F*a
4     aFFa= inner(Fa,Fa)
5     return -0.5*ln(aFFa)
6 def Fiber_1st(F,G,a):
7     Fa = F*a
8     Ga = G*a
9     aFFa= inner(Fa,Fa)
10    return -inner( Fa, Ga )/aFFa
11 def Fiber_2nd(F,G,H,a):
12     Fa = F*a
13     Ga = G*a
14     Ha = H*a
15     aFFa= inner(Fa,Fa)
16    return -( inner(Ha,Ga) -2*inner(Fa,Ga)*inner(Fa,Ha)/aFFa )/aFFa

```

The next part is the inner product  $(\cdot, \cdot)_{\mathcal{U}}$  of the state space  $\mathcal{U}$ . It was defined in (2.28) by the linear model of elasticity. The stiffness matrix  $K_{\text{lin,elast}}$  will be assembled with the help of this function.

```

1 # the linearized strain eps(U)
2 def eps(U):
3     return sym(grad(U))
4
5 # the inner product of the state space U (from linear elasticity)
6 def LEscal(U,V,mu,lmbda):
7     return lmbda*div(U)*div(V) + 2*mu*inner( eps(U), eps(V) )

```

The line search in section 3.1 with the guiding criterion (3.4) requires the guiding function  $G$  from (3.5). We also include the offset  $\beta|\Omega|$  into the guiding function for an easier usage in the solver routines.

```

1 # define guiding function including the offset beta*vol(Omega)
2 def guiding(F,beta):
3     return beta + abs( ln(det(F)) )

```



### 5.2.2. Objective Functions

The penalty/cost terms  $P^{(C)}$  for the control  $C$  can be found in (4.17). As they are quite simple we do not write a new function but rather integrate these lines directly in each of the later UFL files for the forward or optimal control problems.

```

1 # penalty parameter
2 gamma_C = Coefficient(elem3F0) # (piecewise constant)
3
4 # penalty/cost terms P for the objective functions
5 P = 0.5*gamma_C*inner(C,C)*dx # if C is f,t or m
6 P = 0.5*gamma_C*inner(C,C)*ds # if C is g (pick only one line)

```

We give some examples for quality terms  $Q$  from section 4.1.1 and how to implement them into a UFL file.

**Example 5.2.2** (Standard Tracking Type). We extend the standard tracking type  $Q^{\text{track}}$  from (4.4) such that each spatial dimension is penalized separately, that is

$$Q^{\text{track}}(U) := \frac{1}{2} \int_{\Omega} \sum_{i=1}^3 \gamma_i(\mathbf{x}) (U_i(\mathbf{x}) - U_i^{\text{des}}(\mathbf{x}))^2 d\mathbf{x},$$

with a vector-valued penalty parameter  $\gamma : \Omega \rightarrow [0, \infty)^3$ .

```

1 # define penalty parameter and the desired state
2 gamma_U = Coefficient(elem3V0)
3 U_des = Coefficient(elemU)
4
5 # define the quality term Q
6 Q = ( 0.5*gamma[0]*(U[0]-Udes[0])**2 \
7       +0.5*gamma[1]*(U[1]-Udes[1])**2 \
8       +0.5*gamma[2]*(U[2]-Udes[2])**2 ) *dx
9 # and its first derivative I_U
10 I_U = ( gamma[0]*(U[0]-Udes[0])*(testU[0]) \
11         +gamma[1]*(U[1]-Udes[1])*(testU[1]) \
12         +gamma[2]*(U[2]-Udes[2])*(testU[2]) ) *dx

```

**Example 5.2.3** (Regional Penalization). The regional penalization  $Q^{\text{pen}}$  from (4.6) is the quality functional

$$Q^{\text{pen}}(U) = \int_{\Omega} q(\mathbf{x} + U(\mathbf{x})) \det(I + \nabla U(\mathbf{x})) d\mathbf{x},$$



with a continuous function  $q : \mathbb{R}^3 \rightarrow [0, \infty)$ , which can be implemented in the UFL file. The following function will be later used in section 6.2.2. There, we have a half plane

$$H := \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x}_1 - \mathbf{x}_3 - 2 \geq 0\},$$

which is described by the penalization function

$$q(\mathbf{x}) = [\mathbf{x}_1 - \mathbf{x}_3 - 2]_\varepsilon^+,$$

where  $[\cdot]_\varepsilon^+$  is the (smoothed) positive part

$$[x]_\varepsilon^+ := \frac{1}{2} \left( \sqrt{x^2 + \varepsilon^2} + x \right).$$

```

1  # the smoothed positive part function
2  def PositivePartSmoothed(a,eps):
3      return (sqrt(a**2+eps**2)+a)/2
4
5  eps      = Constant(cell3)
6  q        = PositivePartSmoothed( x[0]+U[0]-x[2]-U[2]-2,eps )

```

With the penalty  $q$  we can define the regional penalization  $Q^{\text{pen}}$  in the UFL file.

```

1  # define the position on the undeformed domain Omega
2  x_ref = cell3.x
3
4  # define the quality term Q
5  Q = det(F)*q(x_ref+U)*dx

```

The derivatives then can be gained by AD because it can also generate the derivatives of the function  $q$ . Alternatively, as shown in theorem 4.1.5, the derivative  $Q_{,U}$  can be implemented as an integral over the boundary.

```

1  # the first derivative of the quality term
2  I_U = gamma_U*q*inner(testU, cof(F)*n)*ds

```

Currently, it is not possible to define  $q$  outside the UFL file since  $q$  is evaluated at  $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{U}(\mathbf{x})$  and a **Coefficient** in FEniCS is always evaluated at an integration point  $\mathbf{x}_{\text{int}}$  and not at  $\mathbf{x}_{\text{int}} + \mathbf{U}(\mathbf{x}_{\text{int}})$ .

**Example 5.2.4** (Desired Direction). The simplified quality functional (4.12) for the desired direction  $\mathbf{s}^{\text{des}}$  is

$$Q^s(\mathbf{U}) = \int_{\Gamma^s} \frac{1}{2} \|F(\mathbf{x})\mathbf{s}(\mathbf{x}) - \mathbf{s}^{\text{des}}\|_2^2 \, dS,$$



and its first derivative

$$Q_{,U}^s(U)[\delta U] = \int_{\Gamma^s} \mathbf{s}(\mathbf{x})^\top \nabla \delta U(\mathbf{x}) (F(\mathbf{x})\mathbf{s}(\mathbf{x})\mathbf{s}^{\text{des}})_2^2 dS,$$

can be directly implemented in the UFL file. The reference direction  $\mathbf{s}$  is provided by the user.

```

1 # reference and desired direction
2 s      = Coefficient(elem3V0)
3 s_des  = Coefficient(elem3V0)
4 # the quality term Q and its first derivative I_U
5 Q      = 0.5*gamma_U*inner(F*s-s_des,F*s-s_des)*ds
6 I_U    = gamma_U*inner(grad(testU)*a,F*a-s_des)*ds

```

**Example 5.2.5** (Enclosed volume). The fourth quality functional was the volume between two surfaces. In the case of a plate, the formula has been simplified to the quality term (4.14), that is

$$\int_{\Gamma^V} \mathbf{x}_3[\mathbf{n}(\mathbf{x})]_3 dS = \int_{\Omega^{2D}} d d\mathbf{x}_1 d\mathbf{x}_2,$$

A representation of the first derivative of  $Q^V$  was shown in theorem 4.1.10, that is

$$Q_{,U}^V(U)[\delta U] := \int_{\Omega^{2D}} \delta U(\mathbf{x}_1, \mathbf{x}_2, -d)^\top \text{cof } F(\mathbf{x}_1, \mathbf{x}_2, -d)(0, 0, -1)^\top d\mathbf{x}_1 d\mathbf{x}_2.$$

```

1 # the subindex d refers to the lower surface
2 U_d    = Plate1(U,-1)
3 cofF_d = cof(Identity(3) + MyGrad(U,-1))
4 n_d    = as_vector( [0, 0, -1] )
5
6 # define the quality term Q
7 Q      = -gamma_U*(U_d[2]-d)*cofF_d[2,2]*dx + gamma_U*d*dx
8 # and its first derivative I_U
9 I_U    = -gamma_U*inner(Plate1(testU,-1),cofF_d*n_d)*dx

```

### 5.2.3. Forward Problem

We seek to solve the forward problem (2.46),

$$0 = W_{,U}(U_\star)[\delta U], \quad \forall \delta U \in \mathcal{U},$$



with algorithm 3.1.1, that is Newton's method with a line search strategy. Therefore, the UFL file `Elasticity.ufl` has to contain the bilinear and linear forms of the Newton system (3.2), the functions for a line search and the form for the preconditioner induced by the inner product  $(\cdot, \cdot)_{\mathcal{U}}$ . Let us start with the basic definitions in `Elasticity.ufl`.

```

1 from common import *
2
3 # define the state space, the test and the trial function
4 elemU = elem3V1
5 testU = TestFunction(elemU)
6 trialU = TrialFunction(elemU)
7
8 # current iterate U
9 U = Coefficient(elemU)
10
11 # Lamé constants
12 lmbda = Constant(cell3)
13 mu = Constant(cell3)
14
15 # deformation gradient F
16 F = Identity(3) + grad(U)

```

Let us briefly comment these lines.

- 1: We load the file `common.py` to have access to the mutual definitions.
- 4-6: Since the state space  $\mathcal{U}$  is an  $H^1$ -space, we discretize it with  $\mathcal{P}_1$ -elements, these are continuous, piecewise linear elements on tetrahedra. This applies to the test functions `testU` as well as the solution  $\Delta U_k = \text{trialU}$ .
- 9: Since we have a nonlinear problem where most forms depend on the current iterate  $U_k$  in the Newton's method, we define it as a `Coefficient`.

We will incorporate all four different loads, that are volume loads  $\mathbf{f}$ , boundary loads  $\mathbf{g}$ , inner pressures  $t$  and fiber tensions  $m$ . Since all control spaces are  $L_2$ -spaces, we discretize them with  $\mathcal{P}_0$ -elements, these are piecewise constant elements. Please note that  $\mathbf{f}$  and  $\mathbf{g}$  are vector fields, while  $t$  and  $m$  are scalar fields. The direction for the fibers are discretized like  $m$  with  $\mathcal{P}_0$ -elements.

```

1 # loads: f (volume), g (boundary), t (pressure), m (fiber tension)
2 f = Coefficient(elem3V0)
3 g = Coefficient(elem3V0)
4 t = Coefficient(elem3F0)
5 m = Coefficient(elem3F0)
6

```



```

7 # the direction a of the fibers
8 a = Coefficient(elem3V0)

```

We continue with the stored energy  $W$  from (2.45).

```

1 # stored energy W
2 W = EnergyDensity(F,mu,lmbda)*dx -inner(f, U)*dx -inner(g, U)*ds
   \
3                                     +t*Turgor(F)*dx +m*Fiber(F,a)*dx

```

FENICS offers automatic differentiation (AD) to gain the first  $W_{,U} = W_U$  and the second derivative  $W_{,UU} = W_{UU}$  of the stored energy  $W$ . The command `derivative` can be applied to the stored energy  $W$  and again on its first derivative  $W_{,U} = W_U$  to get the directional derivatives.

```

1 # use AD to get the derivatives W_U and W_UU
2 W_U = derivative( W, U, testU )
3 W_UU = derivative( W_U, U, trialU )

```

The alternative is a differentiation by hand. We already defined the first and second derivatives of the energy density  $w$  and certain energy terms which we can now use to define the derivatives of the stored energy  $W$ . A comparison between AD and differentiation by hand will be given in the section 5.2.6.

```

1 # define the derivatives W_U and W_UU by hand
2 W_U = EnergyDensity_1st(F,grad(testU),mu,lmbda)*dx \
3       -inner( f, testU )*dx -inner( g, testU )*ds \
4       +t*Turgor_1st(F,grad(testU))*dx \
5       +m* Fiber_1st(F,grad(testU),a)*dx
6 W_UU = EnergyDensity_2nd(F,grad(testU),grad(trialU),mu,lmbda)*dx
   \
7       +t*Turgor_2nd(F,grad(testU),grad(trialU))*dx\
8       +m* Fiber_2nd(F,grad(testU),grad(trialU),a)*dx

```

The remaining parts are the guiding function  $G$  from (3.5) as a possible line search and the preconditioner  $K_{\text{lin,elast}}$ . Although the preconditioner  $K_{\text{lin,elast}}$  is given by the bilinear form `a_prec`, we define a dummy right-hand side `L_prec`. This is just due to convenience so that we can use the built-in routine `SystemAssembler` in FENICS to assemble the stiffness matrix  $K_{\text{lin,elast}}$  and apply the Dirichlet boundary data symmetrically on it (see section 3.4.4). The routine `SystemAssembler` requires both, bilinear and linear forms, and assembles the matrix  $K_{\text{lin,elast}}$  as well as a dummy vector which is ignored afterwards. Of course, modifying the assembly routine is also possible but the additional computational work to assemble the dummy vector is negligible.



```

1 # guiding function
2 beta = Constant(tetrahedron)
3 G     = guiding(F,beta)*dx
4
5 # preconditioner: bilinear and (dummy) linear forms
6 a_prec = LEscal(testU,trialU,mu,lmbda)*dx
7 L_prec = testU[0]*dx

```

Finally we state the list of forms which are used later in the `main.cpp`. The UFL file is compiled with FFC to get the header file `Elasticity.h` which then is included in the `main.cpp` in section 5.3.3.

```

1 # name the forms that have to be assembled
2 forms = [ a_elast, L_elast, a_prec, L_prec, W, G]

```

#### 5.2.4. Lagrange Problem

In this section we give details on the UFL-file that we use in the all-at-once approach and the Lagrange-Newton algorithm 4.2.2. First let us recall the Lagrange function (4.18)

$$L(\mathbf{U}, \mathbf{C}, \mathbf{Z}) := I(\mathbf{U}, \mathbf{C}) + W_{\mathbf{U}}(\mathbf{U}, \mathbf{C})[\mathbf{Z}],$$

and the Lagrange equation (4.19)

$$L_{,X}(X^*)[\delta X] = 0 \quad \forall \delta X \in \mathcal{X},$$

with the abbreviations  $X := (\mathbf{U}, \mathbf{C}, \mathbf{Z}) \in \mathcal{X}$  and  $\mathcal{X} := \mathcal{U} \times \mathcal{C} \times \mathcal{U}$ . The Lagrange-Newton method computes the search direction  $\Delta X_k$  in the update  $X_{k+1} := X_k + \alpha \Delta X_k$  by solving the (linear) Lagrange-Newton system (4.22)

$$L_{,XX}(X_k)[\delta X, \Delta X_k] = -L_{,X}(X_k)[\delta X] \quad \forall \delta X \in \mathcal{X}.$$

Therefore we need the bilinear form  $L_{,XX}(X_k)$  and the linear form  $L_{,X}(X_k)$ , which is written in the UFL file `ElasticityLagrange.ufl`. We start with the definition of the finite elements.

```

1 from common import *
2 # define finite elements
3 elemU      = elem3V1
4 elemC      = elem3V0 # if C is f or g
5 elemC      = elem3F0 # if C is t or m (pick only one line)
6 element    = MixedElement([elemU, elemC, elemU])
7

```



```

8  # current iterate X and its split into state, control & adjoint
   state
9  X          = Coefficient(element)
10 U, C, Z    = split(X)
11
12 # test and trial function
13 trialX     = TrialFunction(element)
14 testX      = TestFunction(element)

```

We give some comments on these lines.

4-5: Depending on whether the control  $\mathbf{C}$  is vector-valued or scalar-valued, we choose the appropriate finite element.

6: FENICS has the feature to use mixed function spaces like  $\mathcal{X}_h$  which is defined by discretizing the state space  $\mathcal{U}_h$  and the control space  $\mathcal{C}_h$ .

9-10: A function defined on the mixed function space  $\mathcal{X}_h$  can be split into its subfunctions  $U \in \mathcal{U}_h$ ,  $C \in \mathcal{C}_h$ ,  $Z \in \mathcal{U}_h$ , which can then be used in the forms.

13-14: We define the test function  $\delta X = \text{testX}$  and trial function  $\Delta X_k = \text{trialX}$ .

The next part defines the Lamé constant and a possible fiber direction,

```

1  # Lamé constants
2  mu      = Constant(cell3)
3  lambda  = Constant(cell3)
4  # the direction a of the fibers
5  a       = Coefficient(elem3V0)

```

Parts of the code from section (5.2.2) have to be inserted for the objective functions.

```

1  # define the objective function
2  Q = ... # see section 5.2.2 on quality terms
3  P = ... # see section 5.2.2 on penalty terms
4  I = P + Q

```

Next, we define the variational equation where the multiplier  $\mathbf{Z}$  takes the role of the test function.

```

1  # deformation gradient F
2  F      = Identity(3) + grad(U)
3
4  # characteristic function to describe the control subdomain
5  char_C = Coefficient(elem3F0)
6
7  # the state equation tested with the multiplier Z
8  elast = EnergyDensity_1st(F, grad(Z), mu, lambda)*dx \
9         - char_C*inner(C, Z)*dx # if the control is a volume load f

```



```

10 -char_C*inner(C,Z)*ds # ... boundary load g
11 +char_C*C*Turgor_1st(U,Z)*dx # ... inner pressure t
12 +char_C*C* Fiber_1st(U,Z,a)*dx # ... fiber tension m

```

Remark 4.1.11 mentioned the possibility that the control might be restricted to  $\Omega^{\text{ctrl}} \subset \Omega$  respectively  $\Gamma^{\text{ctrl}} \subset \Gamma^N$ , which can be done by a characteristic function  $\chi^{\text{ctrl}} : \Omega \rightarrow \{0, 1\}$  respectively  $\chi^{\text{ctrl}} : \Gamma^N \rightarrow \{0, 1\}$  such that

$$\chi^{\text{ctrl}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega^{\text{ctrl}} \\ 0 & \text{else} \end{cases}, \quad \text{or} \quad \chi^{\text{ctrl}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Gamma^{\text{ctrl}} \\ 0 & \text{else} \end{cases}. \quad (5.4)$$

This characteristic function  $\chi^{\text{ctrl}}$  will be integrated in lines 9-12 from the previous code fragment into the energy terms  $W^{(C)}$  to restrict the integral in order to the control subdomain  $\Omega^{\text{ctrl}} \subset \Omega$  respectively  $\Gamma^{\text{ctrl}} \subset \Gamma^N$ .

Now we can build the Lagrange function (4.18) and its first and second derivative with the help of AD.

```

1 # Lagrange function and its 1st and 2nd derivatives
2 L      = elast + I
3 L_X    = derivative(L, u, testX)
4 L_XX   = derivative(L_X, u, trialX)

```

The guiding criterion as a line search method requires the guiding function  $G$ .

```

1 # guiding function
2 beta = Constant(cell13)
3 G     = guiding(F,beta)*dx

```

The preconditioner (4.27) consists of the stiffness matrix  $K_{\text{lin,elast}}$  and the mass matrix  $M_C$ , which are defined by bilinear forms `a_prec.U` and `a_prec.C`. Again, we use a dummy linear form `L_prec.U` for the convenient use of the built-in routine `SystemAssembler` for  $K_{\text{lin,elast}}$ . This is not necessary for the mass matrix  $M_C$  because we have no Dirichlet boundary conditions that need to be applied here. The scaling  $s_U, s_C, s_Z$  of the blocks of the preconditioner is done in the solver routine.

```

1 # block preconditioner
2 trialU = TrialFunction(elemU)
3 testU  = TestFunction(elemU)
4 trialC = TrialFunction(elemC)
5 testC  = TestFunction(elemC)
6
7 # bilinear form for the stiffness matrix K_linelast, and the
  dummy rhs
8 a_prec_U = L_Escal(testU,trialU,mu,lmbda)*dx
9 L_prec_U = teU[0]*dx

```



```

10
11 # bilinear form for the mass matrix M_C
12 a_prec_C = inner(testC,trialC)*dx # if C is f, t or m
13 a_prec_C = inner(testC,trialC)*ds \      #(pick only one line)
14         + 0.00001*inner(teC,trC)*dx # if C is g

```

**Remark 5.2.6** (Perturbation of the  $L_2(\Gamma)$  inner product). Line 14 concerns the boundary load  $\mathbf{g}$  which only acts on the boundary  $\Gamma^N$ . Even though FENICS allows the definition of boundary functions which only live on the boundary  $\partial\Omega$ , the current UFL specification does not allow boundary and standard functions in a single form. Therefore, even though  $\mathbf{g}$  is only defined on  $\Gamma^N$ , we implement it as a function discretized on  $\Omega$ . The degrees of freedom in the interior of  $\Omega$  do not show up in the energy term  $W^{(\mathbf{g})}$  (and the forward Newton system) or the optimal control problem itself. Hence these values remain zero if we use an iterative Krylov method like CG or MINRES in Newton's method. Even the Lagrange-Newton system remains solvable, it is singular. Hence our Krylov subspace method MINRES still works fine. But we need to take care of the mass matrix  $M_C$  as we use a direct solver in the block preconditioner. Either

- we modify the solving routine to ignore those inner nodes, or
- we use a projection into the boundary first, solve with the mass matrix  $M_C$  on the boundary and set the inner nodes zero, or
- we perturb the mass matrix  $M_C$  by adding the standard inner product of  $L_2(\Omega)$ .

We choose the last option since it can be easily implemented. However, one might have to modify the factor in front of the  $L_2(\Omega)$  inner product since both quantities have a different order of magnitude, depending on the triangulation  $\mathcal{T}_h$ .

We finalize the UFL file `ElasticityLagrange.ufl` by stating the list of required forms.

```

1 # name the forms that have to be assembled
2 forms = [ L_X, L_XX, G, I, a_prec_U, L_prec_U, a_prec_C ]

```

### 5.2.5. Reduced Optimal Control Problem

In order to solve the reduced optimal control problem (4.32) numerically, we need the finite elements for the state space  $\mathcal{U}_h$  and control space  $\mathcal{C}_h$  and assembly routines for

- the solution operator  $\mathbf{U} = S(\mathbf{C})$  from (4.31),
- the adjoint equation (4.37),
- the derivative  $I_{\mathcal{C}}^{\text{red}}$  from (4.38),



- the stiffness matrix  $K_{\text{lin.elast}}$  from (2.28) for the preconditioner and the inner product  $(\cdot, \cdot)_C$  from table 2.2 for the mass matrix  $M_C$  which is used in the BFGS-update and the norms  $\|\cdot\|_C$  and  $\|\cdot\|_{C^*}$ .

These will be obtained by the UFL file `ElasticityReduced.ufl` which starts with basic definitions of the finite elements.

```

1 from common import *
2
3 # define the state space (continuous, piecewise linear)
4 elemU = elem3V1
5 testU = TestFunction(elemU)
6 trialU = TrialFunction(elemU)
7
8 # define the control space (piecewise constant)
9 elemC = elem3V0 # if C is f or g
10 elemC = elem3F0 # if C is t or m (pick only one line)
11 testC = TestFunction(elemC)
12 trialC = TrialFunction(elemC)
13
14 # current iterates U, C and Z
15 U = Coefficient(elemU) # state
16 C = Coefficient(elemC) # control
17 Z = Coefficient(elemU) # adjoint

```

Next the Lamé constants  $\mu, \lambda$  and possible fiber directions  $\mathbf{a}$  are defined.

```

1 # Lamé constants
2 mu = Constant(cell3)
3 lambda = Constant(cell3)
4
5 # the direction a of the fibers
6 a = Coefficient(elem3V0)

```

The code for objective function can be taken from section (5.2.2). Instead of using AD, one can also write the differentiation by hand.

```

1 # define the objective function
2 Q = ... # see section 5.2.2 on quality terms
3 P = ... # see section 5.2.2 on penalty terms
4 I = P + Q
5 I_U = derivative(I, U, testU)
6 I_C = derivative(I, C, testU)

```

The forward problem is analogous to the part in section 5.2.3, the only difference is the name  $C$  of the control and the characteristic function  $\text{char\_C} = \chi^{\text{ctrl}}$ .



```

1  # deformation gradient F
2  F      = Identity(3) + grad(U)
3
4  # characteristic function to describe the control subdomain
5  char_C = Coefficient(elem3F0)
6
7  # stored energy W (as possible line search)
8  W      = EnergyDensity(F,mu,lmbda)*dx \
9          -char_C*inner(C, U)*dx      # if the control is a volume load
10         f
11         -char_C*inner(C, U)*ds      # ... boundary load
12         g
13         +char_C*C*Turgor(F)*dx      # ... inner pressure
14         t
15         +char_C*C* Fiber(F,a)*dx    # ... fiber tension
16         m
17
18 # define the derivatives W_U and W_UU by hand
19 W_U = EnergyDensity_1st(F,grad(testU),mu,lmbda)*dx \
20     -char_C*inner( C, testU )*dx      # if C =
21     f
22     -char_C*inner( C, testU )*ds      # if C =
23     g
24     +char_C*C*Turgor_1st(F,grad(testU))*dx      # if C =
25     t
26     +char_C*C* Fiber_1st(F,grad(testU),a)*dx    # if C =
27     m
28 W_UU = EnergyDensity_2nd(F,grad(testU),grad(trialU),mu,lmbda)*dx
29     \
30     +char_C*C*Turgor_2nd(F,grad(testU),grad(trialU))*dx #if C
31     =t
32     +char_C*C* Fiber_2nd(F,grad(testU),grad(trialU),a)*dx #if C
33     =m

```

The preconditioner  $K_{\text{lin.elast}}$  for the forward problem and adjoint problem and the mass matrix  $M_C$  is implemented the same way as in section 5.2.4.

```

1  # bilinear form for the stiffness matrix K_linelast, and the
   dummy rhs
2  a_prec_U = LEscal(testU,trialU,mu,lmbda)*dx
3  L_prec_U = testU[0]*dx
4
5  # bilinear form for the mass matrix M_C
6  a_prec_C = inner(testC,trialC)*dx # if C is f, t or m

```



```

7 a_prec_C = inner(testC,trialC)*ds + 0.00001*inner(teC,trC)*dx #
    if g

```

After computing the adjoint state  $\mathbf{Z}$ , we can build the derivative  $I_{,C}^{\text{red}}$  of the reduced objective function  $I^{\text{red}}$  by (4.38).

```

1 # first derivative of the reduced objective function I
2 I_1st = char_C*inner(testC,Z)*dx + I_C          # if C = f
3 I_1st = char_C*inner(testC,Z)*ds + I_C          # if C = g
4 I_1st = -char_C*testC*Turgor_1st(F,grad(Z))*dx + I_C # if C = t
5 I_1st = -char_C*testC* Fiber_1st(F,grad(Z),a)*dx + I_C # if C = m

```

An alternative line search method to the Armijo backtracking is the guiding criterion with the guiding function  $G$ .

```

1 # guiding function
2 beta = Constant(cell13)
3 G     = guiding(F,beta)*dx

```

Finally, we list all forms that we later need.

```

1 forms = [W_UU, W_U, G, W, I, I_U, a_prec_U, L_prec_U, a_prec_C,
    I_1st]

```

### 5.2.6. Automatic Differentiation vs. Differentiation by Hand

In this section we compare the AD and the differentiation by hand in FEniCS. While the AD is very easy for the user in contrast to a differentiation by hand, it might not be as fast as an analytically given derivative. We consider the following benchmark to compare the two approaches.

Let the domain  $\Omega$  be a unit cube  $[0, 1]^3$ . We assemble the stiffness matrix  $A$  from (3.20) for the nonlinear problem by three different ways:

- (1) Full-AD: Differentiate the stored energy twice

```

# use AD to get the derivative W_UU
W    = EnergyDensity(F,mu,lmbda)*dx \
      + t*Turgor_1st(F,grad(testU))*dx \
      + m*Fiber_1st(F,grad(testU),a)*dx
W_U1= derivative( W,    U, testU )
A_1 = derivative( W_U1, U, trialU )

```

- (2) AD on the analytical representation of the first derivative  $w_{,F}$



```
# define the derivatives W_U by hand and gain W_UU by AD
W_U = EnergyDensity_1st(F,grad(testU),mu,lmbda)*dx \
      +t*Turgor_1st(F,grad(testU))*dx \
      + m*Fiber_1st(F,grad(testU),a)*dx
A_2 = derivative( W_U, U, trialU )
```

(3) Analytical representation

```
# define the derivatives W_UU by hand
A_3=EnergyDensity_2nd(F,grad(testU),grad(trialU),mu,lmbda)
    *dx\
    +t*Turgor_2nd(F,grad(testU),grad(trialU))*dx \
    + m*Fiber_2nd(F,grad(testU),grad(trialU),a)*dx
```

Table 5.2.6 shows the results for a mesh with  $51 \times 51 \times 51$  vertices. The time was measured for the assembly of the matrix associated with the second derivative of each of the single energy terms  $W^{(T)}$ ,  $W^{(t)}$ ,  $W^{(m)}$  from the table 2.1 individually and the total stored energy  $W = W^{(T)} + W^{(t)} + W^{(m)}$ . The FFC options to compile the UFL file were

```
-f split -0 -f eliminate_zeros -f precompute_basis_const
```

time in sec. for	$W_{,UU}^{(T)}$	$W_{,UU}^{(t)}$	$W_{,UU}^{(m)}$	$W_{,UU}$	w/o -0
(1) full-AD	41	16	16	45	40
(2) AD on $W_{,U}$	21	16	16	26	58
(3) analytical	20	16	16	26	45

TABLE 5.1. Computational times to assemble the stiffness matrix by (1) Full-AD, (2) AD on the analytical first derivative, (3) analytical representation. The last column states the time for  $W_{,UU}$  if the option -0 is disabled.

Besides that, we checked the correctness of the formulas by comparing the relative errors between the matrices in the Frobenius norm, which all were of the order of the machine accuracy. We draw the following conclusions from these experiments:

- For terms like  $W^{(t)}$  or  $W^{(m)}$  using AD is a practical method to get first and second derivatives without having to deal with differentiation by hand.



This includes derivatives not only of simple terms like  $F : F = \|F\|_F^2$  but also of terms like  $\det F$ .

- For more “difficult” terms like  $\|\text{cof } F\|_F^2$ , applying AD twice still gets the right matrix though the computational time might be larger than applying AD on the first analytic derivative or an analytic representation of the second derivative.
- The assembly time for the analytical representation depends on the UFL code. It seems to be a good idea to avoid matrix inner products and multiple evaluation of identical terms by grouping as much as possible.
- The built-in optimizer of FFC, called by the `-O` option, reduces the total number of arithmetic operation drastically (about the factor 100 to 400 for the above example). However, a big part of the assembly time, around 13 to 14 seconds in the above example, seems to be used for basic operations like computing the Jacobi matrix of the transformation from reference to world element or inserting the local stiffness matrices to the global stiffness matrix. This can also be observed for dummy forms which have virtually almost no computational effort, for example

```
A_4=testU[0]*trial[0]*dx
```

### 5.2.7. Parametrization

The usage of a parametrization  $\mathbf{x} = \mathbf{x}(\tilde{\mathbf{x}})$  was introduced in section 2.4. First, in order to implement the parametrized model into FEniCS, we have to choose how we want to handle the parametrization. The most important quantities for the formulation of the parametrized weak formulation 2.58 are the local volume  $\det \tilde{G}$  and the inverse Jacobi matrix  $\tilde{G}^{-1}$ , but the question is what the user provides and what he leaves to FEniCS to calculate. We will illustrate this with the cylindrical parametrization from example 2.4.1, that is

$$\mathbf{x}(\tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, \tilde{\mathbf{x}}^3) = \begin{pmatrix} \tilde{\mathbf{x}}^1 \cos \tilde{\mathbf{x}}^2 \\ \tilde{\mathbf{x}}^1 \sin \tilde{\mathbf{x}}^2 \\ \tilde{\mathbf{x}}^3 \end{pmatrix},$$

with the Jacobi matrix  $\tilde{G}$  and its inverse  $\tilde{G}^{-1}$ ,

$$\tilde{G}(\tilde{\mathbf{x}}) = \begin{pmatrix} \cos \tilde{\mathbf{x}}^2 & -\tilde{\mathbf{x}}^1 \sin \tilde{\mathbf{x}}^2 & 0 \\ \sin \tilde{\mathbf{x}}^2 & \tilde{\mathbf{x}}^1 \cos \tilde{\mathbf{x}}^2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \tilde{G}^{-1}(\tilde{\mathbf{x}}) = \begin{pmatrix} \cos \tilde{\mathbf{x}}^2 & \sin \tilde{\mathbf{x}}^2 & 0 \\ -\frac{1}{\tilde{\mathbf{x}}^1} \sin \tilde{\mathbf{x}}^2 & \frac{1}{\tilde{\mathbf{x}}^1} \cos \tilde{\mathbf{x}}^2 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

and its inverse  $\tilde{G}^{-1}$  and the local volume  $\det \tilde{G}(\tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, \tilde{\mathbf{x}}^3) = \tilde{\mathbf{x}}^1$ .



**Example 5.2.7** (Passing  $\det \tilde{G}$  and  $\tilde{G}^{-1}$  to the UFL). The user provides both, the local volume  $\det \tilde{G}$  and the inverse Jacobi matrix  $\tilde{G}^{-1}$ , directly to FENICS.

```

1  # define the cylindrical parametrization
2  elem3T0 = TensorElement( "DG", cell3, degree=0, shape =
      (3,3) )
3  detG = Coefficient(elem3F0)
4  invG = Coefficient(elem3T0)

```

This requires the definition of  $\det \tilde{G}$  and  $\tilde{G}^{-1}$  in the C++ code.

```

1  // define the determinant of the Jacobi matrix G
2  class Cylinder_detG : public Expression
3  { public: Cylinder_detG() : Expression() {}
4    void eval(Array<double>& det, const Array<double>& x_p)
      const
5    { det[0] = x_p[0]; }
6  };
7
8  // define the inverse of the Jacobi matrix G
9  class Cylinder_invG : public Expression
10 { public: Cylinder_invG() : Expression(3,3) {}
11   void eval(Array<double>& g, const Array<double>& x_p)
      const
12   { g[0] = cos(x_p[1]); g[1] = sin(x_p[1]); g[2] = 0.0;
13     g[3] = -sin(x_p[1])/x_p[0]; g[4] = cos(x_p[1])/x_p[0];
14     g[5] = 0.0;
15     g[6] = 0.0; g[7] = 0.0; g[8] = 1.0; // | 0 1 2 |
16     // the ordering in the matrix is invG = | 3 4 5 |
17     //                                     | 6 7 8 |
18   }
19 };

```

**Example 5.2.8** (Passing  $\tilde{G}$  to the UFL). The user has to provide the Jacobi matrix  $\tilde{G}$  and passes it as a `Coefficient` to FENICS.

```

1  elem3T0 = TensorElement( "DG", cell3, degree=0, shape =
      (3,3) )
2  # define the cylindrical parametrization by its Jacobi
      matrix
3  G_p = Coefficient(elem3T0)
4  detG = det(G_p)
5  invG = inv(G_p)

```



Here we have to provide the Jacobi matrix  $\tilde{G}$  in the C++ code.

```

1  // define the Jacobi matrix G
2  class Cylinder_G : public Expression
3  { public: Cylinder_G() : Expression(3,3) {}
4    void eval(Array<double>& g, const Array<double>& x_p)
5      const
6      { g[0] = cos(x_p[1]); g[1] = -x_p[0]*sin(x_p[1]); g[2] =
7        0.0;
8        g[3] = sin(x_p[1]); g[4] = x_p[0]*cos(x_p[1]); g[5] =
9        0.0;
10       g[6] = 0.0;          g[7] = 0.0;          g[8] =
11         1.0;
12     }
13 };

```

**Example 5.2.9** (Full Implementation in the UFL). In this approach the parametrization  $\mathbf{x}(\tilde{\mathbf{x}})$  is coded into the UFL file and all calculations are done by FEniCS. A change of the parametrization then requires a recompilation of the UFL file with the FFC.

```

1  # define the cylindrical parametrization
2  def Cylindrical(x_p):
3      x1 = x_p[0]*cos(x_p[1])
4      x2 = x_p[0]*sin(x_p[1])
5      x3 = x_p[2]
6      return as_vector((x1,x2,x3))
7
8  # get the parameter from the FE-mesh
9  x_p = cell13.x
10 # define the global parameter
11 x = Cylindrical(x_p)
12 # define the Jacobi matrix G_p, its inverse and determinant
13 G_p = grad(x)
14 detG = det(G_p)
15 invG = inv(G_p)

```

It is not possible to just pass the parametrization  $\mathbf{x}(\tilde{\mathbf{x}})$  to FEniCS as a **Coefficient** because it cannot use AD to generate the Jacobi matrix  $\tilde{G}$  from a **Coefficient** in the C++ code (like the code for  $\tilde{G}$ ). Objects of the class **Expression** in FEniCS are a black-box that can only be evaluated.

### Comparison of the Three Methods



The major difference between the methods from examples 5.2.7 - 5.2.9 might be the assembly time of the matrices and vectors, mainly of the stiffness matrix. We use the following example for comparison.

**Example 5.2.10** (Half tube). Let  $\Omega$  be a half tube with inner radius  $r_1 = 0.5$ , outer radius  $r_2 = 1$  around the  $\mathbf{x}_3$ -axis and height  $h = 2$ , see figure 5.2. The body is clamped at  $\mathbf{x}_3 = 0$  and a volume force  $\mathbf{f} = (0, \frac{1}{100}, 0)^\top$  acts on the whole body.

We solve the problem on a coarse mesh with  $3 \times 13 \times 9$  vertices and  $L = 3$  mesh refinements. We do not use nested iterations as we are primarily interested in the differences of the assembly times.

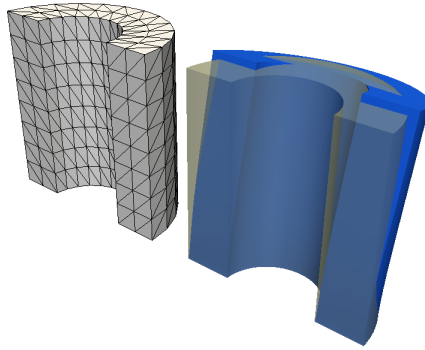


FIGURE 5.2. The left mesh is the coarse mesh with  $3 \times 13 \times 9$  vertices for the parameters  $(\tilde{b}x_1, \tilde{b}x_2, \tilde{b}x_3)$ . The solution is shown in the right blue mesh.

**Remark 5.2.11** (Comparing parametrized and unparametrized formulations of the forward model). At first glance, it seems easy to compare this model with an unparametrized model, which can be obtained if we move the parameter mesh  $\tilde{\Omega}$  to the reference configuration  $\Omega$  first and then solve the problem on  $\Omega$ . But a closer look at the finite elements and the implementation reveals two problems.

The first problem commonly occurs while refining meshes whose boundaries are originally from curved surfaces. Let  $\mathbf{x}(\tilde{\mathbf{x}})$  be curvilinear coordinates to describe the domain  $\Omega = \mathbf{x}(\tilde{\Omega})$ . Let us consider the case where the parameter space  $\tilde{\Omega}$  is a polyhedron and the boundary of the domain  $\Omega$  is curved in some parts. We discretize the parameter space  $\tilde{\Omega}$  into the mesh  $\tilde{\mathcal{T}}_0$  and move its nodes  $\tilde{\mathbf{x}}_i^{\text{glob}}$  by the parametrization to the new coordinates  $\mathbf{x}(\tilde{\mathbf{x}}_i^{\text{glob}})$  and obtain the mesh  $\mathcal{T}_0$ .



time in sec. for	assembly	solving	total
example 5.2.7	83	14	107
example 5.2.8	83	14	107
example 5.2.9	69	14	93
unparametrized	61	15	86

TABLE 5.2. Computational times (in seconds) to solve the deformation problem from example 5.2.10. The unparametrized model is based on a prior mesh move, see remark 5.2.11 for more details.

When we refine the coarse meshes  $\tilde{\mathcal{T}}_0$  or  $\mathcal{T}_0$ , the new elements are generated by a uniform refinement of triangles respectively tetrahedra. This causes no problems for a polyhedral parameter space  $\tilde{\Omega}$ , but if the boundary of the domain  $\Omega$  is curved, the refinement algorithm does not have the information of the curved boundary, so new nodes are not on the boundary  $\partial\Omega$ . Therefore, if we would move the nodes of the refined mesh  $\tilde{\mathcal{T}}_1$  by the parametrization again, we get a different mesh than the refined mesh  $\mathcal{T}_1$ . This does not only hold true for the boundary, even inner nodes might differ as well. But eventually we would solve two different problems because the boundary of the undeformed body is not the same in the two models. As a remedy, one could move the new nodes to their “right” position or generate the meshes  $\mathcal{T}_\ell$  by refining  $\tilde{\mathcal{T}}_\ell$  first and then moving the nodes.

The second problem is that the base functions  $\varphi_i$  transformed on a (world) element  $\mathcal{T}_0 \subset \Omega$  differ between the two models. For example, if we choose continuous, piecewise linear finite elements  $\mathcal{P}_1$  on  $\tilde{\mathcal{T}}_0$ , the transformed base functions in the world mesh are not necessarily linear, but rather nonlinear due to the curvilinear parametrization. Of course, this difference vanishes with decreasing element sizes. However, we will not further discuss which discretization might be “better” for a problem, as this depends on the parametrization and problem itself.

We summarize these thoughts by saying that even though the parametrized and the unparametrized model both describe the same problem, their discretization with finite elements and thus the solution might differ.

## Conclusions



We can see that the unparametrized model uses the least time for the assembly. This is not surprising as a parametrization means additional computational work. Let us recall that the Jacobi matrix  $\tilde{G}$  enters the deformation gradient by  $\tilde{F} = I + \tilde{\nabla}\tilde{U}\tilde{G}^{-1}$  from equation 2.57. Computing a term like the second derivative  $w_{,FF}$  from lemma A.2.1 means a lot of calculations because  $\tilde{G}^{-1}$  enters each  $F$ ,  $G$  and  $H$  in the formula. Considering this idea, the assembly times of the parametrized model seem quite reasonable compared to the unparametrized model. It certainly means additional computational costs, but not more than a third.

Comparing the three methods from examples 5.2.7 - 5.2.9, the third has the smallest assembly time in the example. Here, we implemented the parametrization into the UFL file. The advantage was the following: The Jacobi matrix  $\tilde{G}$  as well as its inverse  $\tilde{G}^{-1}$  have four zero entries. When compiling the UFL file with the FFC, it can recognize these zeros and discards any multiplication with a zero. This shortens the assembly formula and thus the computational costs. In the first and second method, the FFC cannot recognize any matrix entries as potential zeros, because they are given as a **Coefficient** and therefore might be non-zero.

### 5.2.8. Hierarchical Plate Model

In Section 2.5 we introduced the hierarchical plate model with the ansatz

$$U(\mathbf{x}) = \sum_{i=0}^{D^{2D}} p_i\left(\frac{1}{d}\mathbf{x}_3\right) U_i^{2D}(\mathbf{x}_1, \mathbf{x}_2) \quad (\mathbf{x}_1, \mathbf{x}_2) \in \Omega^{2D}, \mathbf{x}_3 \in [-d, d].$$

The coefficients  $p_i(\frac{1}{d}\mathbf{x}_3)$  in front of  $U_i^{2D}$  in the ansatz (2.62) are the Legendre polynomials from remark 2.5.3 and their derivatives which are part of the `common.py`.

```

1  # define Legendre polynomials
2  def pol1(x):
3      return x
4  def pol2(x):
5      return 0.5*(3*x**2-1)
6  def pol3(x):
7      return 0.5*(5*x**2-3*x)
8  def pol4(x):
9      return 0.125*(35*x**4-30*x**2+3)
10 # define the derivatives of the Legendre polynomials
11 def pol1_1(x):
12     return 1
13 def pol2_1(x):
14     return 3*x
15 def pol3_1(x):
16     return 0.5*(15*x-3)

```



```

17 def pol4_1(x):
18     return 0.5*(35*x**3-15*x)

```

The weak formulation (2.66) is a nonlinear PDE in the two-dimensional domain  $\Omega^{2D}$ , so we declare some basic two dimensional finite elements.

```

1 # 2D elements for the plate model
2 cell2 = triangle
3 elem2F0 = FiniteElement( "DG", cell2, degree=0 )
4 elem2V0 = VectorElement( "DG", cell2, degree=0, dim=3 )
5 elem2V1 = VectorElement( "Lagrange", cell2, degree=1, dim=3 )

```

The displacement functions  $U_i^{2D}$  are discretized by piecewise linear finite elements  $\mathcal{P}_1$ . Depending on the degree  $D^{2D}$  of the plate model, we have mixed elements of the following kind.

```

1 # finite elements for the state space for the plate model of
   degree D
2 elemU = MixedElement([elem2V1,elem2V1]) #
   1
3 elemU = MixedElement([elem2V1,elem2V1,elem2V1]) #
   2
4 elemU = MixedElement([elem2V1,elem2V1,elem2V1,elem2V1]) #
   3
5 elemU = MixedElement([elem2V1,elem2V1,elem2V1,elem2V1,elem2V1]) #
   4

```

A function of type `elemU` is a vector-valued function with  $3(D^{2D} + 1)$  components. The chosen sorting is

$$\left( U_0^{2D}, U_1^{2D}, \dots, U_{D^{2D}}^{2D} \right).$$

Instead of using the `split` command, we define an auxiliary function `U_i` which returns  $U_i^{2D}$  for a given  $i$ .

```

1 # return the function U_i with the index i
2 def U_i(u,i):
3     return as_vector([u[3*i],u[3*i+1],u[3*i+2]])

```

With this function `U_i` and the Legendre polynomials `pol`, we define the function `Plate` to evaluate an ansatz function at a given point  $\mathbf{x} = \frac{1}{d}\mathbf{x}_3$ , for the numerical integration over the thickness.

```

1 def Plate1(u,x): # displacement ansatz for degree 1
2     return U_i(u,0) + pol1(x)*U_i(u,1)
3 def Plate2(u,x): # displacement ansatz for degree 2

```



```

4   return U_i(U,0) + pol1(x)*U_i(U,1) + pol2(x)*U_i(U,2)
5 def Plate3(u,x): # displacement ansatz for degree 3
6   return U_i(U,0) + pol1(x)*U_i(U,1) + pol2(x)*U_i(U,2) \
7         + pol3(x)*U_i(U,3)
8 def Plate4(u,x): # displacement ansatz for degree 4
9   return U_i(U,0) + pol1(x)*U_i(U,1) + pol2(x)*U_i(U,2) \
10         + pol3(x)*U_i(U,3) + pol4(x)*U_i(U,4)

```

Likewise, we can define the gradient from (2.64).

```

1  # auxiliary function for the sorting in the gradient
2  # a is the gradient wrt x_1,x_2
3  # b is the gradient wrt x_3,    2d is the thickness
4  def as_plate_grad(a,b,d)
5      return as_matrix([ [a[0,0],a[0,1],b[0]/d],\
6                          [a[1,0],a[1,1],b[1]/d],\
7                          [a[2,0],a[2,1],b[2]/d] ])
8
9  # gradient for plate model of degree 1
10 def Plate1Grad(u,x,d):
11     a = grad(U_i(U,0)) + pol1(x)*grad(U_i(U,1))
12     b = pol1_1(x)*U_i(U,1)
13     return as_plate_grad(a,b,d)
14
15 # gradient for plate model of degree 2
16 def Plate2Grad(u,x,d):
17     a = grad(U_i(U,0)) + pol1(x)*grad(U_i(U,1)) + pol2(x)*grad(U_i(U,2))
18     b = pol1_1(x)*U_i(U,1) + pol2_1(x)*U_i(U,2)
19     return as_plate_grad(a,b,d)
20
21 # gradient for plate model of degree 3
22 def Plate3Grad(u,x,d):
23     a = grad(U_i(U,0)) + pol1(x)*grad(U_i(U,1)) + pol2(x)*grad(U_i(U,2)) \
24         + pol3(x)*grad(U_i(U,3))
25     b = pol1_1(x)*U_i(U,1) + pol2_1(x)*U_i(U,2) \
26         + pol3_1(x)*U_i(U,3)
27     return as_plate_grad(a,b,d)
28 # gradient for plate model of degree 4
29 def Plate4Grad(u,x,d):
30     a = grad(U_i(U,0)) + pol1(x)*grad(U_i(U,1)) + pol2(x)*grad(U_i(U,2)) \
31         + pol3(x)*grad(U_i(U,3)) + pol4(x)*grad(U_i(U,4))
32     b = pol1_1(x)*U_i(U,1) + pol2_1(x)*U_i(U,2) \

```



```

33     +pol3_1(x)*U_i(U,3) +pol4_1(x)*U_i(U,4)
34     return as_plate_grad(a,b,d)

```

The next part concerns the integration over the thickness, see remark 2.5.5. This can be done in two ways: either we pass the integration point  $x_j$  as a **Constant** in the UFL file and then assemble the Gauss quadrature formula in the C++ code, or we write the Gauss quadrature into the UFL file. The first allows a more flexible switch between the degrees  $D^{\text{GL}}$  of the Gauss quadrature and saves lines in the UFL and .h header files since we need only one assembly routine for integrals where only the integration point  $x_j$  varies. On the other hand, as we have seen in the conclusions of section 5.2.6, the offset time for the assembly is noticeable. Therefore it might be better to write the Gauss quadrature into the UFL file and run the assembly routine only once.

We start with the definition of the integration points  $x_j$  and weights  $\omega_j$  from the Gauss-Legendre quadrature formula from table 2.4. We denote the values  $x_j$  and  $w_j$  of the formula with degree  $i$  in the UFL by `x_int{i}{j}` and `w_int{i}{j}`, e.g. `x_int23` =  $x_3 = \frac{1}{5}\sqrt{15}$  for degree 2. These values are saved in the UFL file `common.py`. We start by collecting the integration points and weights for easier use. The factor  $d$  from the quadrature formula (2.68) is included in the weights `w_{i}`.

```

1  # Gauss-Legendre degree D^GL = 2
2  d    = thick*0.5
3  x_1 = x_int21
4  x_2 = x_int22
5  w_1 = w_int21*d
6  w_2 = w_int22*d
7  # Gauss-Legendre degree D^GL = 3
8  d    = thick*0.5
9  x_1 = x_int31
10 x_2 = x_int32
11 x_3 = x_int33
12 w_1 = w_int31*d
13 w_2 = w_int32*d
14 w_3 = w_int33*d
15 # ... (analogously for higher degrees)

```

For a unified usage of later terms, we define a functions `MyPlate` and `MyGrad` to evaluate the displacement from (2.62) and the plate gradient from (2.64).

```

1  def MyPlate(U,x):           # choose only one line
2      return Plate1(U,x)     # for plate degree = 1
3      return Plate2(U,x)     # for plate degree = 2
4      return Plate3(U,x)     # for plate degree = 3
5      return Plate4(U,x)     # for plate degree = 4

```



```

6
7 def MyGrad(U,x):           # choose only one line
8     return Plate1Grad(U,x,d) # for plate degree = 1
9     return Plate2Grad(U,x,d) # for plate degree = 2
10    return Plate3Grad(U,x,d) # for plate degree = 3
11    return Plate4Grad(U,x,d) # for plate degree = 4

```

With these auxiliary functions in the UFL file `common.py`, we can write the terms of the sum in the quadrature formula as individual terms and finally add them. First we specify some basic definitions.

```

1 # define test and trial function
2 testU = TestFunction(elemU)
3 trialU = TrialFunction(elemU)
4
5 # solution of the previous iteration
6 U      = Coefficient(elemU)
7
8 # material constants and thickness 2d
9 mu      = Constant(cell2)
10 lambda = Constant(cell2)
11 thick   = Constant(cell2)
12
13 # loads
14 f = Coefficient(elem2V0)
15 g = Coefficient(elem2V0)

```

We demonstrate the integration over the thickness with the stored energy  $W = W^{(T)} + W^{(f)} + W^{(g)}$ . Recalling (2.69), the quadrature formula for the first part  $W^{(T)}$  reads

$$\begin{aligned}
 W^{(T)}(\mathbf{U}) &= \int_{\Omega} w(F(\mathbf{x})) \, d\mathbf{x} = \sum_{j=1}^{D^{\text{GL}}} d\omega_j \int_{\Omega^{2D}} w(\underbrace{F(\mathbf{x}_1, \mathbf{x}_2, dx_j)}_{=: F_j(\mathbf{x}_1, \mathbf{x}_2)}) \, d\mathbf{x}_1 d\mathbf{x}_2 \\
 &= \sum_{j=1}^{D^{\text{GL}}} d\omega_j \int_{\Omega^{2D}} w(F_j(\mathbf{x}_1, \mathbf{x}_2)) \, d\mathbf{x}_1 d\mathbf{x}_2.
 \end{aligned}$$

Hence we will define the  $F_j := F(\mathbf{x}_1, \mathbf{x}_2, dx_j)$  in the UFL file to re-use them later for the energy term.

```

1 F1 = Identity(3) + MyGrad(U,x_1)
2 F2 = Identity(3) + MyGrad(U,x_2)
3 # ... continue with more terms for more integration points

```



The energy term  $W^{(f)}$  for a volume load  $\mathbf{f}$ , which we assume to be constant over the thickness, was integrated in (2.70), that is

$$W^{(f)}(U) = \int_{\Omega} \mathbf{f}(\mathbf{x})^{\top} U(\mathbf{x}) \, d\mathbf{x} = \sum_{j=1}^{D_{GL}} d\omega_j \int_{\Omega^{2D}} \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2) \cdot \underbrace{U(\mathbf{x}_1, \mathbf{x}_2, d x_j)}_{=MyPlate(U, x_j)} \, d\mathbf{x}_1 d\mathbf{x}_2.$$

We assumed that the boundary load  $\mathbf{g}$  acts only on  $\Gamma^N = \Omega^{2D} \times \{-d, d\}$ , that is the upper and lower surface, and is equal on both surfaces, compare the discussion prior to equation (2.71). The energy term  $W^{(g)}$  then is

$$\begin{aligned} W^{(g)}(U) &= \int_{\Gamma^N} \mathbf{g}(\mathbf{x}) \cdot U(\mathbf{x}) \, dS \\ &= \int_{\Omega^{2D}} \mathbf{g}(\mathbf{x}_1, \mathbf{x}_2) \cdot \left( \underbrace{U(\mathbf{x}_1, \mathbf{x}_2, -d)}_{=MyPlate(U, -1)} + \underbrace{U(\mathbf{x}_1, \mathbf{x}_2, d)}_{=MyPlate(U, 1)} \right) \, d\mathbf{x}_1 d\mathbf{x}_2. \end{aligned}$$

Finally, we sum up the terms  $W^{(T)}$ ,  $W^{(f)}$  and  $W^{(g)}$  to get the stored energy

```

1 # define the terms for the sum
2 W1 = EnergyDensity(F1,mu,lmbda)*dx -inner( f, MyPlate(testU,x_1)
   ) * dx
3 W2 = EnergyDensity(F2,mu,lmbda)*dx -inner( f, MyPlate(testU,x_2)
   ) * dx
4 # ...
5 # build the sum of the quadrature formula
6 W = w_1*W1 + w_2*W2 \ # + ...
7     -inner( g, MyPlate(testU,-1) ) * dx \
8     -inner( g, MyPlate(testU, 1) ) * dx

```

The same procedure can be applied for the first derivative  $W_{,U}$  and the second derivative  $W_{,UU}$  of the stored energy  $W$ , see lemma (A.2.1).

```

1 # define the first derivative W_U
2 W_U1 = EnergyDensity_1st(F1,MyGrad(testU,x_1),mu,lmbda)*dx \
3     -inner( f, MyPlate(testU,x_1) ) * dx
4 W_U2 = EnergyDensity_1st(F2,MyGrad(testU,x_2),mu,lmbda)*dx \
5     -inner( f, MyPlate(testU,x_2) ) * dx
6 # ...
7 # build the sum of the quadrature formula
8 W_U = w_1*W_U1 + w_2*W_U2 \ # ...
9     -inner( g, MyPlate(testU,-1)+MyPlate(testU,1) ) * dx

```



```

1 # define the second derivative W_UU
2 W_UU1 = EnergyDensity_2nd(F1,MyGrad(testU,x_1),
3                               MyGrad(trialU,x_1),mu,lmbda)*dx
4 W_UU2 = EnergyDensity_2nd(F2,MyGrad(testU,x_2),
5                               MyGrad(trialU,x_2),mu,lmbda)*dx
6 # ...
7 W_UU = w_1*W_UU1 + w_2*W_UU2 # + ...

```

Other forms like a guiding function or a preconditioner can be handled analogously.

```

1 # guiding function
2 beta = Constant(cell2)
3 G1 = guiding(F1,beta)*dx
4 G2 = guiding(F2,beta)*dx
5 # ...
6 G = w_1*G1 + w_2*G2 # + ...
7
8 # the preconditioner
9 a_prec1 = LEscal(MyGrad(testU,x_1),MyGrad(trialU,x_1),mu,lmbda)*
10           dx
11 a_prec2 = LEscal(MyGrad(testU,x_2),MyGrad(trialU,x_2),mu,lmbda)*
12           dx
13 # ...
14 a_prec = w_1*a_prec1 + w_2*a_prec2 # + ...

```

We complete the UFL file by stating which forms are assembled.

```

1 forms = [ W_UU, W_U, a_prec, L_prec, G ]

```

## 5.3 Solver Routines

This section concerns the implementation of the solver algorithms in the C++ code. We will not give too many details but rather show the basic ideas how the routines work. The presented code fragments will omit various lines and will not be complete in most cases. We concentrate on the core ideas and illustrate the functionalities of the routines.

### 5.3.1. Multigrid Method

We introduced the multigrid V-cycle (alg. 3.5.5) in section 3.5 as a preconditioner for the CG and MINRES method. Ospald (2012) implemented a geometrical multigrid method called FMG for FENICS 1.2.0. The project homepage (Ospald (2014)) on Launchpad offers the latest version with instructions and demos. The following code is built on basic routines of FMG:



- the multigrid level structure `mg_level` including the Galerkin interpolation for  $A_\ell$  and the V-cycle algorithm itself,
- the assembly routine for the prolongation matrix  $p_{\ell \rightarrow \ell+1}$ ,
- the `adaptor` to adjust the variational problems on the different meshes<sup>2</sup> (which is a modified version of the one from FEniCS 1.0.0, see (Ospald, 2012, ch. 3.2)),
- the smoothing operators.

### The Class `mg_level`

The hierarchy of the different multigrid levels  $\ell = 0, 1, \dots, L$  is stored in the class `mg_level` which contains information on the following:

- pointer to the next coarser and finer level  $\ell - 1$  and  $\ell + 1$ .
- the prolongation matrix  $p_{\ell, \ell+1}$ .
- the (current) coefficient matrix  $A_\ell$  of the linear system.
- the smoothing operators from FMG.
- the Dirichlet BCs on the level  $\ell$ .

We illustrate the initialization of the class `mg_level` with an example: we have a linear variational equality  $a(u, v) = l(v)$  stored in the class `LinearVariationalProblem` and construct a multigrid hierarchy for it. For example, the bilinear form  $a$  could represent the preconditioner `a_prec` from section 5.2.3. The following code is called at the beginning of the program and not only refines the mesh, but also adapts the variational problems, that means the bilinear and linear forms are updated to the new levels. The results are the meshes and adapted problems on each level, as well as the multigrid structure of class `mg_level`. The levels and problems are stored in a `std::vector` for an easier access later.

```

1 // passed objects:
2 // LinearVariationalProblem* _problem - the variational problem
3 // int L - the number of mesh levels/refinements
4 ...
5 // init. vectors for an easier access of the different levels
6 std::vector<boost::shared_ptr<mg_level> > levels;
7 std::vector<boost::shared_ptr<LinearVariationalProblem> >
   problems;
8 levels.resize(L+1);
9 problems.resize(L+1);
10 // pass the problem to the coarsest level
11 problems[0].reset(new LinearVariationalProblem(*_problem));
12
13 // pointer for the current mesh

```

<sup>2</sup>An object `Form` in FEniCS is always defined on a discretized function space. Refining the underlying mesh requires an adaptation of the forms which is done by the FMG-routine `adaptor`.



```

14  const Mesh* mesh_i = problems[0]->test_space()->mesh().get();
15
16  for (int i = 0; i <= L; i++)
17  { // create new level
18      levels[i].reset(new mg_level());
19      if (i > 0)
20      { // set parent/child relation between levels
21          levels[i-1]->set_child(levels[i]);
22          // adapt mesh
23          fmg::Adaptor::adapt(*mesh_i);
24          mesh_i = &(mesh_i->child());
25          // adapt problem
26          fmg::Adaptor::adapt(*problems[i-1], *mesh_i);
27          problems[i].reset(new
28              LinearVariationalProblem(problems[i-1]->child())
29              );
30      }
31      // set function space
32      levels[i]->space = problems[i]->test_space();
33      // init prolongation
34      if (i > 0)
35          levels[i-1]->init_prolongation();
36  }

```

We omitted several lines here, like the handling of the Dirichlet boundary data (which has to be adapted too) and the initialization of certain vectors. In case of the nonlinear problem  $0 = W_U$ , we also have to include lines for the adaptation of the preconditioner, the guiding function and other forms.

For a compatibility with PETSC routines, we define the preconditioner with the appropriate class.

```

1  // multigrid preconditioner
2  class mg_prec : public PETScUserPreconditioner
3  { public:
4      mg_prec(boost::shared_ptr<mg_level> _level0) : level0(_level0)
5      {
6          void solve(PETScVector& x, const PETScVector& b)
7          { level0->recursive_mg(x, b); } // apply one V-.cycle
8      protected:
9          boost::shared_ptr<mg_level> level0;
10     };

```



Before calling the preconditioner, the matrices  $A_\ell$  have to be set on the multigrid levels  $\ell = 0, \dots, L$ . Instead of assembling the matrices on each level, we assemble only the matrix  $A_L$  on the finest level and compute the Galerkin assembly of  $\{A_\ell\}_{\ell=0}^{L-1}$  recursively, see equation (3.25).

```

1 // assemble A and b on the finest mesh and store it in levels[L]
2 assembler.assemble(*(levels[L]->A), *(levels[L]->b0),
3   *(problems[L]->bilinear_form()), *(problems[L]->residual_form())
4   );
5 // do Galerkin assembly on coarser levels
6 for (int i = L-1; i >= 0; i--)
7     levels[i]->init_Galerkin_A();
8
9 // init coarse level solver and smoothers
10 levels[0]->init_solver();
11 for (std::size_t i = 1; i < L+1; i++) {
12     levels[i]->init_smoother();
13 }
14 mg_prec p(levels[c_level_it]);

```

With these pieces of information, the routine `mg_level` has everything in order to perform the V-cycle from algorithm 3.5.5. For example, in a Krylov subspace method, the preconditioner is applied on a residual  $\vec{r}$  to compute the preconditioned residual  $\vec{w}$ , that can be called by:

```

// call of the preconditioner
// r - the residual
// w - the preconditioned residual
p->solve(w,r);

```

### 5.3.2. CG and MinRes Method

There are PETSC implementations for the CG and MINRES method, but we implemented both methods again due to the following deficiencies:

- There is a stopping criterion with the “preconditioned norm” in the PETSC-CG and -MINRES, but it actually computes the Euclidean  $l_2$ -norm of the preconditioned residual  $\vec{w} = P^{-1}\vec{r}$ , even though the norm of the residual can be easily computed by

$$\|r\|_{\mathcal{U}^*}^2 = \vec{r}^\top P^{-1} \vec{r} = \vec{r}^\top \vec{w}.$$

- The implementation of the CG method in PETSC does not incorporate the option to stop when a negative curvature is detected (truncated CG).



The implementation of a truncated CG method (algorithm 3.3.1) and a MINRES method (algorithm 3.3.3) in C++ is straightforward.

### 5.3.3. Forward Problem

In order to solve the forward problem (2.46), that is

$$0 = W_{,\mathcal{U}}(\mathbf{U}_\star)[\delta\mathbf{U}], \quad \forall \delta\mathbf{U} \in \mathcal{U},$$

we apply Newton's method from algorithm 3.1.1. This includes a line search as well as the truncated CG method with a multigrid V-cycle preconditioner. The Newton system (3.2) as well as the bilinear form for the preconditioner are passed as a variational problem to the Newton solver, along with a possible guiding function and the number of refinements for the mesh.

```
1 // exemplary constructor for the Newton solver
2 fmgNewtonSolver solver(problem, prec_problem, G, L);
```

Due to the number of lines in the code for the Newton solver, we give a simplistic pseudo-code in algorithm 5.3.1. The norm  $\|W_{,\mathcal{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*}$  from the stopping criterion (3.3) is computed by the (dual) norm of the right-hand side  $\vec{b}$  (the discretization of  $W_{,\mathcal{U}}(\mathbf{U}_k, \mathbf{C})$ ) of the Newton system  $A\vec{x} = \vec{b}$ , e. g.

$$\|W_{,\mathcal{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*} = \vec{b}^\top P^{-1} \vec{b} = \|\vec{b}\|_{R_h},$$

where  $P^{-1}$  is the application of the preconditioner, which has the same role as the discretized Riesz operator  $R_h = P^{-1}$ .

**Algorithm 5.3.1** (Newton forward solver).

**Input:** problem, prec\_problem, I, G, L

**Output:** solution: displacement  $\mathbf{U}$  and control  $\mathbf{C}$

- 1: refine the mesh and initialize the multigrid levels for problem, prec\_problem, I and G
- 2: **if** (nested iterations are used) **then**
- 3:   Start on  $\mathcal{T}_0$
- 4: **else**
- 5:   Start on  $\mathcal{T}_L$
- 6: **end if**
- 7:  $k := 0$ , set  $\mathbf{U}_0 \equiv \mathbf{0}$  if not given otherwise
- 8: assemble the matrix  $P$  for the preconditioner
- 9: **while** (not converged) **do**
- 10:   assemble the Newton matrix  $A$  and rhs  $\vec{b}$  at the current iterate  $\mathbf{U}_k$   
       (this includes a symmetrical modification from the Dirichlet BCs)
- 11:   solve the Newton system with a truncated CG method



```

12: perform a line search to compute a step length  $\alpha$ 
13: set  $\mathbf{U}_{k+1} := \mathbf{U}_k + \alpha \Delta \mathbf{U}_k$ 
14: assemble the rhs  $\vec{b}$ 
15: if ( $\|W_{,\mathbf{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*} = \|\vec{b}\|_{R_h}$  satisfies the stopping criterion) then
16:     set converged := TRUE
17: end if
18: if (nested iterations are used and converged = TRUE) then
19:     increase the current level  $\ell \rightarrow \ell + 1$ 
20:     interpolate  $\mathbf{U}_k$  to the new level
21:     assemble the matrix  $P$  for the preconditioner on the new level
22:     set converged := FALSE
23: end if
24: set  $k := k + 1$ 
25: end while
26: return  $\mathbf{U}_k$  and  $\mathbf{C}_k$ 

```

**Remark 5.3.2** (On algorithm 5.3.1). The right-hand side  $\vec{b}$  is assembled twice in each Newton iteration. The reason for this lies in the assembly routines of FEniCS. As we use a CG or MINRES method, we need to have symmetric matrices, so the Dirichlet BCs have to be applied symmetrically on  $A$ , which is done by the FEniCS built-in routine `SystemAssembler` which needs bilinear and linear forms simultaneously and assembles both, matrix  $A$  and right-hand side  $\vec{b}$ . The symmetric application of the Dirichlet BCs in this routine is done locally on the elements, since a modification of rows and columns is cheaper for the element matrix than for the global matrix  $A$ .

However, if we would have inhomogeneous Dirichlet BCs and they are not met yet, we have to assemble the matrix  $A$  because the modification of the right-hand side  $\vec{b}$  depends on the matrix  $A$ . On the other hand, if the Dirichlet BCs are not met, we do not have to check the stopping criterion because Newton's method is not finished anyway.

### 5.3.4. Lagrange-Newton Problem

The C++ implementation of the Lagrange-Newton method from algorithm 4.2.2 is very similar to the C++ code in algorithm 5.3.1 for the forward system. The main two differences are the additional term for the objective function  $I$  and the different handling of the block preconditioner (4.27). As the objective function is quite straightforward to implement, the block preconditioner requires some additional work, for example the scaling factors  $s_C, s_U, s_Z > 0$  and the extraction of the three components  $\mathbf{U}, \mathbf{C}, \mathbf{Z}$  from the PETSC vector  $\mathbf{X} = (\mathbf{U}, \mathbf{C}, \mathbf{Z})$ , though we will



not go into detail on the last point. The C++ code for the block preconditioner (without the lines for the extraction) then reads:

```

1  // Block preconditioner with multigrid
2  class block_UCZ_mg_prec : public PETScUserPreconditioner
3  {
4  public:
5      block_UCZ_mg_prec(mg_level _level0, GenericMatrix diag,
6                        double* _scaling) :
7                          level0(_level0), diag(_diag), scaling(_scaling) {}
8      void solve(PETScVector& X, const PETScVector& b)
9      { // get the sub-vectors rhsU, rhsC and rhsZ from b
10         // (lines are omitted)
11
12         // direct solver for the C-block (diagonal mass matrix)
13         LinearSolver lsolver("lu");
14         lsolver.solve(*diag, *C, *rhsC);
15         // multigrid V-cycle for the U block and the Z block
16         level0->recursive_mg(*U, *rhsU);
17         level0->recursive_mg(*Z, *rhsZ);
18         // apply the scaling on the blocks
19         *U *= 1.0/scaling[0];
20         *C *= 1.0/scaling[1];
21         *Z *= 1.0/scaling[2];
22
23         // rebuild the vector X from (U,C,Z) (lines are omitted)
24
25         return;
26     }
27 };

```

### Stopping Criterion and Scaling Factors

The choice of the scaling factors  $s_U$ ,  $s_C$ ,  $s_Z$  influences the norm  $\|\cdot\|_{X^*}$  of the stopping criterion of the Lagrange-Newton (LN). In order to test different combinations of  $s_U$ ,  $s_C$ ,  $s_Z$ , we set the scaling factors to  $s_U = s_C = s_Z = 1$  so that the stopping criterion stays the same and we can compare the performance for different scalings. The testing itself is done by fixing  $s_U = 1$  and varying  $s_C, s_Z$ , for example in powers of 10. Then the overall computational time is measured, next to the number of LN and MINRES iterations. Figure 5.3 shows an example for the optimal control problem from section 6.2.1. We can see that the number of MINRES iterations is more sensitive with respect to the factor  $s_C$  than the factor  $s_Z$  in this example. Furthermore, the plots for the Lagrange-Newton iterations and the overall running time resemble this figure quite well. Still, we are careful with the choice of the



scaling factors as combinations like  $s_U = 1, s_C = 10^{-10}, s_Z = 10^{10}$  are likely to cause problems due to numerical rounding errors.

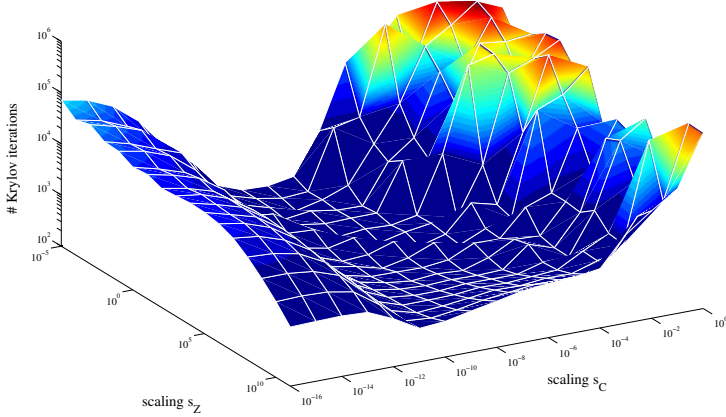


FIGURE 5.3. Plot of the total number of MINRES iterations against the scaling factors  $s_Z, s_C$ .

### 5.3.5. Reduced Problem

This section will illustrate the C++ implementation of the quasi-Newton solver from algorithm 4.3.15 to solve the reduced optimal control problem 4.32. Again, we will not give the code at length but rather show the main structure.

We start by naming the required objects that are passed to the solver. We have

- the variational problem `fwrdd_problem` for the problem  $U = S(C)$  from (4.31),
- the preconditioner for the state  $U$  and the derivative  $I_U$  both merged into a variational problem `prec_U_problem`,
- the mass matrix  $M_C$  and the construction of the derivative  $I_C^{\text{red}}$  from (4.38), both merged into bilinear and linear forms in the variational problem `prec_C_problem`,
- the objective function  $I$  as a form `I`,
- a possible guiding function  $G$  as a form `G`,
- and the number of mesh refinements `L`.



The merging of the preconditioner with  $I_U$  and the mass matrix  $M_C$  with  $I_C^{\text{red}}$  as variational problems is a rather convenient implementation and does not necessarily represent a variational problem which has to be solved.

Besides the multigrid levels, several intermediate/current functions are stored, among them the state  $U$ , the control  $C$ , the adjoint  $Z$ , the derivative  $I_{\text{1st}} = I_C^{\text{red}}$  and the auxiliary functions  $s_i \in C$  and  $y_i \in C^*$  for the BFGS-update. The solver itself is divided into several subroutines, among them are the following:

`init_levels()`

This subroutine is called at the beginning and refines the mesh and builds the multigrid levels for the three variational problems `fwrdd_problem`, `prec_U_problem` and `prec_C_problem`, as well as for the forms  $I$ ,  $G$  and  $L$ .

`SolveFwrdd()`

This call uses the forward solver from section 5.3.3 to solve the forward problem given in `fwrdd_problem`. The result is the state  $U = S(C)$  associated with the current control  $C$ . This includes the line search with a guiding function  $G$  as well as a truncated CG method with the multigrid V-cycle preconditioner from `prec_U_problem`. As we mentioned earlier, we have to solve the forward system preferably accurately so that the Wolfe-Powell conditions (4.45) can be satisfied.

`SolveAdj()`

When the current state  $U = S(C)$  solves the forward problem with the current control  $C$ , the adjoint equation can be solved to compute the adjoint state  $Z$ . The coefficient matrix for the (linear) adjoint equation is the last Newton matrix from the forward solver `SolveFwrdd` and therefore it does not need to be re-assembled. The right-hand side of the system is the derivative  $I_U$  which is stored as the linear form of the variational problem `prec_U_problem`.

Since we assume that  $U_k$  is a local minimum of the energy minimization problem, the stiffness matrix  $A$  is assumed to be positive (semi-)definite. Therefore we can use a CG method to solve the adjoint system, though we still pay attention to negative curvature. Like the forward system, we have to solve the adjoint system quite accurately so that the Wolfe-Powell conditions (4.45) can be satisfied. Afterwards, the reduced derivative  $I_C^{\text{red}}$  is assembled, using the just computed adjoint  $Z$  and the linear form of `prec_C_problem`.

`inv_LM_BFGS_update()`

This subroutine performs the inverse, limited-memory BFGS-update from algorithm 4.3.10 on the (negative) derivative  $-I_C^{\text{red}}$  to compute the search direction  $\Delta C$ . The BFGS-auxiliary functions  $s_i$  and  $y_i$  and the scalars  $\rho_i = (y_i^\top s_i)^{-1}$  are each stored in a `std::vector` for an easier handling. As this subroutine is rather short, we show it exemplarily.



```

1 void fmgQuasiNewtonSolver::inv_lm_BFGS_update(boost::shared_ptr<
    const Vector> I_1st, boost::shared_ptr<Function> p)
2
3 std::size_t k = s.size(); // current number of BFGS-functions
4 double alpha[k];
5 double beta;
6 Vector q( *I_1st ); // copy vector
7
8 for (int i=k-1; i>=0; i--)
9 { alpha[i] = rho[i]* q.inner( *(s[i]->vector()) );
10   q.axpy( -alpha[i], *(y[i]) );
11 }
12 // Applying H_0 means solving with the mass matrix M_C
13 LinearSolver lsolver("lu");
14 lsolver.solve(*(prec_C_levels[c_level_it]->A), *p->vector(), q );
15
16 for (std::size_t i=0; i<k; i++)
17 { beta = rho[i]* y[i]->inner( *(p->vector()) );
18   p->vector()->axpy( (alpha[i]-beta), *(s[i])->vector() );
19 }

```

#### WolfePowell\_linesearch()

The Wolfe-Powell line search from algorithm 4.3.12 along the search direction  $\Delta \mathbf{C}_k$  is used to find a step length satisfying the Wolfe-Powell conditions (4.45). The implementation is quite straightforward, still we give some remarks on it.

We store the starting points  $\mathbf{C}_k$  and  $\mathbf{U}_k$  and modify the variable holding the current control  $\mathbf{C} = \mathbf{C}_k + \alpha \Delta \mathbf{C}_k$ . Whenever the control is changing, we need to “update” the current state by solving the forward problem, i.e. `SolveFwrd()`. This is mandatory for the evaluation of the reduced objective function  $I^{\text{red}}$  as well as for the computation of the reduced derivative  $I_{,\mathbf{C}}^{\text{red}}$ .

In case of divergence of the forward solver, which might happen if the new control  $\mathbf{C}_{k+1} = \mathbf{C}_k + \alpha \Delta \mathbf{C}_k$  is too large to yield a reasonable deformation, the step length  $\alpha$  is drastically reduced, e.g. by a factor  $\frac{1}{10}$ . This can also be interpreted as a line search for the forward solver which simply “tries” whether Newton’s method succeeds in a given number of iterations, hence it could be understood as a very simple trial and error approach to the globalization problem discussed in section 3.2.

As the Wolfe-Powell line search calls the forward solver `SolveFwrd()` and the adjoint solver `SolveAdj()`, the main computational work occurs during this line search. Also, once an admissible step length  $\alpha$  is found, the control is already updated to  $\mathbf{C}_{k+1} = \mathbf{C}_k + \alpha \Delta \mathbf{C}_k$ , as well as the state  $\mathbf{U} = S(\mathbf{C}_{k+1})$  and the reduced derivative  $I_{,\mathbf{C}}^{\text{red}}(\mathbf{C}_{k+1})$ .



`solve()`

The quasi-Newton algorithm 4.3.15 itself is called by `solve()`. It starts the optimization, uses the previous subroutines and manages the handling of the BFGS auxiliary functions  $s_i \in \mathcal{C}$  and  $y_i \in \mathcal{C}^*$  which are stored in lists of type `std::Function` respectively `std::vector`. The implementation directly follows the algorithm.

## 5.4 An Experimental Preconditioner

So far, we used the linear model of elasticity as a preconditioner for  $\mathbf{U}$ , or to be more precise, an approximation by a multigrid V-cycle. The advantage of this choice is clearly its positive definiteness, though it might not be very “similar” to the stiffness matrix  $A_h$  of the nonlinear model, defined in (3.20). Of course, we expect both models to differ from each other for large deformations, so we can expect that the choice  $P = K_{\text{lin,elast}}$  might be not as good for large deformations as it was for small deformations. On the other hand, a V-cycle with the current stiffness matrix  $A_h$  of the nonlinear model might be “closer” to the matrix  $A_h$  itself, but we cannot guarantee its positive definiteness, which is a requirement for a preconditioner in a CG and MINRES methods. Therefore, if we use a V-cycle of  $A_h$  as a preconditioner  $P$ , we have to take some precautions. Let  $\vec{r}$  be the current residual and  $\vec{w} = P^{-1}\vec{r}$  the preconditioned residual. The scalar product  $\vec{r}^\top \vec{w} = \vec{r}^\top P^{-1}\vec{r}$  shows up in the CG (alg. 3.3.1) and the MINRES method (alg. 3.3.3). In both methods, the term  $\sqrt{\vec{r}^\top \vec{w}}$  is computed, either for the stopping criterion in the CG method or the scaling  $\gamma_1$  in the MINRES method. If the square root does not exist because the scalar product  $\vec{r}^\top \vec{w} < 0$  is negative, both methods cannot be expected to continue to work, since a symmetric positive definite preconditioner is a requirement in both. This leads to the following idea.

### “Truncated” Idea for the Preconditioner

We first consider the forward solver with a truncated CG method to solve the Newton system. Similar to the idea of the truncated CG, we stop the CG method if we detect the scalar product  $\vec{r}^\top \vec{w}$  to be negative. This can be understood as a (second) negative curvature check for the (inverted) preconditioner  $P^{-1}$ . Until this point, the matrix  $A_h$  has a positive curvature in the Krylov subspace as well as for the (inverted) preconditioner  $P^{-1}$ . Hence, we are still in the context of a truncated CG method, but now with a possibly earlier stop if the preconditioner is not positive definite. Still, the same arguments from the truncated CG can be applied here again. If the negative curvature for  $P^{-1}$  is detected in the first iteration, we perform a V-cycle with the stiffness matrix  $K_{\text{lin,elast}}$  from (2.28) as a preconditioner which is guaranteed to be positive definite. This then can be understood as a safety preconditioner to which the algorithm falls back in case the preconditioner  $P = A_h$  does not work.



Of course, the experimental preconditioner  $P = A_h$  has an influence on the norm  $\|\cdot\|_R$  in the stopping criterion of the CG method, because the choice of the Riesz operator  $R = P^{-1}$  is equivalent to the choice of the preconditioner. However, we keep the norm based on  $K_{\text{lin,elast}}$  for the stopping criterion (3.3) of the Newton solver, e. g.

$$\frac{\|W, \mathbf{U}(\mathbf{U}_k, \mathbf{C})\|_{K_{\text{lin,elast}}^{-1}}}{\|W, \mathbf{U}(\mathbf{U}_0, \mathbf{C})\|_{K_{\text{lin,elast}}^{-1}}} \leq \text{rTol} \quad \text{or} \quad \|W, \mathbf{U}(\mathbf{U}_k, \mathbf{C})\|_{K_{\text{lin,elast}}^{-1}} \leq \text{aTol}.$$

This requires an (additional) application of the preconditioner with  $K_{\text{lin,elast}}$ , i. e. a multigrid V-cycle with  $K_{\text{lin,elast}}$ , in each Newton step.

We can employ the same idea for the Lagrange-Newton method. The preconditioner  $P = A_h$  takes the  $\mathbf{U}$  and  $\mathbf{Z}$  block in the block preconditioner, that is

$$P = \begin{bmatrix} s_{\mathbf{U}} A_h & & \\ & s_{\mathbf{C}} M_{\mathbf{C}} & \\ & & s_{\mathbf{Z}} A_h \end{bmatrix}.$$

Again, a negative scalar product  $\vec{r}^\top \vec{w} < 0$  will cause the MINRES method to break down, so we could stop the method at this point and return the intermediate solution. If this happens in the first iteration, we perform a V-cycle with the earlier mentioned (safety) block preconditioner (4.27). Another strategy could be a MINRES method with the safety block preconditioner, though this might result in a higher number of MINRES iterations. The stopping criterion for the Lagrange-Newton method itself remains to be based on this safety preconditioner.

For the quasi-Newton method, we apply the same strategy for the solution operator `SolveFwd` that we already suggested for the forward solver. As for the adjoint solver, we still assume the (semi-)positive definiteness of  $A_h$  at a local solution  $\mathbf{U}_k = S(\mathbf{C}_k)$ , so we use a CG method with the new preconditioner which, of course, changes the norm in the stopping criterion of this CG method. However, the norm  $\|\cdot\|_{\mathbf{C}}$  in the stopping criterion of the quasi-Newton solver is independent of the preconditioner.

We test the experimental preconditioner  $P = A_h$  and compare it to the previous preconditioner  $K_{\text{lin,elast}}$  with the following example 5.4.1.

**Example 5.4.1** (Bar with Fibre Tension). Let  $\Omega = (0, 5) \times (0, 1) \times (0, 1)$  be a bar with a fiber direction of  $\mathbf{a}(\mathbf{x}) = (1, 0, 0)^\top$  aligned along the bar. The fiber tension  $m$  is given by

$$m(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = s \frac{1}{10} (\mathbf{x}_3 - \frac{1}{2}),$$



with an increment  $s \in [0, 1]$ . That means the fibers on the top tend to shorten while the fibers at the bottom tend to elongate. We choose a coarse mesh with  $11 \times 3 \times 3$  vertices and a mesh refinement level of  $L = 2$ . The relative stopping criterion for the linear solver is  $\text{rTol} = 10^{-8}$ . The solutions for the increments  $s = 0, \frac{1}{10}, \frac{2}{10}, \dots, 1$  are plotted in figure 5.4.



FIGURE 5.4. Solutions of the example 5.4.1 for the load increments  $s = 0, \frac{1}{10}, \frac{2}{10}, \dots, 1$ . The dark blue mesh is the solution for the incremental  $s = 1.0$  while the transparent red mesh in the lower right corner is the undeformed body, i. e.  $s = 0$ .

The comparison of both preconditioners in terms of the total number of CG iterations for each increment  $s \in \{\frac{1}{10}, \frac{2}{10}, \dots, 1\}$  is illustrated in figure 5.5. One can clearly see that the number of CG iterations grows with the increment  $s$ . In the case of the experimental preconditioner  $P = A_h$ , this results in about 7 to 9 CG iterations per Newton step. For the preconditioner  $K_{\text{lin.elast}}$ , the average number of CG iterations per Newton step grows from 22 for  $s = 0.1$  up to 70 for  $s = 0.8$ . Here, the negative curvature occurred much more often in the truncated CG than for  $P = A_h$ . Another reason for the better results of the experimental preconditioner is the fact that the second derivative of the energy term  $W^{(m)}$  contributes to the stiffness matrix  $A_h$ . The preconditioner  $K_{\text{lin.elast}}$  does not incorporate this term because it might not be positive definite. Hence, a V-cycle of  $K_{\text{lin.elast}}$  is only a good approximation for  $A_h$  if the deformation and the fiber tension  $m$  are small.



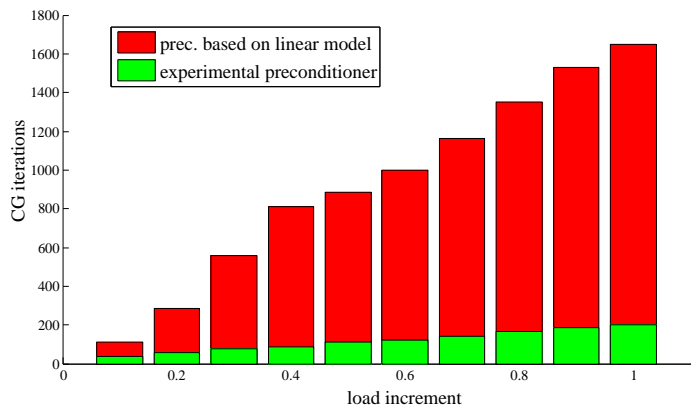


FIGURE 5.5. The total number of CG iterations in order to solve the forward problem for an increment  $s \in \{\frac{1}{10}, \frac{2}{10}, \dots, 1\}$ .



# 6 Numerical Experiments

## Contents

---

6.1	Experiments on the Forward Problem	153
6.1.1.	Line Search and Iterates of Newton's Method	153
6.1.2.	Plate Model	156
6.2	Experiments on the Optimal Control Problem	159
6.2.1.	Elevated Bar with Fiber Tension Control	159
6.2.2.	Regional Penalization	167
6.2.3.	Flower Movement by Turgor Pressure	170
6.2.4.	Enclosed Volume	173

---

This chapter presents a couple of numerical experiments on the forward problem and on the optimal control problem. First, in section 6.1, we compare the different line search methods from section 3.2 with the example 3.2.1 of a thick plate which results in divergence for a standard Newton's method. The same example is then used to test the hierarchical plate model from section 2.5 and compare it with the full 3D model. Here we also regard a thinner plate to get an idea which degree  $D^{2D}$  of the plate model is needed to solve the forward problem to satisfactory accuracy.

In section 6.2, we present four optimal control problems each of which demonstrates a certain aspect of this thesis. We start with the introductory example 4.1.1 from the chapter on optimal control. We compare the Lagrange-Newton and quasi-Newton methods by the performance to solve this optimal control problem numerically, followed by a discussion on the advantages of both algorithms. The second example illustrates the usage of the regional penalization objective from definition 4.1.4. We give numerical solutions for a couple of penalty parameters of the control showing the influence on the deformation of the solution. The third problem is the example 4.1.8 on heliotropism, the movement of flowers towards the sun. Here we use cylindrical coordinates to describe a simple model of the stem below the flower. The fourth example will be on the enclosed volume by a deformed plate and its reference configuration. We try to maximize the volume, though we add a penalty term for the control in order to bound the optimization problem. We will then vary the penalty parameter for the control to show the influence on the enclosed volume. This example also demonstrates an optimal control problem on the basis of the hierarchical plate model.



**Remark 6.0.2** (Settings for the forward solver (algorithm 3.1.1)).

- $\text{rTol} = 10^{-8}$  for the stopping criterion (3.3) for Newton's method.
- inexact Newton with  $\text{rTol} = \min \{10^{-4}, \|W_{,\mathcal{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*}\}$  for the CG method
- nested iterations from algorithm 3.5.6
- preconditioner with the current stiffness matrix  $A_h$ , see section 5.4
- guiding criterion from algorithm 3.2.6 with  $\beta = 0.1$

**Remark 6.0.3** (Settings for the Lagrange-Newton method (algorithm 4.2.2)).

- $\text{rTol} = 10^{-6}$  for the stopping criterion (4.23) for the Lagrange-Newton method.
- inexact Newton with  $\text{rTol} = \min \{10^{-4}, \|W_{,\mathcal{U}}(\mathbf{U}_k, \mathbf{C})\|_{\mathcal{U}^*}\}$  for the MINRES method
- nested iterations from algorithm 4.2.3
- block preconditioner with the current stiffness matrix  $A_h$  and the mass matrix  $M_C$ , see section 5.4
- guiding criterion from algorithm 3.2.6 with  $\beta = 10^{-2}$

**Remark 6.0.4** (Settings for the quasi-Newton method (algorithm 4.3.15)).

- $\text{rTol} = 10^{-5}$  for the stopping criterion (4.46) for the quasi-Newton method
- $\text{rTol} = 10^{-6}$  for the stopping criterion for the linear solver
- guiding criterion from algorithm 3.2.6 for the forward solver with  $\beta = 0.1$
- preconditioner with the current stiffness matrix  $A_h$ , see section 5.4
- BFGS history maximum  $m = 50$ , initial scaling  $s_C = 1$
- Wolfe-Powell line search parameters  $c_1 = 10^{-3}$ ,  $c_2 = 0.9$

**Remark 6.0.5** (Standard material parameters).

If not otherwise stated, we choose for the material parameters a Young's modulus of  $E = 1 \text{ N mm}^{-2}$  and a Poisson's ratio of  $\nu = 0.3$ , that are the Lamé parameters

$$\mu = \frac{E}{2(1+\nu)} = \frac{5}{13} \frac{\text{N}}{\text{mm}^2}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} = \frac{15}{26} \frac{\text{N}}{\text{mm}^2}.$$

This gives the material parameters

$$a = \frac{25}{208} \frac{\text{N}}{\text{mm}^2}, \quad b = \frac{15}{208} \frac{\text{N}}{\text{mm}^2}, \quad c = \frac{15}{208} \frac{\text{N}}{\text{mm}^2}, \quad d = \frac{35}{52} \frac{\text{N}}{\text{mm}^2}, \quad e = -\frac{135}{208} \frac{\text{N}}{\text{mm}^2},$$

for the polyconvex energy density (2.18). The choice of the Young's modulus  $E$  is not essential to our numerical experiments because eventually  $E$  is a scaling of the



control  $C$ , because  $E$  enters the polyconvex energy linearly just like the control  $C$ . The Poisson's ratio of  $\nu = 0.3$  is a simple example for a compressible material.

All computations were done in FENICS 1.2.0 on an Intel Xeon Dual Core CPU with 16 GB RAM. If not stated otherwise, we use the following settings for the algorithms.

## 6.1 Experiments on the Forward Problem

### 6.1.1. Line Search and Iterates of Newton's Method

As we already announced in section 3.2, we compare the presented globalization strategies. Of course, the behavior of these techniques depends on the example itself and there are situations where one strategy works better than the other while in another example it might fail. We used example 3.2.1 earlier to show that a standard Newton's method with fixed step length  $\alpha = 1$  might diverge. Now, we solve this problem with Newton's method and with different globalization strategies:

INC is the incremental method from algorithm 3.2.3 with a simple equidistant distribution for the scalings  $s_k = \frac{k}{n}$ , e.g.  $0, \frac{1}{n}, \frac{2}{n}, \dots, \frac{n}{n}$  with  $n + 1$  steps in total. The number of steps is chosen as the smallest number where Newton's method converges for  $s_1 = \frac{1}{n}$ . We solve only the last increment  $s_N = 1$  accurately, the increments before have a larger relative tolerance of  $\text{rTol}_2 = 10^{-4}$ .

SIB is the simple backtracking algorithm 3.2.4.

ARM is the Armijo backtracking from algorithm 3.2.5 on the stored energy  $W$ . The parameter inside the criterion is chosen to be  $\sigma = 0.1$ .

GUI is the backtracking algorithm 3.2.6 with the guiding function  $G$  and the parameter  $\beta = 0.1$ .

In all backtracking line search methods (SIB, ARM, GUI) we choose the factor to decrease the step length to be  $s_\alpha = 0.9$ .

We expect the solution  $U$  to be symmetric, thus we exploit this symmetry by working only on a quarter of  $\Omega$ , e.g.  $(0, 1) \times (0, 1) \times (0, \frac{1}{10})$  and set symmetric Dirichlet BCs on the new boundary, e.g.

$$\begin{aligned} [U]_1 &= 0 & \text{on } \Gamma_1^D &= 1 \times [0, 1] \times [0, \frac{1}{10}] \\ [U]_2 &= 0 & \text{on } \Gamma_2^D &= [0, 1] \times 1 \times [0, \frac{1}{10}]. \end{aligned}$$

The domain  $\Omega$  is discretized by a structured mesh with  $11 \times 11 \times 3$  vertices. The mesh is uniformly refined twice, i.e.  $L = 2$ . We do not use nested iterations here because the later usage of the globalization strategies *inside* an optimization algorithm would



not allow that.<sup>1</sup> The results of the line search test can be found in table 6.1 and a plot of a few iterates are shown in figure 6.1

strategy	Newton iter.	Krylov iter.	total time in s	remarks
INC	22	255	52	
SIB ( $s_\alpha = 0.9$ )	-	-	-	no convergence
SIB ( $s_\alpha = 0.5$ )	17	155	33	6× neg. curv. in the CG
ARM	6	76	15	
GUI	6	78	15	

TABLE 6.1. Numerical results for testing the globalization strategies from section 3.2. The second and third columns refer to the total number of iterations for Newton’s method and the truncated CG method. The time is the total running time of the program.

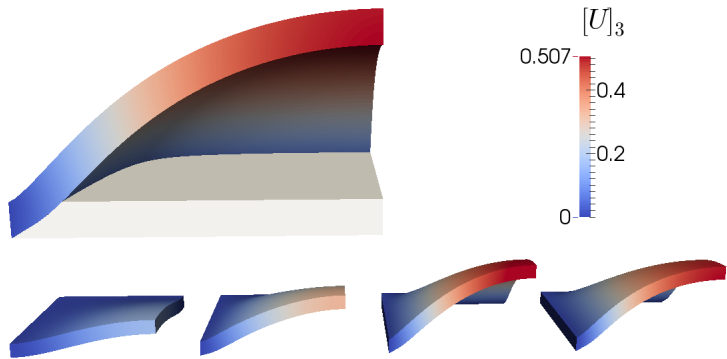


FIGURE 6.1. The upper plot shows the solution of the example 3.2.1. The lower plots show the first four iterates of Newton’s method with GUI. The last two iterates (the 5th and 6th) are not visually distinguishable from the fourth iterate.

Conclusions

<sup>1</sup>There, a nested iteration would be applied on the optimization method itself, not on the sub-routines of it.



In the example 3.2.1, local negative volume change  $J < 0$  was the main reason for divergence of a standard Newton's method. The results clearly show the necessity for a globalization strategy. We are looking for a method to deal with large deformations, i.e. it can reliably find a solution.

The incremental method does not meet our demands as it is difficult for it to adapt itself to a given control. Fixing the number of increments is not an option for the optimal control problem and a trial and error approach seems very costly. Therefore, we will not use this strategy.

The simple backtracking SIB behaves rather unpredictably. The only user-defined parameter in SIB is the factor  $s_\alpha$  which controls the new trial step length  $\alpha := s_\alpha \alpha$ . This however is not sufficient to control the line search to behave desirably. Larger values of  $s_\alpha$  might cause the method to accept step lengths that produce updates with a locally very small volume change  $J \approx 0$ . This produces bad starting points and Newton's method diverges. If this is the case, we could set  $s_\alpha$  smaller in the hope that the iterates stay away from  $J \approx 0$ . However, this might result in very small  $s_\alpha$  so Newton's method is too strongly damped. As a conclusion, we find SIB to be unsatisfactory for large deformation.

The Armijo backtracking line search and the guiding criterion behave very similarly. Therefore, we increase the volume load to  $\mathbf{f} = (0, 0, 1)^\top$ , so that the problem becomes more difficult to solve, see the results in table 6.2. GUI seems to offer a better criterion than ARM for Newton's method to solve example 3.2.1.

strategy	Newton iter.	Krylov iter.	total time in s
ARM	10	109	24
GUI	7	106	18

TABLE 6.2. Numerical results similar to table 6.1, but with an increased volume load  $\mathbf{f} = (0, 0, 1)^\top$ .

As a final conclusion, the Armijo backtracking and the guiding criterion are both reliable line search methods for the globalization of Newton's method. The Armijo backtracking has the advantage that it arises naturally in Newton's method for the energy minimization. Thus it fits in the known and well studied context of line search methods in nonlinear optimization, with the theoretical foundations about existence of an admissible step length and sufficient decrease of the energy to ensure convergence of Newton's method.

The guiding criterion on the other hand, does not consider the energy minimization but rather is a special damping strategy to avoid certain unfavorable step lengths.



In particular, this means local negative volume change in elasticity models for large deformations. Even without the theoretical basis, numerical experiments suggest that the guiding criterion is a reliable strategy.

So far, the guiding function  $G$  from (3.5), that was

$$G(\mathbf{U}) = \begin{cases} \int_{\Omega} |\ln(\det(I + \nabla \mathbf{U}(\mathbf{x})))| \, d\mathbf{x} \\ \infty & \text{if the integral is not defined} \end{cases}$$

has only considered the local volume change  $J$ . However, if the volume change is not a good criterion to judge a step length, it is possible to add other terms like  $\|F\|_F^2$  or  $\|\text{cof } F\|_F^2$  to the guiding function  $G$ . This could help to prevent too large Newton steps and therefore serve as a globalization strategy for Newton's method.

### 6.1.2. Plate Model

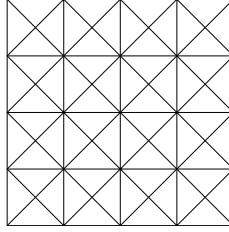
We reconsider the thick plate from example 3.2.1 and apply the hierarchical plate model to it. First we will compare the solutions from the standard full 3D model and the hierarchical plate model for a few degrees  $D^{2D}$ . We note that such comparisons should be viewed carefully since a 3D model differs from a 2D model. However, the hierarchical plate model can be understood as imposing a certain structure on the 3D displacement  $\mathbf{U} : \Omega \rightarrow \mathbb{R}^3$ . This structure is defined by functions  $\mathbf{U}_i^{2D} : \Omega^{2D} \rightarrow \mathbb{R}^3$  which happen to be two-dimensional. Still, the displacement  $\mathbf{U}$  with the hierarchical ansatz is a vector field defined on  $\Omega$ . This is the reason why we can plug the hierarchical ansatz  $\mathbf{U} = \sum_i p \mathbf{U}_i^{2D}$  into the 3D model in the first place.

The forward problem was formulated as an energy minimization problem “min  $W$ ” in this work. It seems natural to compare solutions from different discretizations by the value of their stored energy  $W$ . This allows us not only to compare different meshes and degrees  $D^{2D}$  of the hierarchical plate model, but also to compare the full 3D model with the plate model.

We triangulate the midsurface by the coarse mesh shown in figure 6.2. To have a fair comparison with the full 3D model, we do not choose boundary layers near the rim  $\partial\Omega^{2D}$  in order to improve the quality of the discretization. On the base of other numerical experiments, the degree of the Gauss-Legendre quadrature is chosen to be  $D^{GL} = D^{2D} + 1$ .

We note that the deformations in this example can be considered “large”, as seen in figure 6.1. A plot of the stored energy  $W$  of the solutions for the full 3D model and the hierarchical plate model up to degree 4 can be found in figure 6.3. First, we can see that each plate degree tends to converge to a certain value of the stored energy  $W$ . This suggests that the discretization of the midsurface  $\Omega^{2D}$  converges and that there exists a lower bound for  $W$  for each degree  $D^{2D}$ . This is exactly the behavior one would expect if the hierarchical plate model is viewed as a reduction



FIGURE 6.2. The coarse mesh of the midsurface  $\Omega^{2D}$ .

technique: by an increase in the degree  $D^{2D}$  of the model, the approximation of the full model improves. The figure also suggests that there is only a small difference between degrees 3 and 4 in this example. This means that the structure of the deformation is well resolved with the degree  $D^{2D} = 3$ . In comparison, the full 3D model needs many more degrees of freedom to achieve similar values of the stored energy  $W$  than the plate model with degree  $D^{2D} = 3$ . This shows the advantage of the hierarchical plate model over the full 3D model even for plates with a thickness of  $2d = 0.1$ .

### Thin Plates

Next, we will shrink the thickness  $2d$  to get an idea which degree  $D^{2D}$  is actually needed for a good approximation of a thin plate. The volume load  $\mathbf{f} = (0, 0, \frac{1}{4})^\top$  stays the same during this test, that means the total force acting on the body scales with the thickness  $2d$ . With shrinking thickness, the plate loses its bending stiffness and behaves more and more like a membrane. Figure 6.4 shows a comparison of the degrees  $D^{2D}$  of the plate model for the thickness  $2d = 0.01$ . We can see that a degree of  $D^{2D} = 1$  already yields a good approximation and additional degrees offer only a minor improvement. The full 3D model behaves even worse for thin plates. The value of the stored energy of the full 3D model is always above the values of similar levels of the plate model. The effect will become even worse for thinner plates and is often called *locking*. For thin plates, this means that the discretized elasticity model behaves “stiffer” as the thickness decreases.

So far, we discussed the quality of the discretization with respect to the degree  $D^{2D}$ . However, computational costs are important for practical problems as well. In order to compare the different models, we regard the computational time *per* number of degrees of freedom (dofs). Figure 6.5 shows this quantity with respect to the thickness  $2d$ . First, the computational costs increase with the degree  $D^{2D}$  of the plate model. This is mostly due to the fact that more CG iterations are necessary to solve the linear system in Newton’s method. This suggests that the preconditioner depends on the degree  $D^{2D}$ , that means on the degree of the Legendre polynomials



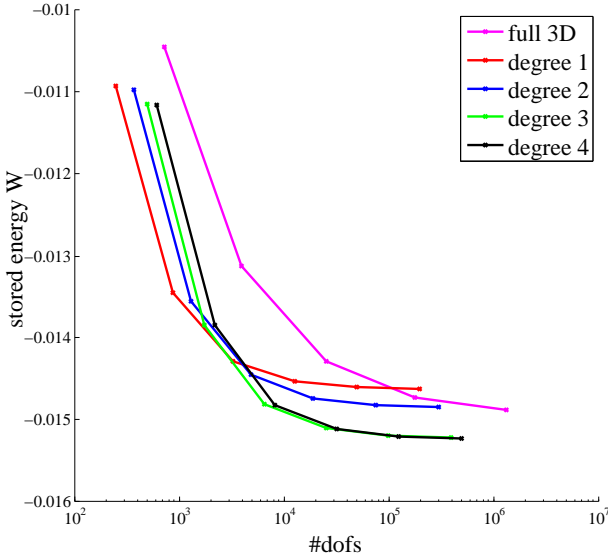


FIGURE 6.3. The stored energy  $W$  of the solutions of example 3.2.1 for the full 3D model and the hierarchical plate model with the degrees  $D^{2D} \in \{1, 2, 3, 4\}$ , plotted against the number of degrees of freedom  $N^{\mathcal{U}}$ .

$p_i$ . This dependency of the multigrid method on the polynomial degree of the ansatz functions is also observed in other discretizations, for example NURBS, see Gahala et al. (2013). In their work, the iteration numbers for a full V-cycle multigrid method on a Poisson problem increases with the polynomial degree of the NURBS, though they show independence with respect to the mesh size.

In the results of figure 6.5, the plate degrees  $D^{2D} = 1, 2$  show a mesh independent behavior of the overall running time with respect to the degrees of freedom. The multigrid V-cycle might not be such a good approximation for the higher degrees  $D^{2D} = 3, 4$  and their results do not show the mesh independency as clearly as lower degrees. Though the results do not suggest that their running time per number of dofs continues to grow like in the case of the full 3D model. Here, the low computational costs grow with decreasing thickness of the plate.

## Conclusions

As we have seen in figure 6.4, a hierarchical plate degree of  $D^{2D} = 1$  already gives very good results for a plate where the ratio of thickness and a characteristic length



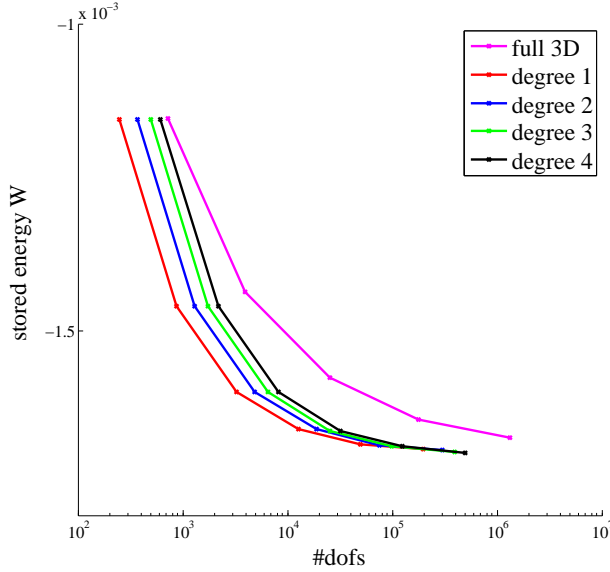


FIGURE 6.4. The stored energy  $W$  of the solutions of example 3.2.1 with the thickness  $2d = 0.01$ , for the full 3D model and the plate model, plotted against the number of degrees of freedom  $N^{\mathcal{U}}$ .

(e.g. the length or width) is about  $\frac{1}{100}$ . Higher degrees  $D^{2D} > 1$  are barely needed in this example and the higher computational costs would not justify them. Moreover, we can see that the full 3D-model is not practical for thin plates with respect to quality of the discretization and the computational costs.

## 6.2 Experiments on the Optimal Control Problem

### 6.2.1. Elevated Bar with Fiber Tension Control

In section 4.1 we introduced an optimal control problem with the example 4.1.1 of a bar whose right part is elevated, which we are going to solve with a Lagrange-Newton method (LN) and a quasi-Newton method (QN). As a load, we choose the fiber tension  $m$ , so one could imagine the movement of a tongue behind the example 4.1.1. As a fiber direction, we set  $\mathbf{a}(\mathbf{x}) = (1, 0, 0)^\top$ , that means the fibres are aligned to the length of the bar. The load can only be applied in the left part  $\Omega^{\text{left}}$  and the



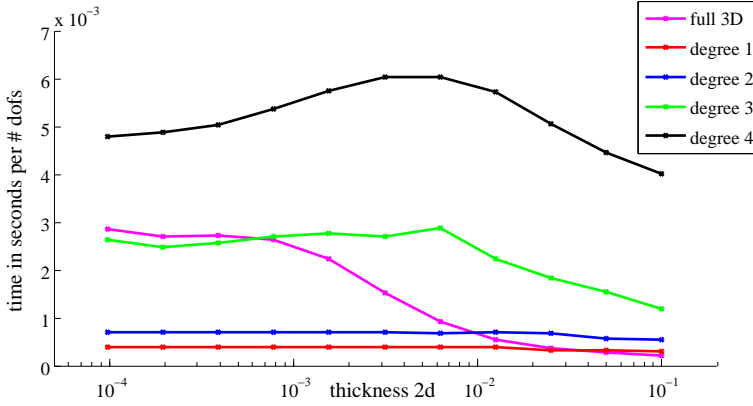


FIGURE 6.5. Computational costs per number of degrees of freedom for different degrees of the plate model and the full 3D model, with respect to the thickness  $2d$  ( $L = 3$  total mesh refinements).

penalty term is

$$P^{(m)}(m) := \frac{1}{10} \int_{\Omega^{\text{left}}} (m(\mathbf{x}))^2 d\mathbf{x}.$$

The quality term  $Q$  will be a standard tracking type

$$Q^{\text{track}}(\mathbf{U}) := \frac{1}{2} \int_{\Omega} \gamma(\mathbf{x}) \|\mathbf{U}(\mathbf{x}) - \mathbf{U}^{\text{des}}(\mathbf{x})\|_2^2 d\mathbf{x},$$

with the desired displacement

$$\mathbf{U}^{\text{des}}(\mathbf{x}) = (0, 0, \frac{1}{5})^\top,$$

and the weight function

$$\gamma(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega^{\text{right}} \\ 0 & \text{else} \end{cases}.$$

We discretize the domain  $\Omega = (0, 2) \times (0, \frac{1}{5}) \times (0, \frac{1}{10})$  by a coarse mesh with  $21 \times 3 \times 3$  vertices. We use the standard settings from remarks 6.0.3 and 6.0.4 for the LN and QN algorithm, except for the relative tolerance  $\text{rTol} = 10^{-6}$  for the stopping criterion of the QN method. Additionally we set the scaling factors of the block preconditioner in the Lagrange-Newton method to  $s_U = 1$ ,  $s_C = 10$ ,  $s_Z = 10^{-3}$  and the initial scaling of the BFGS update in the quasi-Newton method to  $s_C = 10$ .



All three methods converged visually to the same solution, plotted in figure 6.6. The numerical results and the convergence of the residual norm can be found in

- table 6.3 and figure 6.7 for the Lagrange-Newton method (LN),
- table 6.4 and figure 6.8 for the Lagrange-Newton method with nested iterations (LNnest),
- table 6.5 and figure 6.9 for the quasi-Newton method (QN).

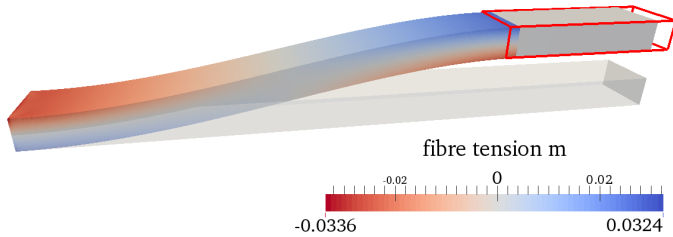


FIGURE 6.6. The solution to the optimal control problem from 4.1.1 on the finest mesh level  $L = 4$ . The red box illustrates the desired height  $h = 0.2$ , where the red right part  $\Omega^{\text{right}}$  has to be elevated to.

total #dofs ( $U, m, Z$ )	obj. value $I^* \cdot 10^7$		total	total	time in %
		LN-it.	Krylov it.	time in s	ass. / solve
1 614	5.63837	6	317	1	79 / 21
9 990	2.26274	7	2524	26	38 / 62
70 086	1.43993	8	3455	285	32 / 68
524 934	1.24786	9	4146	2921	28 / 72
4 063 494	1.19886	9	4238	24467	28 / 72

TABLE 6.3. [LN] Numerical results for the Lagrange-Newton method: The total number of degrees of freedom refer to the unknown  $X_h = (U_h, m_h, Z_h)$ . The optimal objective function value is given by  $I^*$ . The total number of iterations in the LN algorithm and in the MINRES method are found in the third and fourth columns. The total time is the running time of the program and the last column states the proportions of the times for assembly and solving.



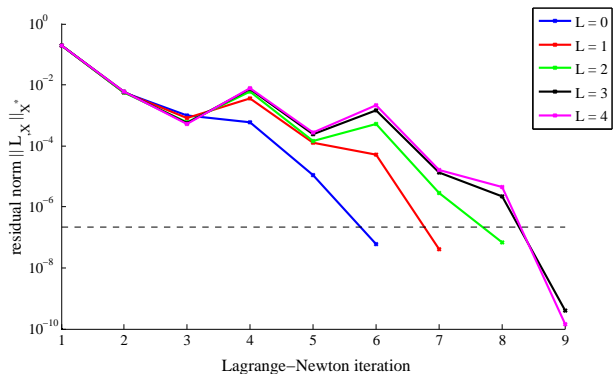


FIGURE 6.7. [LN] Norm of the residual  $\|L_X\|_{X^*}$  over the Lagrange-Newton iterations for the mesh refinement levels  $L = 0, 1, \dots, 4$ . The dotted line is the value for the relative stopping criterion.

total #dofs ( $U, m, Z$ )	obj. value $I^* \cdot 10^7$	total LN-it.	total Krylov it.	total time in s	time in % ass. / solve
1 614	5.63837	6	317	1	78 / 22
9 990	2.26279	+3	+1101	13	42 / 58
70 086	1.43993	+3	+1273	120	34 / 66
524 934	1.24786	+2	+ 866	756	32 / 68
4 063 494	1.19886	+2	+ 856	5963	31 / 69

TABLE 6.4. [LNnest] Numerical results for the Lagrange-Newton method with nested iterations: The notation is the same as in table 6.3. The numbers of new iterations in the LNnest algorithm and in the MINRES method on a level are found in the third and fourth columns.

Interpretations of the results

First of all, we notice that the optimal objective value  $I^*$  is quite the same for all three methods. Minor differences could be diminished by lowering the relative tolerance rTol in the stopping criteria of the LN, LNnest and QN methods.

We can see that the Lagrange-Newton (LN) method without the nested iterations shows a mesh independent behavior, that means the number of LN iterations is bounded from above. Even more, the number of iterations of the MINRES method



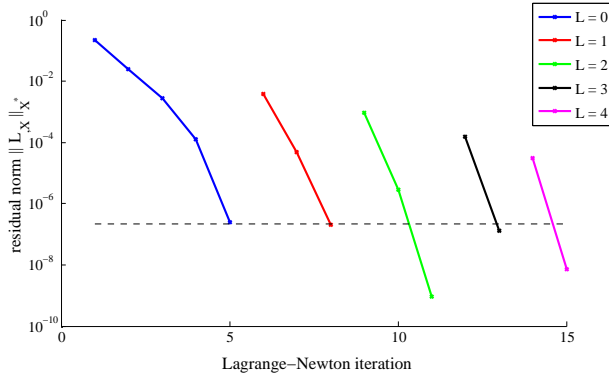


FIGURE 6.8. [LNnest] Norm of the residual  $\|L_{X^*}\|_{X^*}$  over the Lagrange-Newton iterations for the mesh refinement levels  $L = 0, 1, \dots, 4$  and with nested iterations. The dotted line is the value for the relative stopping criterion.

state #dofs $\mathbf{U}$	obj. value $I^* \cdot 10^7$	total QN-it.	total Krylov it.	total time in s	time in % ass. / solve
567	5.63838	24	79	1	63 / 26
3 075	2.26279	27	1846	11	76 / 18
19 683	1.43993	23	1429	78	66 / 32
139 587	1.24787	17	1564	750	67 / 32
1 048 707	1.19887	19	1327	4986	62 / 35

TABLE 6.5. [QN] Numerical results for the quasi-Newton method: The notation is the same as in table 6.3. Since the forward and adjoint systems have to be solve in  $\mathcal{U}_h$ , only the degrees of freedom of  $\mathbf{U}_h$  matter.

seems bounded too. In this example, one LN step needed around 400-500 MINRES iterations on average, which shows that the block preconditioner might not work that well for the example. However, the computational work can be reduced to a quarter by using nested iterations (LNnest). Here we can observe that the number of LNnest iterations on the new level is much smaller compared to LN, it even tends to decrease with the growing levels.

The quasi-Newton method (QN) shows a mesh independent behavior too. The CG iterations are the total number of Krylov iterations during the QN method, that is



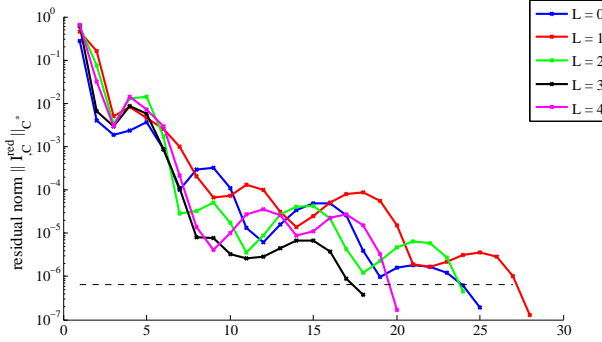


FIGURE 6.9. [QN] Norm of the residual  $\|I_{\mathcal{C}}^{\text{red}}\|_{\mathcal{C}^*}$  over the quasi-Newton iterations for the mesh refinement levels  $L = 0, 1, \dots, 4$ . The dotted line is the value for the relative stopping criterion on the level  $L = 4$ .

the truncated CG iterations in the forward Newton solver and the CG iterations for the adjoint solver. One QN step needed around 60-70 CG iterations on average in this example. Most of them are used in the beginning of the QN method because here the Wolfe-Powell line search rejects a few step lengths, which requires the forward problem to be solved multiple times. In the end, when the solution converges towards the solution, the Wolfe-Powell line search is likely to accept the step length  $\alpha = 1$  and the forward system is normally solved in one or two Newton steps. Furthermore, the CG method for the adjoint equation is faster if we start with the previous solution of the adjoint state  $\mathbf{Z}$  as an initial guess for the adjoint equation. Therefore one can notice that the last steps take less computational time than the beginning, see figure 6.10. However, as the quasi-Newton method with LM-BFGS update offers at best q-superlinear local convergence, which itself depends on the accuracy of the CG methods, we cannot expect that a decrease in the relative tolerance rTol for the quasi-Newton method comes at only little additional computational costs.

The costs for the BFGS update are less than 0.5% of the total run time, including the update itself and the handling of the storage of the auxiliary BFGS vectors. This allows to set the maximum number of BFGS vectors quite high, since it is very cheap compared to the other computational costs. However, the BFGS history makes it difficult to apply nested iterations. On the new level, we could discard the history and just start at a better initial guess, but this advantage turns out to be not beneficial. It takes a couple of iterations for a good BFGS approximation of the reduced (inverse) Hessian matrix, see table 6.6. It would be certainly better to use the information of the BFGS history on the old level. A straightforward idea could



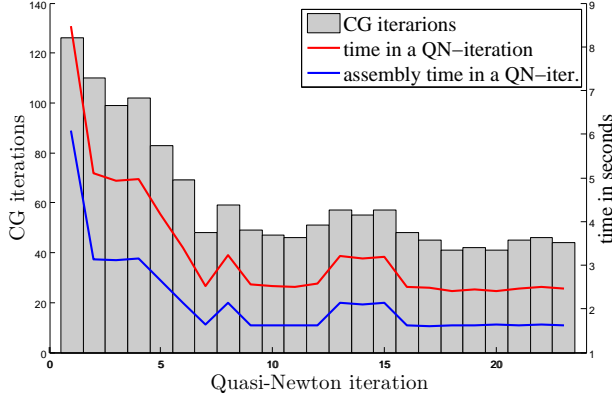


FIGURE 6.10. Computational work load over the iterations of the quasi-Newton method for the mesh level  $\ell = 2$ . The gray bars are the number of CG iterations in a QN iteration while the red and blue lines indicate the total time and time for assembly in a QN iteration.

be to interpolate the auxiliary BFGS vectors  $(\tilde{s}_i^\ell)_i$  and  $(\tilde{y}_i^\ell)_i$  from level  $\ell$  to the new level  $\ell + 1$ . This is straightforward for the increments  $(\tilde{s}_i^\ell)_i$  because they represent elements of a discretized control space, e. g.

$$\tilde{s}_i^{\ell+1} = p_{\ell \rightarrow \ell+1} \tilde{s}_i^\ell.$$

This would not work for the increments  $(\tilde{y}_i^\ell)_i$ . Please recall that  $(\tilde{y}_i^\ell)_i$  are discretizations of elements of the dual space  $\mathcal{C}^*$ . We would need to *pull-back*  $(\tilde{y}_i^\ell)_i$  to the primal space  $\mathcal{C}$ , i. e.  $(M_C^\ell)^{-1} \tilde{y}_i^\ell$ , prolongate, i. e.  $p_{\ell \rightarrow \ell+1} M_C^{-1} \tilde{y}_i^\ell$ , and then *push-forward* to the dual space  $\mathcal{C}^*$  again, that is

$$\tilde{y}_i^{\ell+1} = M_C^{\ell+1} p_{\ell \rightarrow \ell+1} M_C^{-1} \tilde{y}_i^\ell.$$

However, the prolonged increments  $(\tilde{y}_i^\ell)_i$  do not work well in the quasi-Newton method, as the number of iterations of the new level are not lower, see table 6.6. This is caused by the fact that the prolonged  $(\tilde{y}_i^{\ell+1})_i$  are not the increments of the derivatives belonging to the increments  $(\tilde{s}_i^{\ell+1})_i$ , see figure 6.11. However, this is a requirement for the BFGS update to work properly. Computing the derivatives to build  $(\tilde{y}_i^{\ell+1})_i$  would require to restore the iterates  $\mathbf{C}_k$  from the iterates  $(\tilde{s}_i^{\ell+1})_i$  and solving the forward problem and adjoint equation for each iterate  $\mathbf{C}_k$  on the new level  $\ell + 1$ . These are the same computational costs as a normal quasi-Newton method started on the level  $\ell + 1$ , but without the line search. Considering that we have to continue in the QN on the new level, this is hardly an advantage compared



to the QN without nested iterations. Furthermore, the fact that these new iterates were not checked by the Wolfe-Powell line search reveals another possible problem: the new BFGS history  $(\bar{s}_i^{\ell+1})_i$  and  $(\bar{y}_i^{\ell+1})_i$  might not satisfy the curvature condition  $(\bar{y}_i^{\ell+1})^\top \bar{s}_i^{\ell+1} > 0$  which guarantees the positive definiteness of the BFGS matrix  $H_k$ . Therefore, using the information of the BFGS history on the old level  $\ell$  to build a history on the new level  $\ell + 1$  is not straightforward.

level	discarding BFGS history (QN-iterations)	interpolating BFGS history (QN-iterations)	no nested iterations
$\ell = 0$	24	24	24
$\ell = 1$	+33	+42	27
$\ell = 2$	+25	+29	23
$\ell = 3$	+25	+44	17
$\ell = 4$	+19	+36	19

TABLE 6.6. Iteration numbers of the quasi-Newton method with nested iterations. Two methods were tested: discarding the BFGS history from the previous level and building it anew, or interpolating the BFGS history from the previous level. (The initial residual norm  $\|I_{\mathcal{C}}^{\text{red}}(\mathbf{C}_0)\|_{\mathcal{C}^*}$  on the coarse mesh level  $\ell = 0$  was used for the stopping criteria on the other levels as well, in order to have the same stopping criteria across all levels.)

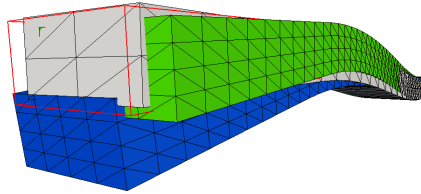


FIGURE 6.11. Nested iterations in the quasi-Newton method by interpolating the BFGS history: The gray mesh shows the displacement of the solution of the optimal control problem on mesh level  $\ell = 0$ . The displacements of the first (blue) and second (green) quasi-Newton iteration on the mesh level  $\ell = 1$  show a left or right drift, which is an artifact from the interpolation from the coarse mesh.



We conclude the following: both methods, nested Lagrange-Newton and quasi-Newton, are quite comparable to the overall runtime of the optimization. However, most of the time in the LN is used for the solution of the linear system, mostly because the block preconditioner is not as good as the preconditioner for the forward problem. On the other hand, the large part of the time in the QN method is spent for assembly, mostly for the stiffness matrix  $K$  in the forward solver. Another difference is the memory usage of both algorithms. The LN needed about three times as much working memory than the QN. The observed main differences between both methods are shown in table 6.7.

Lagrange-Newton	quasi-Newton
requires $W_{,FFF}$ for $L_{,XX}$	approximation of the reduced Hessian $I_{,CC}^{\text{red}}$ by a BFGS update formula
“only” block preconditioner	“established” preconditioner
nested iterations work well	how to interpolate the BFGS history?
high working memory load	lower memory load
rTol of the MINRES method determined by inexact Newton	accuracy in CG method is needed for the Wolfe-Powell line search

TABLE 6.7. Comparison of the Lagrange-Newton and quasi-Newton method, based on the numerical results from section 6.2.1

### 6.2.2. Regional Penalization

A case where a desired state  $\mathbf{U}^{\text{des}}$  is not available is given in the next example 6.2.1.

**Example 6.2.1** (Bar and Plane). Let  $\Omega = (0, 5) \times (0, 1) \times (0, 1)$  be a bar that is clamped at the left boundary  $\Gamma^{\text{D}} := \{0\} \times (0, 1) \times (0, 1)$ . The control is the fiber tension  $m$  along the fiber direction  $\mathbf{a} = (1, 0, 0)^{\top}$ . We seek a control  $m$  such that the deformed body avoids the half plane

$$H := \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x}_1 - \mathbf{x}_3 - 2 \geq 0\}.$$

We penalize this region by the quality functional

$$Q^{\text{pen}}(\mathbf{U}) = \int_{\Omega} q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \det F(\mathbf{x}) \, d\mathbf{x},$$



with the penalization function

$$q(\mathbf{x}) = [\mathbf{x}_1 - \mathbf{x}_3 - 2]_\varepsilon^+,$$

where  $[\cdot]_\varepsilon^+$  is the (smoothed) positive part

$$[x]_\varepsilon^+ := \frac{1}{2} \left( \sqrt{x^2 + \varepsilon^2} + x \right).$$

We use theorem 4.1.5 to provide the first derivative

$$Q_{,U}^{\text{pen}}(\mathbf{U})[\delta \mathbf{U}] = \int_{\Gamma} q(\mathbf{x} + \mathbf{U}(\mathbf{x})) \delta \mathbf{U}(\mathbf{x})^\top F(\mathbf{x})^{-\top} \mathbf{n}(\mathbf{x}) \det F(\mathbf{x}) \, dS.$$

Figure 6.12 shows the undeformed body  $\Omega$  and the half plane  $H$ .

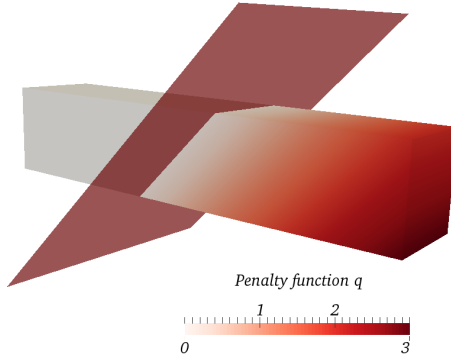


FIGURE 6.12. The undeformed body  $\Omega$  and the half plane  $H$  from example 6.2.1. The colored scaling indicates the value of the penalization function  $q$ .

We solve this problem with a coarse mesh of  $11 \times 3 \times 3$  vertices and  $L = 3$  mesh refinements, yielding 70227 degrees of freedom of the state  $\mathbf{U}$  on the finest mesh. We choose the smoothing parameter  $\varepsilon = 0.01$  and vary the penalty parameter  $\gamma^{(m)}$  of the cost functional

$$P^{(m)}(m) := \gamma^{(m)} \int_{\Omega} (m(\mathbf{x}))^2 \, d\mathbf{x},$$

to illustrate the dependence of the solution on the parameter  $\gamma^{(m)}$ . We choose the quasi-Newton algorithm with the standard settings from remark 6.0.4. The solutions are plotted in figure 6.13.



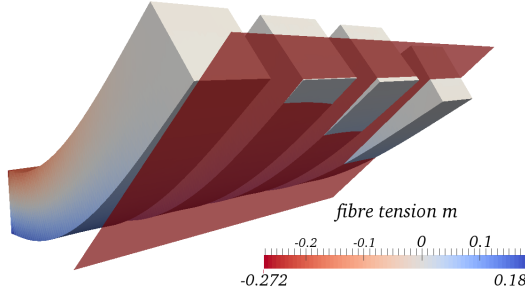


FIGURE 6.13. Deformations of the solutions of the optimal control problem with regional penalization for the penalty parameters  $\gamma^{(m)} = 10^{-1}, 1, 10, 10^2$ , from left to right. The red plane is the boundary of the half plane  $H$ .

Figure 6.14 shows the dependency of the functional value  $Q$  of the solution with respect to the penalty parameter  $\gamma^{(m)}$ . For smaller values of  $\gamma^{(m)}$ , the regional penalization  $Q$  tends to zero while the penalty term  $P$  slightly increases. Please note that the optimal control problem would be unbounded for  $\gamma^{(m)} = 0$  because of the smoothed positive part  $[\cdot]_{\varepsilon}^{+}$ . On the other hand, for large values of  $\gamma^{(m)}$ , the penalty term  $P$  tends to zero while the regional penalization  $Q$  converges to  $\approx 3.16700$  which is the value for the undeformed body, that is  $Q(\mathbf{0}) \approx 3.16700$ .

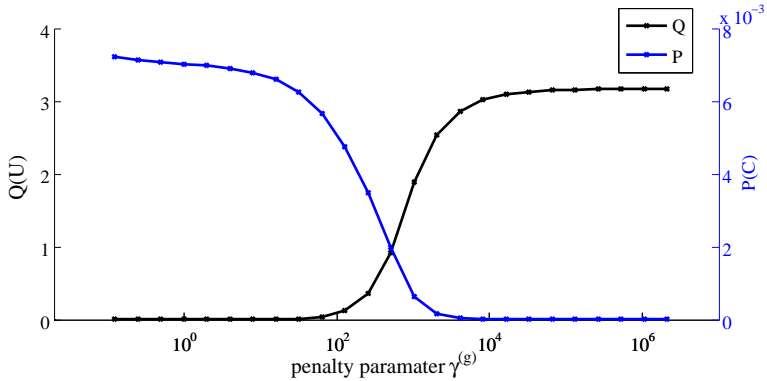


FIGURE 6.14. Values of the quality function  $Q$  of the state and the penalty function  $P$  for the control of the solution of the optimal control problem with respect to the penalty parameter  $\gamma^{(m)}$ .



Remark 4.1.7 already discussed possible problems in the numerical algorithm due to numerical integration in the quality term  $Q$ . The smoothing of the positive part is important for the behavior of the numerical algorithm. For example, setting  $\varepsilon = 0$  yields the standard positive part function  $[x]^+ := \frac{1}{2}(|x| + x)$ . We demonstrate this by a sequence  $\varepsilon \in \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$  for the smoothing parameter in  $[x]_\varepsilon^+ := \frac{1}{2}(\sqrt{x^2 + \varepsilon^2} + x)$ . Table 6.8 shows the results. The failures in the Wolfe-Powell line search are caused by errors in the quality term  $Q^{\text{pen}}$  due to the numerical integration.

$\varepsilon$	QN-it	comments
$10^{-4}$	10	stopping criterion satisfied
$10^{-5}$	15	stopping criterion satisfied
$10^{-6}$	(10)	failure in the Wolfe-Powell line search
$10^{-7}$	(12)	failure in the Wolfe-Powell line search
$10^{-8}$	(11)	failure in the Wolfe-Powell line search

TABLE 6.8. Convergence results for the regional penalization with varying smoothing parameter  $\varepsilon$  in  $[x]_\varepsilon^+ := \frac{1}{2}(\sqrt{x^2 + \varepsilon^2} + x)$ .

### 6.2.3. Flower Movement by Turgor Pressure

Example 4.1.8 introduced heliotropism: the movement of flowers towards the sun by a turgor pressure in the stem below the flower. This biological phenomenon motivated the quality term (4.9) of a desired direction. We will model an optimal control problem based on heliotropism in the following way.

**Example 6.2.2** (Part of a Flower Stem). Let us consider a part of a hollow stem which is the tube with an inner radius  $r_1 = 0.7$ , an outer radius  $r_2 = 1$  and length  $h = 5$ . We use cylindrical coordinates

$$\mathbf{x}(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \tilde{\mathbf{x}}_3) = \begin{pmatrix} \tilde{\mathbf{x}}_1 \cos \tilde{\mathbf{x}}_2 \\ \tilde{\mathbf{x}}_1 \sin \tilde{\mathbf{x}}_2 \\ \tilde{\mathbf{x}}_3 \end{pmatrix} \quad \tilde{\mathbf{x}} \in \tilde{\Omega} := (r_1, r_2) \times (0, 2\pi) \times (0, h),$$

to describe the reference configuration  $\Omega := \mathbf{x}(\tilde{\Omega})$ . There are Dirichlet boundary conditions applied to the bottom, that is  $\mathbf{x}_3 = 0$ , respectively  $\tilde{\Gamma}^D = (r_1, r_2) \times (0, 2\pi) \times \{0\}$ . The control is the inner pressure  $t$  from table 2.1 and we allow it to act between the heights  $\underline{h} = 0.5$  and  $\bar{h} = 4.5$ , that is the control domain  $\tilde{\Omega}^{\text{ctrl}} := (r_1, r_2) \times (0, 2\pi) \times (\underline{h}, \bar{h})$ . We seek to align the top part of the stem with a given direction  $\mathbf{s}^{\text{des}} \in \mathbb{R}^3$  which points towards the sun, see figure 6.15.



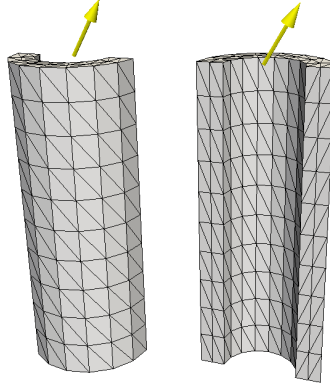


FIGURE 6.15. Front and back view of a triangulation of half of the hollow stem. The yellow arrow is the desired direction  $\mathbf{s}^{\text{des}}$  which points towards the sun. We seek a deformation where the top of the stem bends in the direction of  $\mathbf{s}^{\text{des}}$ .

To solve this problem, we exploit the rotational symmetry of the stem  $\Omega$  and rotate the coordinate system such that the direction  $\mathbf{s}^{\text{des}}$  lies in the  $\mathbf{x}_1\text{-}\mathbf{x}_3$ -plane. As we assume  $\mathbf{s}^{\text{des}}$  to be normalized, we can describe it uniquely by an angle, e.g.  $\mathbf{s}^{\text{des}} = (\sin \theta, 0, \cos \theta)^\top$  with  $\theta \in [0, \frac{\pi}{2}]$ . Additionally, as we expect a symmetric solution with respect to this plane, we exploit this mirror-symmetry. This means we consider only half of the tube

$$\Omega \cap \{\mathbf{x} \in \mathbb{R}^3 : x_2 > 0\}.$$

The corresponding parameter space is

$$\tilde{\Omega} = (r_1, r_2) \times (0, \pi) \times (0, h),$$

and the additional symmetry boundary conditions are

$$[\mathbf{U}]_2 = 0 \quad \text{on} \quad \Gamma^{\text{sym}} := (r_1, r_2) \times \{0, \pi\} \times (0, h).$$

We summarize the optimal control problem which we use to compute our example of the turgor pressure in a flower stem. In the following, the reference domain  $\Omega$  already includes the symmetry exploitation, that means  $\Omega$  is only half of the stem. The same applies for the parameter space  $\tilde{\Omega}$ .

- The finite elements are defined in the parameter space  $\tilde{\Omega} := (r_1, r_2) \times (0, \pi) \times (0, h)$ .



- The Dirichlet boundary conditions are

$$\begin{aligned} \mathbf{U} = \mathbf{0} \in \mathbb{R}^3 & \quad \text{on} \quad \Gamma^D := (r_1, r_2) \times (0, \pi) \times \{0\} \\ \text{and } [\mathbf{U}]_2 = 0 \in \mathbb{R} & \quad \text{on} \quad \Gamma^{\text{sym}} := (r_1, r_2) \times \{0, \pi\} \times (0, h). \end{aligned}$$

- The reference domain  $\Omega := \mathbf{x}(\tilde{\Omega})$  is parametrized by cylindrical coordinates with its Jacobian  $\tilde{G}$ , see (2.52). This influences the gradient operator  $\nabla$ , therefore the deformation gradient  $F$ , and the volume and area transformations.
- The control  $\mathbf{C}$  is the inner pressure  $t$  acting in  $\tilde{\Omega}^{\text{ctrl}} := (r_1, r_2) \times (0, \pi) \times (h, \bar{h})$ .
- The reference direction is  $\mathbf{s} := (0, 0, 1)^\top$  and the desired direction pointing to the sun is  $\mathbf{s}^{\text{des}} := (\sin \theta, 0, \cos \theta)^\top$  with  $\theta = \frac{\pi}{6}$ . They are used to formulate the objective function  $I = P + Q$ , with the penalty term

$$P(t) := \int_{\Omega} t(\mathbf{x})^2 d\mathbf{x} = \int_{\tilde{\Omega}} t(\mathbf{x}(\tilde{\mathbf{x}}))^2 |\det \tilde{G}(\tilde{\mathbf{x}})| d\tilde{\mathbf{x}},$$

and the quality term

$$\begin{aligned} Q(\mathbf{U}) &= \int_{\Gamma^{\text{top}}} \|F(\mathbf{x})\mathbf{s} - \mathbf{s}^{\text{des}}\|_2^2 dS \\ &= \int_{\tilde{\Gamma}^{\text{top}}} \|\tilde{F}(\tilde{\mathbf{x}})\mathbf{s} - \mathbf{s}^{\text{des}}\|_2^2 \underbrace{|\det \tilde{G}(\tilde{\mathbf{x}})| \|\tilde{G}^{-\top}(\tilde{\mathbf{x}})\tilde{\mathbf{n}}(\tilde{\mathbf{x}})\|_2}_{\text{from the area transformation}} d\tilde{S}, \end{aligned}$$

with the top surface  $\Gamma^{\text{top}} := \mathbf{x}(\tilde{\Gamma}^{\text{top}})$  and  $\tilde{\Gamma}^{\text{top}} := (r_1, r_2) \times (0, \pi) \times \{h\}$ . Please note that the transformation of the gradient operator  $\nabla$  is included in the deformation gradient  $\tilde{F}$ , see formula (2.57).

- For simplification, we assume an isotropic, elastic material behavior which can be described by the polyconvex energy density (2.18), with the standard material parameters from remark 6.0.5.

### Numerical solution

The parameter space  $\tilde{\Omega}$  is discretized by a structured mesh with  $3 \times 9 \times 11$  vertices as seen in figure 6.15. We use the quasi-Newton method to solve the optimal control problem on the mesh refinement levels  $L = 0, 1, \dots, 4$  and the standard settings from remark 6.0.4, except that we set the relative tolerance  $\text{rTol} = 10^{-4}$  for the stopping criterion of the quasi-Newton method. The results can be seen in table 6.9. Again we can observe that the convergence of the quasi-Newton method does not deteriorate with increasing mesh refinement level  $L$ .

Figure 6.16 shows a plot of the solution on the finest level  $L = 4$  for half of the stem. We can see that the side facing the sun has a negative turgor pressure which



state dofs $\mathbf{U}$	obj. value $I^* \cdot 10^3$	QN-it	total Krylov it.	total time in s
891	5.40817	16	59	1
5 355	4.25306	16	1551	17
36 531	3.99353	19	1984	184
268 515	3.92927	18	1942	1472
2 056 131	3.91472	17	1912	11293

TABLE 6.9. Numerical results from the quasi-Newton method for the optimal control problem of the stem with turgor pressure.

causes the cells to contract and hence shorten this side of the stem. The opposite side (averted from the sun) has a positive turgor pressure, leading to an increase of the cell volume in this region and therefore stretching this side. As a result, the stem bends towards the direction  $\mathbf{s}^{\text{des}}$  of the sun.

#### 6.2.4. Enclosed Volume

This experiment deals with the enclosed volume between a deformed plate and its reference configuration, see the part *Enclosed Volume* in section 4.1.1. Let  $\Omega = \Omega^{2D} \times (-d, d)$  be a plate of thickness  $2d = 0.01$  with the midsurface  $\Omega^{2D} := (0, 1) \times (0, 1)$  and a Young's modulus of  $E = 100 \text{ N mm}^{-2}$ . The plate is clamped at the rim  $\Gamma^D := \partial\Omega^{2D} \times (-d, d)$ . The control is a boundary force  $\mathbf{g} : \Gamma^N \rightarrow \mathbb{R}^3$  acting on the lower surface  $\Gamma^N := \Omega^{2D} \times \{-d\}$ . During the deformation, the undeformed lower surface  $\Gamma^N$  and the deformed lower surface  $\hat{\Gamma}^N$  enclose a domain whose volume we seek to maximize. The quality term  $Q$  is the volume of the enclosed domain from (4.15), that is

$$Q^V(\mathbf{U}) := \int_{\Omega^{2D}} d + (-d + [\mathbf{U}(\mathbf{x}_1, \mathbf{x}_2, -d)]_3) [\text{cof } F(\mathbf{x}_1, \mathbf{x}_2, -d)]_{33} d\mathbf{x}_1 d\mathbf{x}_2.$$

In order that the optimization problem does not become unbounded, we add a penalty term for the control  $\mathbf{g}$ , that is

$$P^{(g)}(\mathbf{g}) := \int_{\Gamma^N} \mathbf{g}(\mathbf{x})^\top \mathbf{g}(\mathbf{x}) dS = \int_{\Omega^{2D}} \mathbf{g}(\mathbf{x}_1, \mathbf{x}_2)^\top \mathbf{g}(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2.$$

Hence we seek to minimize the objective functional

$$I(\mathbf{U}, \mathbf{g}) = -Q^V(\mathbf{U}) + \gamma^{(g)} P^{(g)}(\mathbf{g}),$$

with a penalty parameter  $\gamma^{(g)} > 0$ . We are interested in the connection between the penalty parameter  $\gamma^{(g)}$  and the volume  $Q$  of the solution.



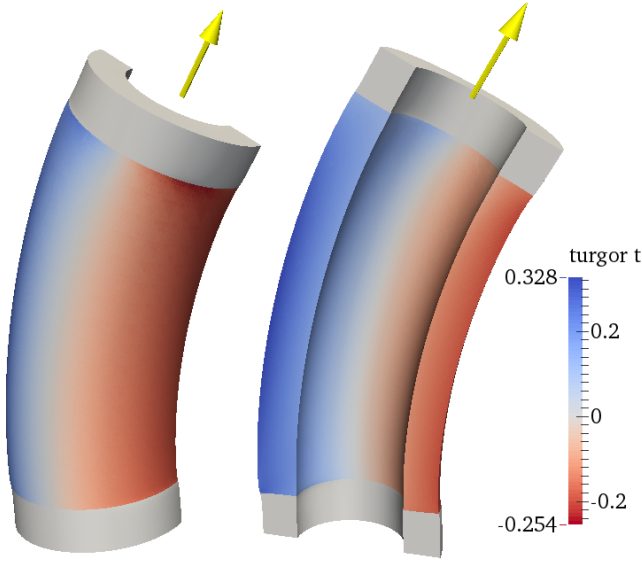


FIGURE 6.16. The solution of the optimal control problem of a stem with a turgor pressure. The yellow arrow indicates the desired direction towards the sun. The blue region indicates a rise and the red region a decrease in the turgor pressure  $t$ . One can also see the thickening and thinning of these regions.

### Numerical Solution

We use the hierarchical plate model from section 2.5 to model the forward problem. The coarse mesh for the midsurface  $\Omega^{2D}$  is shown in figure 6.17. We use simple boundary layers in this example to enhance the quality of the discretization. The numerical tests on the plate model in section 6.1.2 suggest to use the degree  $D^{2D} = 1$  for the thickness  $2d = 0.01$ . The optimal control problem is solved with the quasi-Newton method with the standard settings from remark 6.0.4.

Figure 6.18 shows the undeformed and deformed plate of the solution for a penalty parameter of value  $\gamma^{(g)} = 2^{-5}$ . Most of the load is concentrated in the center of the plate and pulls it upwards.

Next, we vary the penalty parameter  $\gamma^{(g)}$  to illustrate its influence on the solution of the optimal control problem. Figure 6.19 shows the deformed plates of several solutions for a couple of values of the penalty parameter  $\gamma^{(g)}$ . As  $\gamma^{(g)}$  increases, the weight of the cost term  $P$  in the objective functional  $I = -Q + \gamma^{(g)}P$  increases and



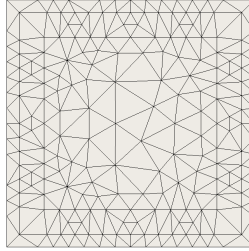


FIGURE 6.17. The coarse mesh of the midsurface  $\Omega^{2D}$  with smaller elements at the rim to improve the quality of the discretization.

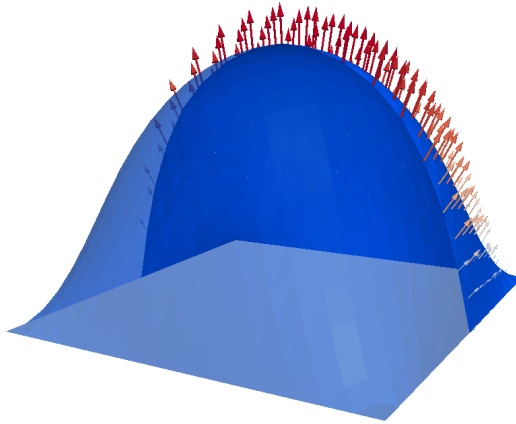


FIGURE 6.18. The undeformed and deformed plate (in blue) attached with arrows which represent the boundary load  $\mathbf{g}$ . Red arrows correspond to a larger load than white arrows.

the volume  $Q$  of the solution shrinks. This monotone relation is plotted in figure 6.20.



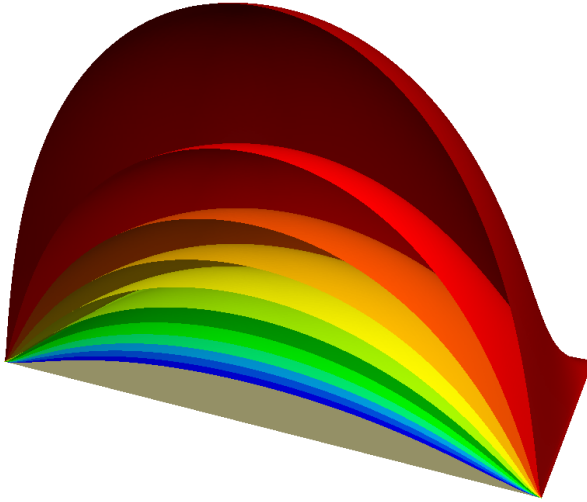


FIGURE 6.19. Half of the deformed plates of the solutions of the optimal control problem for penalty parameters  $\gamma^{(g)} = 2^{-5}, 2^{-4}, \dots, 2^6$  in red to blue and the undeformed plate in gray.

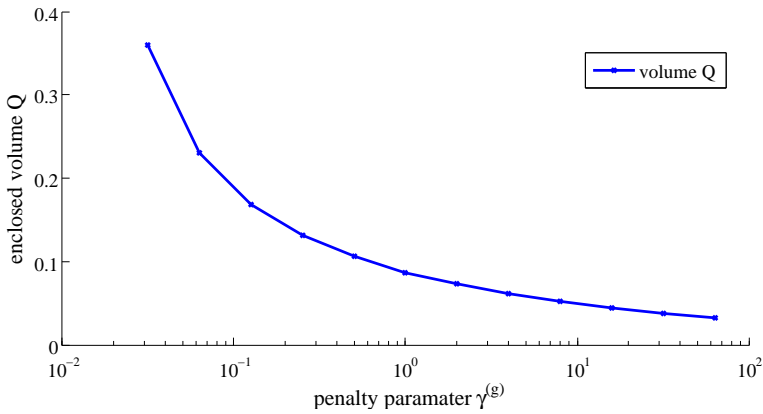


FIGURE 6.20. Plot of the volume  $Q$  of the solution with respect to the penalty parameter  $\gamma^{(g)}$ .



## 7 Conclusions and Perspectives

This dissertation is devoted to numerical algorithms to solve optimal control problems in elasticity with large deformations.

### Forward Problem

The forward problem of a large elastic deformation is a nonlinear variational equality derived from an energy minimization. The standard loads, exterior volume or boundary loads, cannot be used to model certain phenomena in biology, like turgor pressure or muscle tension. We gave models for both of these interior loads in the case of large elastic deformations. In order to incorporate these new loads into the energy minimization, we formulated corresponding energy functionals for these loads.

The computation of the solution of the forward problem by Newton's method can be challenging for large deformations. Here, we proposed a new line search based on a guiding criterion that is applicable in many situations and solves the problem of a local negative local volume change  $J < 0$ . The underlying guiding function  $G$  can be adapted to other elastic models as well, e.g. by adding terms like  $\|\text{cof } F\|_F^2$ .

Newton's method requires the repeated solution of linear systems. The underlying linear operators are self-adjoint but not necessarily positive definite. Hence, the usage of the current stiffness matrix  $A_h$  for a multigrid V-cycle preconditioner might result in a failure of a MINRES or CG method. Still, the current stiffness matrix  $A_h$  offers a faster convergence than the stiffness matrix  $K_{\text{lin, elast}}$  of the linear elasticity model. We proposed a strategy similar to the idea behind a truncated CG method to use a multigrid approximation of  $A_h$  as a preconditioner as long as it does not prove to be indefinite.

### Optimal Control Problem

During this work, we use an *optimize-then-discretize* approach. That means we formulated Newton's method and the MINRES and CG method in the Hilbert space  $\mathcal{U}$ . This allows us not only to formulate the appropriate stopping criterion, but also shows the importance of the initial Hessian  $H_0$  in the BFGS update. At the end, this results in optimization algorithms which show no mesh dependence with respect to iteration numbers and convergence behavior. Combined with a fast forward solver, this even allows the solution of large scale problems in a reasonable time.

The standard tracking-type objective functional does not always work in elasticity with large deformations because we might not be able to provide a desired state



$\mathbf{U}^{\text{des}}$ . We demonstrated the modeling of three alternative functionals suitable for different problems. Numerical experiments were performed to show how these quality functionals work. Additionally, we incorporated a parametrization by curvilinear coordinates and a hierarchical plate model into the forward problem and the optimal control problem.



## Outlook

One application of optimal control of elasticity from Lubkoll et al. (2012) is the shape reconstruction of facial structures by implants. Here, the problem remains how to measure the shape of the facial structure and how to match it. The difficulties of this approach were discussed in the paragraph *Desired Shape* in section 4.1.1. The representation of the shape depends on the measuring method and might not be directly included into a quality term that quantifies the difference between the current shape and the desired shape.

In this work, we assume an isotropic and elastic material behavior modeled by a polyconvex energy density. However, this model might still be a simplification for problems with biological tissues (see Holzapfel and Ogden (2006) and Balzani et al. (2006)) or fiber-reinforced composite material (see for example Bunsell and Renard (2005)). In order to prevent the damage of a tissue or plastic deformation of a mechanical component, we could consider constraints in the optimal control problem, for example an upper bound on the von-Mises stress.

Another potential application is the deep-drawing process of plates. Here, not only the plasticity plays a central role but also the elasticity, for example in the undesirable spring-back effect. Work on this has been done by Wachsmuth (2012a) and Wachsmuth (2012b) for the case of small deformations, though a simulation and optimization requires further techniques like plate models that can deal with plasticity and corresponding material laws to describe the behavior of a deep-drawing process. Eventually, it can be expected that the computational work to optimize deep-drawing process is huge due to very complex models.

The quality of the solution obtained by the finite element method depends strongly on the discretization, i. e. the elements and the mesh. Here, improvements could be gained by using adaptive mesh refinement methods. First, one would need an error estimator made for the forward problem, see Meyer (2007). Taking the estimator from the linear model is not an option as both models differ too much for large deformations. Even more, the implementation of an adaptive mesh refinement is not straightforward for nonlinear problems, let alone optimal control problems. Another promising approach could be full multigrid on the optimal control problem, see for example Schöberl et al. (2011) and Pillwein and Takacs (2014).

The bottleneck in the Lagrange-Newton method is clearly the preconditioner for the MINRES method. Even with the current stiffness matrix  $A_h$ , the MINRES method needs a large number of iterations with the block preconditioner. A reduction of these iteration numbers by a better preconditioner would highly improve the Lagrange-Newton method. Also, the scaling of the blocks is very difficult because spectral estimates of the Lagrange system matrix are hardly known. A method to determine good scaling factors for the blocks of the preconditioner would be very desirable.



On the other hand, the unfortunate handicap of the quasi-Newton method was the incompatibility of the BFGS history with the idea of nested iterations. An idea how to gain information from a previous mesh level could improve the performance of this optimization algorithm even further.

The idea of combining curvilinear coordinates and the hierarchical plate model could be a promising way to deal with shells like thin tubes. For example, one could use the plate model of the parameter space which is then mapped to the reference configuration. The intriguing point in this approach is the fact that it does not need the curvature information of the shell, which is a common element in shell models and might be quite challenging.



# A Appendix

## A.1 Matrix Calculus

In this section we recall some calculation rules for matrices and derivatives with respect to matrices, as they are repeatedly used in this work.

**Lemma A.1.1** (Basic calculus for matrices). Let  $A, B \in \mathbb{R}^{n \times n}$  be regular matrices and let  $Q \in \mathbb{R}^{n \times n}$  be orthogonal, e. g.  $Q^\top Q = I$ , with positive determinant  $\det Q = 1$ . Then we have the following identities:

- (1)  $\text{tr}(AB) = \text{tr}(BA)$
- (2)  $A : B = A^\top : B^\top = B : A$
- (3)  $A : (BC) = (B^\top A) : C = (AC^\top) : B$
- (4)  $\|QA\|_F = \|A\|_F$
- (5)  $\text{cof}(AB) = \text{cof}(A) \text{cof}(B)$
- (6)  $\text{cof } Q = Q$
- (7)  $\|\text{cof } F\|_F^2 = \frac{1}{2}(F : F)^2 - \frac{1}{2} \text{tr}(F^\top F)^2$

*Proof.*

- (1) We have  $[AB]_{ij} = [A]_{ik}[B]_{kj}$  and hence the trace operator is

$$\text{tr}(AB) = \sum_{ik} A_{ik} B_{ki} = \sum_{ik} B_{ki} A_{ik} = \text{tr}(BA).$$

- (2) Using the definition for  $A : B := \text{tr}(B^\top A)$  from section 1.3, (1) and  $\text{tr } A = \text{tr } A^\top$  gives

$$\begin{aligned} A : B &= \text{tr}(A^\top B) = \text{tr}(BA^\top) = B^\top : A^\top \\ &= \text{tr}(B^\top A) = B : A. \end{aligned}$$

- (3) A direct calculation gives

$$\begin{aligned} A : (BC) &= \text{tr}(A^\top BC) = (B^\top A) : C \\ &= \text{tr}(CA^\top B) = (AC^\top) : B \end{aligned}$$

- (4) With the definition of the Frobenius norm we have

$$\|QA\|_F^2 := (QA) : (QA) = \text{tr}(A^\top Q^\top QA) = \text{tr}(A^\top A) = A : A = \|A\|_F^2.$$



- (5) The definition of the cofactor gives  
 $\text{cof}(AB) = \det(AB)(AB)^{-\top} = (\det A) A^{-\top} (\det B) B^{-\top} = \text{cof } A \text{ cof } B.$
- (6) Using the definition of the cofactor gives  $\text{cof } Q = \det Q Q^{-\top} = 1 Q = Q.$
- (7) See (Ciarlet, 1988, pg. 186).

The next lemma states some formulas for the differentiation with respect to matrices.

**Lemma A.1.2** (Formulas for the differentiation with respect to matrices). Let  $f : \mathbb{M}_+^3 \rightarrow \mathbb{R}$  be a differentiable function. Then we have the identities for the following derivatives with respect to  $F \in \mathbb{M}_+^3$  in direction  $G \in \mathbb{M}^3$ :

- (1)  $\frac{\partial}{\partial F} (F : F)[G] = 2 F : G$
- (2)  $\frac{\partial}{\partial F} (F^\top)[G] = G^\top$
- (3)  $\frac{\partial}{\partial F} (\text{tr}(AFB))[G] = A G B$
- (4)  $\frac{\partial}{\partial F} (\det F)[G] = (\det F) F^{-\top} : G = (\text{cof } F) : G$
- (5)  $\frac{\partial}{\partial F} (\ln \det F)[G] = F^{-\top} : G$
- (6)  $\frac{\partial}{\partial F} (F^{-1})[G] = -F^{-1} G F^{-1}$
- (7)  $\frac{\partial}{\partial F} (\|\text{cof } F\|_F^2)[G] = 2(F : F)(F - F F^\top F) : G$
- (8)  $\frac{\partial}{\partial F} (F^{-\top})[G] = -F^{-\top} G^\top F^{-\top}$
- (9)  $\frac{\partial}{\partial F} (\text{cof } F)[G] = (\text{cof } F : G) F^{-\top} - \text{cof } F G^\top F^{-\top}$

*Proof.* For (1) - (6), we refer to K.M. Petersen (2008). For (7), we use A.1.1-(1), A.1.1-(6), (1), (3) and the product rule to get

$$\begin{aligned}
 \frac{\partial \|\text{cof } F\|_F^2}{\partial F}[G] &= \frac{\partial}{\partial F} \left( \frac{1}{2} (F : F)^2 - \frac{1}{2} \text{tr}(F^\top F F^\top F) \right) \\
 &= 2(F : F)(F : G) - \frac{1}{2} \text{tr}(G^\top F F^\top F) - \frac{1}{2} \text{tr}(F^\top G F^\top F) \\
 &\quad - \frac{1}{2} \text{tr}(F^\top F G^\top F) - \frac{1}{2} \text{tr}(F^\top F F^\top G) \\
 &= 2(F : F)(F : G) - 2 \text{tr}(F^\top F F^\top) : G \\
 &= 2(F : F)(F : G) - 2(F F^\top F) : G = 2(F : F)(F - F F^\top F) : G.
 \end{aligned}$$

For (8), we use (6) and (2) to get the proposition, e.g.

$$\frac{\partial}{\partial F} (F^{-\top})[G] = \frac{\partial}{\partial F^\top} (F^{-\top})[G^\top] = -F^{-\top} G^\top F^{-\top}.$$

Formula (9) is the application of the product rule, (4) and (8), e.g.

$$\frac{\partial}{\partial F} (\text{cof } F)[G] = \frac{\partial}{\partial F} (\det F F^{-\top})[G] = (\text{cof } F : G) F^{-\top} - (\det F) F^{-\top} G^\top F^{-\top}.$$



## A.2 Derivatives of Energy Functionals

The following lemmas deal with the derivatives of certain energy terms.

**Lemma A.2.1** (Derivatives of the energy density  $w$ ). The first and second derivative of  $w$  from (2.18) are

$$w_{,F}(F)[G] = \left( 2a F + 2b(F : F)F - 2b F^\top F F^\top + c(\det F)^2 F^{-\top} - d F^{-\top} \right) : G,$$

and

$$w_{,FF}(F)[G, H] =$$

$$\begin{aligned} & \left( (2a + 2b F : F)G + 4b(F : G)F - 2b(F(F^\top G + G^\top F) + GF^\top F) \right. \\ & \left. + 4c \operatorname{cof} F(G : \operatorname{cof} F) + (d(\det F)^{-2} - 2c) \operatorname{cof}(F)G^\top \operatorname{cof}(F) \right) : H. \end{aligned}$$

*Proof.* Applying lemma A.1.2 yields for the first derivative

$$\begin{aligned} w_{,F}(F)[G] &= \frac{\partial}{\partial F} \left( a \|F\|_F^2 + b \|\operatorname{cof} F\|_F^2 + c(\det F)^2 - d \ln(\det F) + e \right) [G] \\ &= \left( 2a F + 2b(F : F)F - 2b F F^\top F + 2c(\det F)^2 F^{-\top} - d F^{-\top} \right) : G. \end{aligned}$$

The second derivative  $w_{,F}$  can be obtained by

$$\begin{aligned} w_{,FF}(F)[G, H] &= \frac{\partial}{\partial F} \left( 2a F : G + 2b(F : F)F : G - 2b(F F^\top F) : G \right. \\ & \quad \left. + 2c(\det F)^2 F^{-\top} : G - d F^{-\top} \right) [H] \\ &= 2a(G : H) + 2b(F : F)(G : H) + 4b(F : G)(F : H) \\ & \quad - 2b((H F^\top F) + (F H^\top F) + (F F^\top H)) : G \\ & \quad + 2c(\det F)^2 (F^{-\top} : G)(F^{-\top} : H) - 2c(\det F)^2 (F^{-\top} H^\top F^{-\top}) : G \\ & \quad - d(F^{-\top} H^\top F^{-\top}) : G \\ &= 2a(G : H) + 2b(F : F)(G : H) + 4b(F : G)(F : H) \\ & \quad - 2b(F(F^\top H + H^\top F) + H F^\top F) : G \\ & \quad + 2c(\operatorname{cof} F : G)(\operatorname{cof} F : H) - (2c + d(\det F)^{-2})(\operatorname{cof} F H^\top \operatorname{cof} F) : G, \end{aligned}$$

and some simple re-arrangements.

**Lemma A.2.2** (Derivatives of the inner pressure). The first and second derivative of  $W_{,U}^{(t)}$  from (2.39) are

$$W_{,U}^{(t)}[\delta U] = - \int_{\Omega} t(\mathbf{x}) (\operatorname{cof} F(\mathbf{x}) : \nabla \delta U(\mathbf{x})) d\mathbf{x},$$



and (omitting the argument “ $(\mathbf{x})$ ”)

$$\begin{aligned} W_{,\mathbf{U}\mathbf{U}}^{(t)}[\delta\mathbf{U}, \delta\mathbf{V}] &= - \int_{\Omega} t(F^{-\top} : \nabla\delta\mathbf{U}) (\operatorname{cof} F : \nabla\delta\mathbf{V}) \\ &\quad - (F^{-1}\nabla\delta\mathbf{U}) : (\nabla\delta\mathbf{V}^{\top} \operatorname{cof} F) d\mathbf{x}. \end{aligned}$$

*Proof.* The first derivative follows directly from A.1.2-(4) and for the second derivative we apply the product rule and A.1.2-(9), e. g. with  $G = \nabla\delta\mathbf{U}$  and  $H = \nabla\delta\mathbf{V}$ ,

$$\begin{aligned} \frac{\partial}{\partial F} (\operatorname{cof} F : G)[H] &= (\operatorname{cof} F : H)F^{-\top} : G - (\operatorname{cof} FH^{\top}F^{-\top}) : G \\ &= (\operatorname{cof} F : H)F^{-\top} : G - (\operatorname{cof} FH^{\top}) : (F^{-1}G). \end{aligned}$$

**Lemma A.2.3** (Derivatives of the fiber tension). The first and second derivatives of  $W_{,\mathbf{U}}^{(m)}$  from (2.44) is

$$W_{,\mathbf{U}}^{(m)}[\delta\mathbf{U}] = - \int_{\Omega} m(\mathbf{x}) \frac{(\mathbf{a}(\mathbf{x})^{\top} F^{\top}(\mathbf{x}) \nabla\delta\mathbf{U}(\mathbf{x}) \mathbf{a}(\mathbf{x}))}{\|F(\mathbf{x})\mathbf{a}(\mathbf{x})\|_2^2} d\mathbf{x}$$

and (omitting the argument “ $(\mathbf{x})$ ”)

$$\begin{aligned} W_{,\mathbf{U}\mathbf{U}}^{(m)}[\delta\mathbf{U}, \delta\mathbf{V}] &= - \int_{\Omega} m \frac{(\mathbf{a}^{\top} \nabla\delta\mathbf{V}^{\top} \nabla\delta\mathbf{U} \mathbf{a})}{\|F\mathbf{a}\|_2^2} \\ &\quad - 2m \frac{(\mathbf{a}^{\top} F^{\top} \nabla\delta\mathbf{U} \mathbf{a}) (\mathbf{a}^{\top} F^{\top} \nabla\delta\mathbf{V} \mathbf{a})}{\|F\mathbf{a}\|_2^4} d\mathbf{x} \end{aligned}$$

*Proof.* We omit the argument “ $(\mathbf{x})$ ” to a better readability in the following. We rewrite  $W^{(m)}$  as

$$\begin{aligned} W^{(m)}(\mathbf{U}, m) &= - \int_{\Omega} m \ln (\mathbf{a}^{\top} F^{\top} F \mathbf{a})^{\frac{1}{2}} d\mathbf{x} \\ &= - \int_{\Omega} \frac{1}{2} m \ln (\mathbf{a}^{\top} F^{\top} F \mathbf{a}) d\mathbf{x}, \end{aligned}$$

and apply the chain rule to obtain

$$\begin{aligned} W_{,\mathbf{U}}^{(m)}(\mathbf{U}, m)[\delta\mathbf{U}] &= - \int_{\Omega} \frac{1}{2} m (\mathbf{a}^{\top} F^{\top} F \mathbf{a})^{-1} 2(F\mathbf{a})^{\top} (\nabla\delta\mathbf{U} \mathbf{a}) d\mathbf{x} \\ &= - \int_{\Omega} m \frac{(\mathbf{a}^{\top} F^{\top} \nabla\delta\mathbf{U} \mathbf{a})}{\|F\mathbf{a}\|_2^2} d\mathbf{x}. \end{aligned}$$



The second derivative is gained by the quotient rule, i. e.

$$W_{,UU}^{(m)}(\mathbf{U}, m)[\delta \mathbf{U}, \delta \mathbf{V}] = - \int_{\Omega} m \frac{(\mathbf{a}^{\top} \nabla \delta \mathbf{V}^{\top} \nabla \delta \mathbf{U} \mathbf{a}) - 2 (\mathbf{a}^{\top} F^{\top} \nabla \delta \mathbf{U} \mathbf{a}) (\mathbf{a}^{\top} F^{\top} \nabla \delta \mathbf{V} \mathbf{a})}{\|F \mathbf{a}\|_2^4} d\mathbf{x}.$$

**Lemma A.2.4** (Green's formulas). Let  $\Omega \subset \mathbb{R}^n$  be a bounded and non-empty set with piecewise continuous boundary  $\partial\Omega$  and the outer normal vector  $\mathbf{n}$ . Let furthermore  $u, v : \Omega \rightarrow \mathbb{R}$  be two continuous differentiable scalar fields and  $\mathbf{V} : \Omega \rightarrow \mathbb{R}$  be a continuous differentiable vector field. Then we have the equations

$$\int_{\Omega} u(\mathbf{x}) v_{,i}(\mathbf{x}) d\mathbf{x} = - \int_{\Omega} u_{,i}(\mathbf{x}) v(\mathbf{x}) d\mathbf{x} + \int_{\partial\Omega} u(\mathbf{x}) v(\mathbf{x}) \mathbf{n}_i(\mathbf{x}) dS$$

and

$$\int_{\Omega} \operatorname{div} \mathbf{V}(\mathbf{x}) d\mathbf{x} = \int_{\partial\Omega} \mathbf{V}(\mathbf{x})^{\top} \mathbf{n}(\mathbf{x}) dS. \quad (\text{A.1})$$

*Proof.* Green's identity is a direction implication of integrating by parts.



## B Theses

- (1) This thesis regards numerical aspects of optimal control problems with large deformation elasticity. The forward problem is modeled by an energy minimization based on a polyconvex energy density, whose first order optimality conditions are a nonlinear partial differential equation. The framework of an energy minimization has the advantage that we can view the forward problem as an optimization problem, which allows new algorithms and ensures the positive semi-definiteness of the stiffness matrix in a solution.
- (2) The conventional volume and boundary loads are not sufficient to describe certain phenomena in biology, e. g. turgor pressure or muscle tension. We model both of these internal loads and incorporate them into the energy minimization by deriving corresponding energy terms for the loads, showing that they are conservative.
- (3) Some problems are conveniently formulated in curvilinear coordinates. We show how such parametrizations of the coordinates  $\mathbf{x} = \mathbf{x}(\tilde{\mathbf{x}})$  can be included into the framework of the forward problem. Furthermore, we demonstrate how the hierarchical plate model is viewed as a reduction technique which imposes a certain structure on the displacement field  $\mathbf{U}$ . Therefore, the hierarchical plate model can be integrated into the full 3D model of elasticity.
- (4) To solve the (nonlinear) forward system, we apply Newton's method. However, a globalization strategy is required to ensure the convergence of Newton's method. We give an overview of popular strategies and suggest a new one: a guiding criterion based on the guiding function  $G$ . This method can be applied in many situations and shows a similar behavior to an Armijo line search on the stored energy  $W$ .
- (5) The preconditioner based on the linear model of elasticity does not behave well for large deformations, since the linear and nonlinear model of elasticity differ too much in these cases. An experimental preconditioner is suggested which allows the usage of the current stiffness matrix  $A_h$  and



improves the convergence of the CG and MINRES method noticeably.

- (6) In the optimal control problem, the standard tracking type functional with a desired state  $\mathbf{U}^{\text{des}}$  might not work for certain problems because the desired state  $\mathbf{U}^{\text{des}}$  is unknown. We give three alternative quality functionals based on a regional penalization, a desired direction and an enclosed volume. We also derive representations of the first derivatives of these functionals by the Hadamard calculus. These formulas are not only simpler than a direct differentiation, but also reveal the dependencies of the first derivatives.
- (7) Two methods are presented to solve the optimal control problem: an all-at-once approach with a Lagrange-Newton method and a reduced formulation with a quasi-Newton method. In the latter, the state  $\mathbf{U}$  is eliminated by a solution operator  $S$ . In the quasi-Newton method, we work with the derivative  $I_{\mathcal{C}}^{\text{red}}$  of the reduced objective function instead of the gradient  $\nabla_{\mathcal{C}} I^{\text{red}}$ .
- (8) Newton's method and the CG and MINRES methods are formulated in a function space setting. This follows the "optimize-then-discretize" approach and appears to be a natural approach for these optimal control problems. This is useful in order to derive algorithms that are mesh independent. We can observe this in our numerical experiments.
- (9) The finite element software FENICS is used to implement the algorithms and the problem formulations themselves. Here we demonstrate how the different problems can be written in a joint framework that allows an easy incorporation of parametrizations and the hierarchical plate model.



# C Curriculum Vitæ

## Personal information

Name	Andreas Günnel
Date of Birth	March 12, 1986
Place of Birth	Zwickau, Germany
Marital Status	Unmarried
Citizenship	German

## Education

- 2010 Diploma in Technomathematics from TU Chemnitz of Technology, Germany. Title:  
*Adaptive Mesh Design in Shape Optimization with the Discrete Adjoint Method*  
Supervisor: Dr. René Schneider
- 2008–2009 Studies in Physics and Mathematics, Aberystwyth University of Wales, United Kingdom
- 2004–2010 Studies in Technomathematics (Minors: Mechanics), TU Chemnitz, Germany
- 1996–2004 Abitur (*A-levels*) at Georgengymnasium Zwickau (later Käthe-Kollwitz-gymnasium), mark *1,4*

## Professional record

- 2010–present Scientific assistant, TU Chemnitz, Department of Mathematics
  - SFB 393, A11: Gemischte Formulierungen: adaptive anisotrope finite Elemente und parallele Löser
  - BMBF project: Differenziertes Mentoring- und Betreuungsprogramm (Qualitätspakt Lehre)
  - DFG project: Cluster of Excellence MERGE, Interacting Research Domain F: Modelling, Integrative Simulation and Optimisation
- 2006–2010 Student assistant, TU Chemnitz, Department of Mathematics

## Teaching experience

- 2012–2013 Coordinator and Tutor, Learning Center for Mathematics



---

2012-2013	Teaching Assistant, Optimization for Non-mathematicians
2012-2013	Tutor, Preparatory courses for Business Administration students
2011-2012	Teaching Assistant, Nonlinear Optimization
2011-2012	Teaching Assistant, Statistics for Business Administration students
2010, 2012	Teaching Assistant, Mathematics for CS, IT and ET students
2007-2008	Tutor for athletes, Higher Mathematics
2006-2012	Teaching Assistant, Higher Mathematics

### Diploma theses co-supervised

Sarah Stoppe, 2012	<i>Simulation elastoplastischer Verformungen unter großen Deformationen</i>
Philipp Menzel, 2013	<i>Geometrische Verfahren für Hamiltonsche Systeme: Theorie und Anwendungen</i>
Anna Bauer, 2013	<i>Optimierungsverfahren für die ressourceneffiziente Planung von Montageanlagen</i>

### Publications in Journals

- [1] A. Günnel, R. Herzog and E. Sachs. A Note on Preconditioners and Scalar Products in Krylov Subspace Methods for Self-Adjoint Problems in Hilbert Space. *Electronic Transactions on Numerical Analysis*, 41:13–20, 2014.

### Presentations

#### Presentations at Conferences and in Minisymposia

- (1) *Optimal Control of Large Deformation Elasticity by Fibre Tension*, 85th Annual Meeting of GAMM, March 10-14, 2014, Erlangen, Germany
- (2) *Numerical Aspects of Plates under Large Deformations*, 25th FEM-Symposium September 24-26, 2012, Chemnitz, Germany
- (3) *Optimal Control of Elasticity with Large Deformations*, 5th International Conference on High Performance Scientific Computing, March 5-9, 2012, Hanoi, Vietnam
- (4) *Contact Problem for Elasticity with large Deformations*, 24th FEM-Symposium September 28-20, 2011, Holzhau, Germany

#### Colloquium and Seminar Talks

- (5) *Optimal Control of Elasticity for Large Deformations*, Research Seminar Numerics, January 7, 2014, TU Chemnitz
- (6) *Elasticity for Large Deformations: Modelling and Optimal Control*, Research Seminar Numerics, July 2, 2013, TU Chemnitz
- (7) *Elasticity for Large Deformations: Modelling and Optimal Control*, Chemnitzer Seminar zur Optimalsteuerung March 02-09, 2013, Haus in Ennstal, Austria



- (8) *Formulations of Elasticity for Large Deformations*, Research Seminar Numerics, January 22, 2013, TU Chemnitz
- (9) *Numerical Aspects of Plates under Large Deformations*, Research Seminar Scientific Computing, July 13, 2012, TU Chemnitz
- (10) *Optimal Control of Elasticity with large deformations*, Chemnitzer Seminar zur Optimalsteuerung February 11-18, 2012, Haus in Ennstal, Austria
- (11) *Optimal Control of Elasticity with Large Deformations*, Research Seminar Numerics, January 10, 2012, TU Chemnitz
- (12) *Contact Problem for Elasticity with large Deformations*, Research Seminar Numerics, July 5, 2011, TU Chemnitz
- (13) *Differential Geometry and Large Deformations*, June 24, 2011, Kooperationsseminar Chemnitz-Magdeburg, MPI Magdeburg
- (14) *An Introduction to three-dimensional Differential Geometry*, Chemnitzer Seminar zur Optimalsteuerung March 6-13, 2010, Haus in Ennstal, Austria
- (15) *Plate Equation with Large Deformations*, Research Seminar Numerics, January 18 2011, TU Chemnitz
- (16) *Adaptive Mesh Design in Shape Optimization with the Discrete Adjoint Method*, Chemnitzer Seminar zur Optimalsteuerung March 6-13, 2010, Haus in Ennstal, Austria

## Organizing

- 2013 3. European Conference on Computational Optimization, TU Chemnitz, Germany
- 2011–2014 Chemnitzer Seminar zur Optimalsteuerung, Haus im Ennstal, Austria
- 2011–2012 FEM-Symposium, Holzhau an TU Chemnitz, Germany

## Committee work and volunteer activities

- 2014–present Member of the examination board, Department of Mathematics, TU Chemnitz
- 2013–present Member of the library commission, TU Chemnitz
- 2012–2014 Board member of the VAMC (Vertretung akademischer Mittelbau der TU Chemnitz)
- 2012–present Member of the study commission *Integrated International Master and PhD Program in Mathematics*, TU Chemnitz
- 2010–present Coordinator, *Korrespondenzzirkel für Mathematik (Klassenstufe 11 und 12)*,



# Bibliography

- Alnæs, M. S., Logg, A., Mardal, K.-A., Skavhaug, O., and Langtangen, H. P. (2009). Unified framework for finite element assembly. *International Journal of Computational Science and Engineering*, 4(4):231–244.
- Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., and Wells, G. N. (2013). Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, To appear.
- Alnæs, M. S. and Mardal, K.-A. (2010). On the efficiency of symbolic computations combined with code generation for finite element methods. *ACM Transactions on Mathematical Software*, 37(1).
- Alt, W. (1990). The Lagrange-Newton method for infinite-dimensional optimization problems. *Numerical Functional Analysis and Optimization. An International Journal*, 11(3-4):201–224.
- Alt, W. (1994). Local convergence of the Lagrange-Newton method with applications to optimal control. *Control and Cybernetics*, 23(1-2):87–105. Parametric optimization.
- Alt, W. (2001). Mesh-independence of the Lagrange-Newton method for nonlinear optimal control problems and their discretizations. *Annals of Operations Research*, 101:101–117. Optimization with data perturbations, II.
- Alt, W. and Malanowski, K. (1993). The Lagrange-Newton method for nonlinear optimal control problems. *Computational Optimization and Applications. An International Journal*, 2(1):77–100.
- Antman, S. S. (1984). *Nonlinear Problems of Elasticity*. Springer, New York.
- Arnold, D. N. and Falk, R. S. (1989). A uniformly accurate finite element method for the Reissner-Mindlin plate. *SIAM Journal on Numerical Analysis*, 26(6):1276–1290.
- Babuška, I., D’Harcourt, J. M., and Schwab, C. (1993). Optimal shear correction factors in hierarchical plate modelling. *Mathematical Modelling and Scientific Computing*, 1(1-2):1–30.
- Babuška, I. and Li, L. (1991). The problem of plate modelling - theoretical and computational results. *Comput. Methods Appl. Mech. Engrg.*, 100:249–273.



- Babuška, I., Szabó, B. A., and Actis, R. L. (1992). Hierarchic models for laminated composites. *International Journal for Numerical Methods in Engineering*, 33(3):503–535.
- Babuška, I., Whiteman, J. R., and Strouboulis, T. (2011). *Finite elements*. Oxford University Press, Oxford. An introduction to the method and error estimation.
- Ball, J. M. (1976/1977). Convexity conditions and existence theorems in nonlinear elasticity. *Archive for Rational Mechanics and Analysis*, 63(4):337–403.
- Balzani, D., Neff, P., Schröder, J., and Holzapfel, G. A. (2006). A polyconvex framework for soft biological tissues. Adjustment to experimental data. *International Journal of Solids and Structures*, 43(20):6052–6070.
- Bertram, A. (2008). *Elasticity and Plasticity of Large Deformations: An Introduction*. Springer, Berlin Heidelberg.
- Bower, A. T. (2009). *Applied Mechanics of Solids*. CRC Press, Taylor and Francis Group, Boca Bato.
- Bower, A. T. (2012). Applied mechanics of solids. <http://solidmechanics.org/>.
- Braess, D. (2007). *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, Cambridge.
- Braess, D. and Hackbusch, W. (1983). A new convergence proof for the multigrid method including the  $V$ -cycle. *SIAM journal on numerical analysis*, 20(5):967–975.
- Brenner, S. and Scott, L. (2002). *The Mathematical Theory of Finite Element Methods*. Springer, New York, second edition.
- Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms. II. The new algorithm. *Journal of the Institute of Mathematics and its Applications*, 6:222–231.
- Broyden, C. G., Dennis, Jr., J. E., and Moré, J. J. (1973). On the local and superlinear convergence of quasi-Newton methods. *Journal of the Institute of Mathematics and its Applications*, 12:223–245.
- Bunsell, A. R. and Renard, J. (2005). *Fundamentals of Fibre Reinforced Composite Materials*. MPG Book Ltd., Bodmin, Cornwall.
- Campbell, N. A. (1997). *Biologie*. Spektrum Akademischer Verlag, Heidelberg, Berlin, Oxford.



- Ciarlet, P. G. (1988). *Mathematical elasticity. Vol. I*, volume 20 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam. Three-dimensional elasticity.
- Ciarlet, P. G. (1997). *Mathematical elasticity. Vol. II*, volume 27 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam. Theory of plates.
- Ciarlet, P. G. (2005). *An introduction to differential geometry with applications to elasticity*. Springer, Dordrecht.
- Ciarlet, P. G. and Geymonat, G. (1982). Sur les lois de comportement en élasticité non linéaire compressible. *Comptes Rendus des Séances de l'Académie des Sciences. Série II. Mécanique, Physique, Chimie, Sciences de l'Univers, Sciences de la Terre*, 295(4):423–426.
- Daniel, J. W. (1967). The conjugate gradient method for linear and nonlinear operator equations. *SIAM J. Numer. Anal.*, 4:10–26.
- Deuffhard, P. (2004). *Newton methods for nonlinear problems*, volume 35 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin. Affine invariance and adaptive algorithms.
- Elman, H., Silvester, D., and Wathen, A. (2005). *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York.
- Eppler, K. (2010). On Hadamard shape gradient representations in linear elasticity. Technical Report SPP1253-104, Priority Program 1253, German Research Foundation.
- Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322.
- Gahalaut, K. P. S., Kraus, J. K., and Tomar, S. K. (2013). Multigrid methods for isogeometric discretization. *Computer Methods in Applied Mechanics and Engineering*, 253:413–425.
- Geiger, C. and Kanzow, C. (1999). *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben*. Springer, New York.
- Gergelits, T. and Strakoš, Z. (2013). Composite convergence bounds based on Chebyshev polynomials and finite precision conjugate gradient computations. *Numer. Algorithms*, pages 1–24.
- Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24:23–26.



- Griewank, A. (2000). *Evaluating Derivatives*. SIAM, Philadelphia.
- Günnel, A., Herzog, R., and Sachs, E. (2014). A note on preconditioners and scalar products in Krylov subspace methods for self-adjoint problems in Hilbert space. *Electronic Transactions on Numerical Analysis*, 41:13–20.
- Hackbusch, W. (1980). *Multi-grid Methods and Applications*. Numerical Mathematics and Scientific Computation. Springer-Verlag, Berlin.
- Herzog, R. and Kunisch, K. (2010). Algorithms for PDE-constrained optimization. *GAMM Reports*, 33(2):163–176.
- Herzog, R., Meyer, C., and Wachsmuth, G. (2013). B- and strong stationarity for optimal control of static plasticity with hardening. *SIAM Journal on Optimization*, 23(1):321–352.
- Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, 49:409–436 (1953).
- Hinze, M. and Kunisch, K. (2004). Second order methods for boundary control of the instationary Navier-Stokes system. *ZAMM*, 84(3):171–187.
- Holzapfel, G. A. and Ogden, R. W. (2006). *Mechanics of Biological Tissue*. Springer, Berlin.
- Ito, K. and Kunisch, K. (1996a). Augmented Lagrangian-SQP methods for nonlinear optimal control problems of tracking type. *SIAM Journal on Control and Optimization*, 34:874–891.
- Ito, K. and Kunisch, K. (1996b). Augmented Lagrangian-SQP Methods in Hilbert spaces and application to control in the coefficients problem. *SIAM Journal on Optimization*, 6(1):96–125.
- Kelley, C. T. (1995). *Iterative methods for linear and nonlinear equations*, volume 16 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA. With separately available software.
- Kelley, C. T. (1999). *Iterative methods for optimization*, volume 18 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.
- Kelley, C. T. (2003). *Solving nonlinear equations with Newton's method*, volume 1 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.
- Kelley, C. T. and Sachs, E. W. (1989). A pointwise quasi-Newton method for unconstrained optimal control problems. *Numerische Mathematik*, 55(2):159–176.



- 
- Kelley, C. T., Sachs, E. W., and Watson, B. (1991). Pointwise quasi-Newton method for unconstrained optimal control problems. II. *Journal of Optimization Theory and Applications*, 71(3):535–547.
- Kevan, P. G. (1975). Sun-tracking solar furnaces in high arctic flowers: Significance for pollination and insects. *Science*, 189(4204):723–726.
- Kirby, R. C. (2004). Algorithm 839: Fiat, a new paradigm for computing finite element basis functions. *ACM Transactions on Mathematical Software*, 30(4):502–516.
- Kirchhoff, G. (1850). Über das Gleichgewicht und Bewegung einer elastischen Scheibe. *J. Reine Angew. Math.*, 40:51–58.
- K.M. Petersen, M. P. (2008). *The Matrix Cookbook*. Technical University of Denmark.
- Lang, A. R. G. and Begg, J. E. (1979). Movements of helianthus annuus leaves and head. *Journal of Applied Ecology*, 16(1):299–305.
- Lee, P.-S. and Bathe, K.-J. (2010). The quadratic MITC plate and MITC shell elements in plate bending. *Advances in Engineering Software*, 21:712–728.
- Logg, A. (2007). Automating the finite element method. *Archives of Computational Methods in Engineering*, 14(2):93–138.
- Logg, A., Mardal, K.-A., Wells, G. N., et al. (2012a). *Automated Solution of Differential Equations by the Finite Element Method*. Springer.
- Logg, A., Ølgaard, K. B., Rognes, M. E., and Wells, G. N. (2012b). *FFC: the FEniCS Form Compiler*, chapter 11. Springer.
- Logg, A. and Wells, G. N. (2010). Dofin: Automated finite element computing. *ACM Transactions on Mathematical Software*, 37(2).
- Love, A. E. H. (1888). On the small free vibrations and deformations of elastic shells. *Philosophical trans. of the Royal Society (London)*, N17:491–549.
- Lubkoll, L., Schiela, A., and Weiser, M. (2012). An optimal control problem in polyconvex hyperelasticity. Technical report, Konrad-Zuse-Zentrum für Informationstechnik Berlin.
- Meyer, A. (2007). Grundgleichungen und adaptive Finite-Elemente-Simulation bei “großen deformationen”. *Chemnitz Scientific Computing Preprints*.
- Mindlin, R. D. (1945). Influence of rotatory inertia and shear on flexural motions of isotropic elastic plate. *J. Appl. Mech.*, 18:31–38.



- Nagaiah, C., Kunisch, K., and Plank, G. (2013). Optimal control approach to termination of re-entry waves in cardiac electrophysiology. *Journal of Mathematical Biology*, 67(2):359–388.
- Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782.
- Nocedal, J. and Wright, S. (2006). *Numerical Optimization*. Springer, New York, second edition.
- Ogden, R. W. (1972). Large deformation isotropic elasticity: On the correlation of theory and experiment for compressible rubber-like solids. *Mathematical Proceedings of the Cambridge Philosophical Society*, A328:567–583.
- Ogden, R. W. (1984). *Non-Linear Elastic Deformations*. Dover Publications, Mineola, New York.
- Ospald, F. (2012). Implementation of a geometric multigrid method for FEniCS and its application. Diploma thesis, Technische Universität Chemnitz, Germany.
- Ospald, F. (2014). FMG. <http://www.launchpad.net/fmg>.
- Ovtchinnikov, E. E. and Xanthis, L. S. (1995). Effective dimensional reduction for elliptic problems. *Comptes Rendus de l'Académie des Sciences. Série I. Mathématique*, 320(7):879–884.
- Paige, C. and Saunders, M. (1975). Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629.
- Paumier, J.-C. and Raoult, A. (1997). Asymptotic consistency of the polynomial approximation in the linearized plate theory. Application to the Reissner-Mindlin model. In *Élasticité, viscoélasticité et contrôle optimal (Lyon, 1995)*, volume 2 of *ESAIM Proc.*, pages 203–213 (electronic). Soc. Math. Appl. Indust., Paris.
- Penzler, P., Rumpf, M., and Wirth, B. (2012). A phase-field model for compliance shape optimization in nonlinear elasticity. *ESAIM. Control, Optimisation and Calculus of Variations*, 18(1):229–258.
- Pillwein, V. and Takacs, S. (2014). A local Fourier convergence analysis of a multigrid method using symbolic computation. *Journal of Symbolic Computation*, 63:1–20.
- Plato, R. (2004). *Numerische Mathematik kompakt*. Vieweg, Wiesbaden, second edition.
- Raven, P., Evert, R., and Eichhorn, S. (2001). *Biology of Plants*. W.H. Freeman and Company Publishers, New York, 7th edition.



- Reissner, E. (1944). On the theory of bending of elastic plates. *Journal of Mathematics and Physics*, 23:184–191.
- Reissner, E. (1945). The effect of transverse shear deformation on the bending of elastic plates. *J. Appl. Mech.*, 12:A–69–A–77.
- Rückert, J. (2013). Kirchhoff plates and large deformations – modelling and  $C^1$ -continuous discretization. <http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-121275>.
- Rumpf, M. and Wirth, B. (2009). A nonlinear elastic shape averaging approach. *SIAM Journal on Imaging Sciences*, 2(3):800–833.
- Rumpf, M. and Wirth, B. (2011). An elasticity-based covariance analysis of shapes. *International Journal of Computer Vision*, 92(3):281–295.
- Salenko, J. (2001). *Handbook of Continuum Mechanics*. Springer, Berlin Heidelberg.
- Schöberl, J., Simon, R., and Zulehner, W. (2011). A robust multigrid method for elliptic optimal control problems. *SIAM Journal on Numerical Analysis*, 49(4):1482–1503.
- Schwab, C. and Wright, S. (1995). Boundary layers of hierarchical beam and plate models. *Journal of Elasticity. The Physical and Mathematical Science of Solids*, 38(1):1–40.
- Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24:647–656.
- Simo, J. C. and Hughes, T. J. R. (1998). *Computational Inelasticity*. Springer, New York.
- Sokołowski, J. and Zolésio, J.-P. (1992). *Introduction to Shape Optimization*. Springer, New York.
- Spencer, A. J. M. (2004). *Continuum mechanics*. Dover Publications Inc., Mineola, NY.
- Totland, Ö. (1996). Flower heliotropism in an alpine population of ranunculus acris (ranunculaceae): Effects on flower temperature, insect visitation, and seed production. *American Journal of Botany*, 83(4):452–458.
- Trottenberg, U., Oosterlee, C. W., and Schüller, A. (2001). *Multigrid*. Academic Press Inc., San Diego, CA. With contributions by A. Brandt, P. Oswald and K. Stüben.



- Turner, P. R. and Huntley, E. (1976). Variable metric methods in Hilbert space with applications to control problems. *Journal of Optimization Theory and Applications*, 19(3):381–400.
- Turner, P. R. and Huntley, E. (1977). Direct-prediction quasi-Newton methods in Hilbert space with applications to control problems. *Journal of Optimization Theory and Applications*, 21(2):199–211.
- Wachsmuth, G. (2012a). Optimal control of quasistatic plasticity with linear kinematic hardening, part I: Existence and discretization in time. *SIAM Journal on Control and Optimization*, 50(5):2836–2861.
- Wachsmuth, G. (2012b). Optimal control of quasistatic plasticity with linear kinematic hardening, part II: Regularization and differentiability. *submitted*.
- Weiser, M., Deuffhard, P., and Erdmann, B. (2007). Affine conjugate adaptive Newton methods for nonlinear elastomechanics. *Optimization Methods & Software*, 22(3):413–431.
- Weisstein, E. W. (2014). Legendre-Gauss quadrature. from Mathworld—A Wolfram Web Resource. <http://mathworld.wolfram.com/Legendre-GaussQuadrature.html>.
- Wesseling, P. (1992). *An introduction to multigrid methods*. Pure and Applied Mathematics (New York). John Wiley & Sons Ltd., Chichester.
- Zeidler, E. (1986). *Nonlinear functional analysis and its applications. I*. Springer-Verlag, New York. Fixed-point theorems, Translated from the German by Peter R. Wadsack.
- Zeidler, E. (2013). *Springer-Taschenbuch der Mathematik*. Springer Spektrum, third edition.
- Zwillinger, D., editor (2003). *CRC standard mathematical tables and formulae*. Chapman & Hall/CRC, Boca Raton, FL, st edition.