Numerical Solution of Nonlinear Finite Element Problems in Elasticity

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktorin der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation (Monografie)

von

Eva Lina Fesefeldt

aus

Hannover

2026

1. Gutachterin: Prof. Dr. Sabine Le Borne
2. Gutachter: Prof. Dr. Alexander Düster

Tag der mündlichen Prüfung: 16. Februar 2026

DOI: 10.15480/882.16878

ORCiD: https://orcid.org/0009-0004-7257-2232

Summary

In this thesis, we model the behavior of a solid body under external loads. The deformation of the body depends on its material. We focus on elastic material models, which are used in deformation simulations of rubber or biological tissue. In general, the deformation of an elastic body is nonlinear with respect to the external pressure or traction.

Analytical solutions are rarely available for nonlinear elastic deformation problems. Therefore, iterative methods such as Newton's method are used to linearize the problem. High-order finite element discretizations (p-FEM) provide accurate numerical solutions to the deformation problem. Based on a benchmark problem in computational mechanics, we demonstrate that the computation time for p-FEM discretizations is dominated by the numerical integration and assembly of the global stiffness matrices. The convergence of Newton's method depends on an initial guess. Thus, to converge with large displacements, boundary conditions must sometimes be applied incrementally to ensure that the initial guess for Newton's method is good enough.

Building on the traditional incremental load step approach, we introduce a new approach that exploits a hierarchical high-order finite element discretization. Instead of increasing the load on the full model, we iterate on reduced-order models in early stages of the computation. Furthermore, the early load steps are solved with a relaxed tolerance for the termination criterion in Newton's method. We demonstrate that the new approach has the potential to reduce computation time to $40 - 60\%$ of the original CPU time, depending on the geometry of the problem.

Acknowledgements

I am fortunate to be surrounded by a supportive community, both academically and personally.

I thank my supervisor Sabine Le Borne for her invaluable feedback and guidance during my four years as a PhD student. I thank Alexander Düster and Lars Radtke for our regular meetings, their constructive feedback and their patience with my questions. Roman Sartorti, Christian Kühne, Mahan Gorji and Wadhah Garhuom for including me in the weekly `AdhoC++` developer meetings and helping me navigate the deep waters of the `AdhoC++` code base.

Thorben Abel, Jonas Grams and Michael Koch proofread parts of this thesis. More importantly, we shared countless office days and lunch breaks. Your presence made me feel at home in the institute.

My parents Bettina and Martin Fesefeldt always believe in my abilities and have supported me during my studies. Thank you for encouraging my curiosity, for teaching me to think critically and for always being there when I need you.

I thank my brother, Julius, and my sister, Greta, who always remind me that normal people don't study math. Julius also provided valuable advice regarding the graphical representations included in this thesis.

Special thanks go to my friends. To Janina and Doro for brightening my office days with wonderful lunch dates. To Birte and Eileen. You made my study years unforgettable, and living with you will always be one of my favorite memories. Thank you for being there during the lows and celebrating the highs with me.

And finally to Lars, my partner in crime. I am deeply grateful for all your listening, coffee-making, support and love. I can't count how many times you've made me laugh on a challenging day, and I am so lucky to have you by my side.

Contents

List of Abbreviations

Abbreviation	**Description**	**Page**
CSR	compressed sparse row-major format	75
DOF	degrees of freedom	29
FEM	finite element method	30
LSN	load step Newton method	68
PDE	partial differential equation	8
SLSN	surrogate load step Newton method	88

List of Symbols

Symbol	Description	Page
$H^1(\overline{\Omega};\mathbb{R}^3)$	Sobolev space of functions $f : \overline{\Omega} \to \mathbb{R}^3$ (all components and their respective first weak derivatives are in $L^2(\overline{\Omega};\mathbb{R})$)	19
$L^2(\overline{\Omega};\mathbb{R})$	space of measurable functions $f : \overline{\Omega} \to \mathbb{R}$ with $\int_\Omega f(\mathbf{x})^2 \, \mathrm{d}\mathbf{x} < \infty$	ix
Γ_1	part of the boundary with traction/pressure loads	6
Γ	boundary of Ω	4
$\mathbf{A} : \mathbf{B}$	Frobenius scalar product	9
$\mathbf{E}_{ext}$	external load vector	34
$\mathbf{E}_{int}$	internal load vector	34
$\mathbf{F}$	deformation gradient, Jacobian of the deformation	4
$\mathbf{K}$	global stiffness matrix	37
$\mathbf{P}$	first Piola-Kirchhoff stress tensor	8
$\mathbf{S}$	second Piola-Kirchhoff stress tensor	25
$\boldsymbol{\varphi}$	deformation of a three-dimensional solid body	4
$\mathbf{x} \cdot \mathbf{y}$	standard scalar product $\mathbf{x}^\mathrm{T}\mathbf{y}$	10
$\mathbb{M}^3_+$	matrices with positive determinant in $\mathbb{R}^{3\times3}$	54
$\mathbb{M}^3_{\mathrm{symm},+}$	symmetric matrices with positive determinant in $\mathbb{R}^{3\times3}$	54
$\mathbb{M}^3_{\mathrm{symm}}$	symmetric matrices in $\mathbb{R}^{3\times3}$	11
$\mathbb{M}^3$	$\mathbb{R}^{3\times3}$	11
$\mathcal{C}$	elasticity tensor of order 4	26
$\nabla\boldsymbol{\eta}$	Jacobian of a function $\boldsymbol{\eta} : \overline{\Omega} \to \mathbb{R}^3$	9
deg	degree of a polynomial (highest degree of its monomials with a non-zero coefficient)	38
$\otimes$	tensor product, dyadic product	12
$\overline{\Omega}$	domain in $\mathbb{R}^3$, represents a solid body	4

List of Figures

List of Tables

Chapter 1

Introduction

As computational resources grow, the potential of computer simulations increases. Yesterdays large-scale simulations run on an office computer today. The accessibility of accurate and efficient simulations allows engineers to explore a wide range of designs and materials. Additionally, replacing prototypes with computer simulations can be a step towards a sustainable future.

One type of simulation is the deformation analysis of solid structures under load, such as gravity, pressure or traction on the boundary of the structure. In this thesis, we focus on elastic materials. Elastic material laws are used to model biological materials such as soft tissue, as well as rubber-like solids. In general, there is no analytical solution available for the deformation. We therefore use the finite element method (FEM) to discretize the solid structure. With its roots in variational calculus, the finite element method has a solid mathematical foundation. It is a powerful and versatile tool for numerically solving differential equations and, about 80 years after its discovery, remains the method of choice for many computational problems in structural analysis.

To correctly predict the behavior of structures under load, it is often necessary to allow for nonlinear behavior of the material. An iterative method such as Newton's method is required to numerically solve the discretized equations of a nonlinear material model.

Some elastic deformation problems are ill-conditioned. For an ill-conditioned problem, decreasing the element size h (h-FEM) improves the approximation quality only slowly. While eventual convergence to the correct solution may still be guaranteed, the traditional h-FEM approach is therefore not feasible for ill-conditioned problems. This effect is called *locking*. High-order elements have been shown to prevent the undesirable locking effect. Increasing the polynomial degree of the basis functions instead of decreasing the element size is called p-version of the finite element method (or p-FEM).

This thesis addresses the high computational effort of Newton's method in high-order finite element methods for elastic problems. The goal is to answer the following questions:

- Which factors increase the computational effort?
- What is the bottleneck of the solution algorithm?
- Can we use reduced-order models to decrease computation time?

We review a benchmark problem in computational mechanics and demonstrate that, in the case of p-FEM, the set-up time for the system matrices in Newton's method dominates the computation time. The convergence of Newton's method depends on the initial guess. In case of divergence, the external loads are applied in increments to reach convergence (load step Newton method), which increases the computational burden.

We propose a modification to the traditional load step Newton method that allows to compute accurate numerical solutions for high-order finite element methods in reduced computation time. This new approach is based on the hierarchical basis functions that we use in the high-order discretization. Instead of incrementing the load on the full model, we use reduced-order models for some increments. This allows us to apply Newton's method to a small subsystem. For the final load increment, we switch to the full model. The new approach leads to a decrease in computation time while preserving the quality of the approximation. We also explore the possibility of relaxing the Newton tolerance on intermediate load steps, decreasing computation time even further.

This thesis gives a comprehensive overview of the concepts and ideas needed to understand the numerical solution of deformation problems of elastic bodies and is structured as follows. In Chapters 2, 3 and 5, we introduce the fundamental equations that connect the deformation with the geometry and material of an elastic body. Chapter 4 covers the discretization of the continuous deformation using finite elements. Chapter 6 introduces a benchmark problem in computational mechanics and motivates the use of high-order finite element methods. Chapter 7 introduces the new approach. Numerical tests are performed and discussed in Chapter 8, showing that the new approach reduces computation time for the benchmark problem as well as for a large-scale deformation problem.

Style and academic writing. The language and typesetting in this thesis follow the recommendations of Nicholas J. Higham's handbook [15]. References are given as specific as possible, for example by referring to a specific page. The cited textbooks are relevant to the topic also beyond the cited pages.

Chapter 2

Displacement Problems in Solid Mechanics

In this work, we focus on displacement problems. Two general examples are:

- We want to know how an elastic component of a machine (e.g., a sealing ring) deforms during the operation of a machine.
- We want to investigate the negative impacts of forces acting on a human body in a car accident.

Simulation-aided design provides valuable insights and improves design efficiency. In order to be a useful tool, simulations must be as accurate and efficient as possible.

We consider displacement problems in nonlinear materials. Our analysis is based on the concept of a continuum. A continuum is:

- A macroscopic model of a solid body.
- A connected domain in three-dimensional space.
- Bordered by a boundary that has a finite number of contiguous parts.

These statements already contradict the properties of any real-world object since objects are made of discrete particles. Additionally, we assume that:

- Small volumes inside the body have the same properties as the whole body – in particular, we should be able to apply forces to volumes of any size and measure their response.

Again, approaching the molecule level, this assumption is invalid. Despite these limitations, continuum modeling yields valuable predictions for real-world objects.

2.1 Deformations of a continuous body

In the following, we put a continuum and its deformations into mathematical notation. A one-body continuum is represented as the closure

$$\overline{\Omega} \subset \mathbb{R}^3$$

of an open, bounded subset Ω of $\mathbb{R}^3$.

The boundary $\Gamma = \partial\Omega$ is assumed to be piecewise Lipschitz-continuous. A deformation of Ω is an injective, differentiable mapping

$$\begin{aligned} \boldsymbol{\varphi} &: \Omega \to \mathbb{R}^3, \\ \boldsymbol{\varphi} &: \mathbf{x} \mapsto \boldsymbol{\varphi}(\mathbf{x}). \end{aligned}$$

As a justification for the assumption of injectivity, consider that a non-injective mapping would locally lead to infinite compression of the material which is physically impossible. Note that $\boldsymbol{\varphi}$ does not depend on time. We essentially work with two states here:

- The **initial configuration** Ω, the space that the body occupies at rest, i.e., in the absence of external forces like gravity or pressure.
- The deformed state $\boldsymbol{\varphi}(\Omega)$, called **deformed** or **current configuration**.

Therefore,

$$\boldsymbol{\varphi} = \begin{pmatrix} \varphi_1, & \varphi_2, & \varphi_3 \end{pmatrix} : \Omega \to \boldsymbol{\varphi}(\Omega) \subset \mathbb{R}^3$$

is an invertible mapping. We introduce the **deformation gradient**

$$\mathbf{F}(\mathbf{x}) := J_{\boldsymbol{\varphi}}(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial \varphi_1}{\partial x_1}(\mathbf{x}) & \dfrac{\partial \varphi_1}{\partial x_2}(\mathbf{x}) & \dfrac{\partial \varphi_1}{\partial x_3}(\mathbf{x}) \\ \dfrac{\partial \varphi_2}{\partial x_1}(\mathbf{x}) & \dfrac{\partial \varphi_2}{\partial x_2}(\mathbf{x}) & \dfrac{\partial \varphi_2}{\partial x_3}(\mathbf{x}) \\ \dfrac{\partial \varphi_3}{\partial x_1}(\mathbf{x}) & \dfrac{\partial \varphi_3}{\partial x_2}(\mathbf{x}) & \dfrac{\partial \varphi_3}{\partial x_3}(\mathbf{x}) \end{pmatrix} \tag{2.1}$$

of $\boldsymbol{\varphi}$ as the Jacobian of the deformation. We always assume that $\det(\mathbf{F}) > 0$.

When dealing with deformed bodies and variables defined with respect to them, we use the notation

$$\begin{aligned} \mathbf{x}^{\boldsymbol{\varphi}} &:= \boldsymbol{\varphi}(\mathbf{x}), \\ \Omega^{\boldsymbol{\varphi}} &:= \boldsymbol{\varphi}(\Omega), \\ \overline{\Omega}^{\boldsymbol{\varphi}} &:= \boldsymbol{\varphi}(\overline{\Omega}) \end{aligned}$$

as proposed in [8].

In this thesis, we focus on three-dimensional problems. Unless stated otherwise, all vectors (lowercase **bold** symbols) have three components, and indices range as $i = 1, 2, 3$.

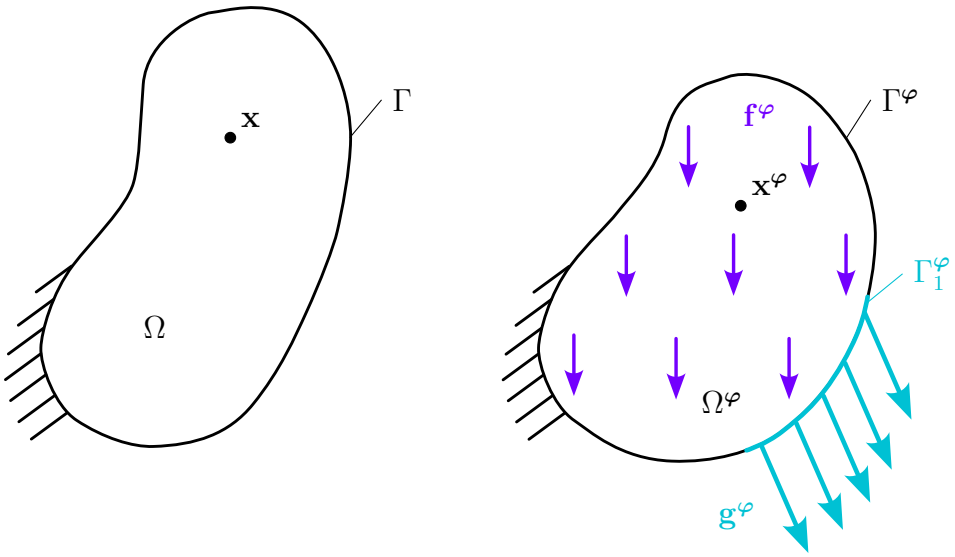


Figure 2.1: The body load $\mathbf{f}^{\boldsymbol{\varphi}}$ and the traction $\mathbf{g}^{\boldsymbol{\varphi}}$ result in a deformation $\boldsymbol{\varphi}$ of Ω. The body loads and traction are defined on the body under load $\Omega^{\boldsymbol{\varphi}}$ and its boundary $\Gamma^{\boldsymbol{\varphi}}$ instead of the original, undeformed body Ω. This is because the deformed body is in equilibrium with the external forces $\mathbf{f}^{\boldsymbol{\varphi}}$ and $\mathbf{g}^{\boldsymbol{\varphi}}$, or equivalently, both situations (left and right sub-figure) fulfill the axioms of force and moment balance. For small deformations, the simplification $\Omega^{\boldsymbol{\varphi}} = \Omega$ may be used. In this thesis, however, we always work with the general case of Ω being unknown. External loads that account for this general case and are defined in the deformed configuration are also called follower loads.

2.2 Force and moment balance

A continuum can be subjected to two types of external forces, see also Figure 2.1:

- body forces acting on the whole body, i.e., gravity and
- boundary forces, i.e., pressure or traction on the boundary of the body.

These external forces cause the deformation by propagating through the body. The propagation depends on the body's geometry and material properties. The internal stress caused by this is expressed in the Cauchy stress vector introduced in the following Proposition 2.1. Force and moment balance are fundamental axioms of mechanics. The following results are derived in [8]. In the proposition, all forces are given as densities (force per area). This is because traction and pressure act on subsets of the boundary with a surface area greater than zero since point loads are physically impossible. On the other hand, the total load is measured in experiments. Therefore, it makes sense to express the force as a density, since it can be easily obtained by dividing the applied force by the area.

Proposition 2.1 (Force and moment balance)**.** *Consider a body occupying a deformed configuration $\overline{\Omega}^{\boldsymbol{\varphi}}$ and subjected to applied forces represented by densities*

$$\mathbf{f}^{\boldsymbol{\varphi}} : \Omega^{\boldsymbol{\varphi}} \to \mathbb{R}^3$$
$$\mathbf{g}^{\boldsymbol{\varphi}} : {\Gamma_1}^{\boldsymbol{\varphi}} \to \mathbb{R}^3,$$

where $\Gamma_1^{\boldsymbol{\varphi}} \subset \Gamma^{\boldsymbol{\varphi}}$. Then there exists a vector field

$$\mathbf{t}^{\boldsymbol{\varphi}} : \overline{\Omega}^{\boldsymbol{\varphi}} \times S^2 \to \mathbb{R}^3$$

where $S^2 := \{\mathbf{x} \in \mathbb{R}^3, \|x\|_2 = 1\}$ denotes the unit sphere, such that:

- *For any subdomain $A^{\boldsymbol{\varphi}} \subset \overline{\Omega}^{\boldsymbol{\varphi}}$ and at any point $\mathbf{x}^{\boldsymbol{\varphi}} \in \Gamma_1^{\boldsymbol{\varphi}} \cap \partial A^{\boldsymbol{\varphi}}$ where the unit outer normal vector $\mathbf{n}^{\boldsymbol{\varphi}}$ to $\Gamma_1^{\boldsymbol{\varphi}} \cap \partial A^{\boldsymbol{\varphi}}$ exists, there holds*

$$\mathbf{t}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}, \mathbf{n}^{\boldsymbol{\varphi}}) = \mathbf{g}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}).$$

- *For any subdomain $A^{\boldsymbol{\varphi}} \subset \overline{\Omega}^{\boldsymbol{\varphi}}$ there holds the axiom of force balance:*

$$\int_{A^{\boldsymbol{\varphi}}} \mathbf{f}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}) \, \mathrm{d}\mathbf{x}^{\boldsymbol{\varphi}} + \int_{\partial A^{\boldsymbol{\varphi}}} \mathbf{t}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}, \mathbf{n}^{\boldsymbol{\varphi}}) \, \mathrm{da}^{\boldsymbol{\varphi}} = \mathbf{0} \tag{2.2}$$

  where $\mathbf{n}^{\boldsymbol{\varphi}}$ denotes the unit outer normal vector along $\partial A^{\boldsymbol{\varphi}}$.

- *For any subdomain $A^{\boldsymbol{\varphi}} \subset \overline{\Omega}^{\boldsymbol{\varphi}}$ there holds the axiom of moment balance:*

$$\int_{A^{\boldsymbol{\varphi}}} \mathbf{x}^{\boldsymbol{\varphi}} \times \mathbf{f}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}) \, \mathrm{d}\mathbf{x}^{\boldsymbol{\varphi}} + \int_{\partial A^{\boldsymbol{\varphi}}} \mathbf{x}^{\boldsymbol{\varphi}} \times \mathbf{t}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}, \mathbf{n}^{\boldsymbol{\varphi}}) \, \mathrm{da}^{\boldsymbol{\varphi}} = \mathbf{0} \tag{2.3}$$

  where $\times$ denotes the standard cross product.

  The vector field $\mathbf{t}^{\boldsymbol{\varphi}}$ is called ***Cauchy stress vector****.*

This axiom provides us with the stress vector $\mathbf{t}^{\boldsymbol{\varphi}} : \overline{\Omega}^{\boldsymbol{\varphi}} \times S^2 \to \mathbb{R}^3$, associating each pair of a point in the deformed configuration and normal vector $\mathbf{n}^{\boldsymbol{\varphi}}$ with a stress vector, see Figure 2.2. The direction and magnitude of that stress vector gives us a measure of the force density inside the continuum.

2.3 Stress tensors and equilibrium

One of the fundamental results of continuum mechanics is the following Theorem 2.2. It states that there exists a tensor field $\boldsymbol{\sigma}^{\boldsymbol{\varphi}} : \overline{\Omega}^{\boldsymbol{\varphi}} \to \mathbb{R}^{3\times 3}$ such that the stress vector can be expressed as

$$\mathbf{t}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}, \mathbf{n}) = \boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}})\mathbf{n}.$$

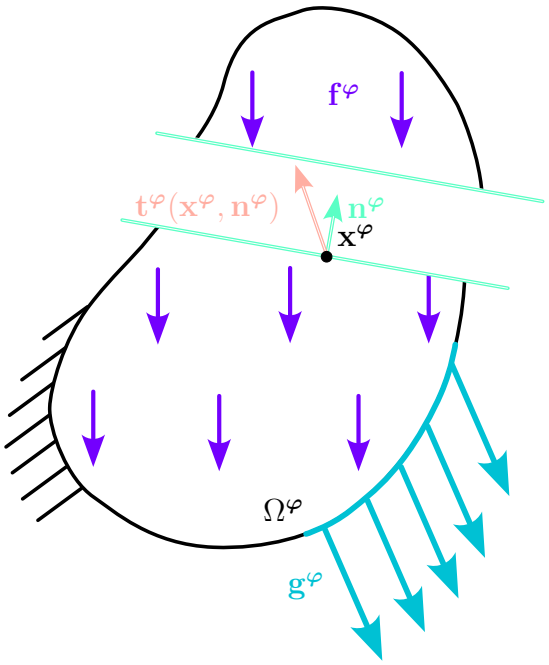


Figure 2.2: The stress vector at $\mathbf{x}^{\varphi}$ depends on the orientation of the cut surface.

The stress tensor $\boldsymbol{\sigma}^{\varphi}$ simplifies the stress analysis by providing complete information about the stress state of the body. In the following, we use the divergence operator in a row-wise sense, i.e., with the components $\boldsymbol{\sigma}^{\varphi} = (\boldsymbol{\sigma}_1^{\varphi}, \boldsymbol{\sigma}_2^{\varphi}, \boldsymbol{\sigma}_3^{\varphi})^{\mathrm{T}}$ of the stress tensor, the divergence operator in the deformed configuration is defined as

$$\operatorname{div}^{\varphi} \boldsymbol{\sigma}^{\varphi}(\mathbf{x}^{\varphi}) := \begin{pmatrix} \operatorname{div}^{\varphi} \boldsymbol{\sigma}_1^{\varphi}(\mathbf{x}^{\varphi}) \\ \operatorname{div}^{\varphi} \boldsymbol{\sigma}_2^{\varphi}(\mathbf{x}^{\varphi}) \\ \operatorname{div}^{\varphi} \boldsymbol{\sigma}_3^{\varphi}(\mathbf{x}^{\varphi}) \end{pmatrix}.$$

Theorem 2.2 (Cauchy's Theorem, [8, p. 62])**.** *Assume that the applied body force density* $\mathbf{f}^{\varphi} : \overline{\Omega}^{\varphi} \to \mathbb{R}^3$ *is continuous and that the Cauchy stress vector field*

$$\mathbf{t}^{\varphi} : \overline{\Omega}^{\varphi} \times S^2 \to \mathbb{R}^3, \quad (\mathbf{x}^{\varphi}, \mathbf{n}) \mapsto \mathbf{t}^{\varphi}(\mathbf{x}^{\varphi}, \mathbf{n})$$

is continuously differentiable with respect to the variable $\mathbf{n} \in S^2$ *for each* $\mathbf{x}^{\varphi} \in \overline{\Omega}^{\varphi}$*. Then the axioms of force and moment balance, Equations* (2.2) *and* (2.3)*, imply that there exists a continuously differentiable tensor field*

$$\boldsymbol{\sigma}^{\varphi} : \overline{\Omega}^{\varphi} \to \mathbb{R}^{3\times 3}, \quad \mathbf{x}^{\varphi} \mapsto \boldsymbol{\sigma}^{\varphi}(\mathbf{x}^{\varphi})$$

such that the Cauchy stress vector satisfies

$$\mathbf{t}^{\varphi}(\mathbf{x}^{\varphi}, \mathbf{n}) = \boldsymbol{\sigma}^{\varphi}(\mathbf{x}^{\varphi})\mathbf{n} \quad \forall \mathbf{x}^{\varphi} \in \overline{\Omega}^{\varphi}, \mathbf{n} \in S^2,$$

and such that

$$-\operatorname{div}^{\varphi} \boldsymbol{\sigma}^{\varphi}(\mathbf{x}^{\varphi}) = \mathbf{f}^{\varphi}(\mathbf{x}^{\varphi}) \quad \forall \mathbf{x}^{\varphi} \in \Omega^{\varphi}, \tag{2.4}$$

$$\boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}) = \boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}})^{\mathrm{T}}, \tag{2.5}$$

$$\boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}})\mathbf{n}^{\boldsymbol{\varphi}} = \mathbf{g}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}) \quad \forall \mathbf{x}^{\boldsymbol{\varphi}} \in \Gamma_1^{\boldsymbol{\varphi}}, \tag{2.6}$$

where $\mathbf{n}^{\boldsymbol{\varphi}}$ *is the unit outer normal vector along* $\Gamma_1^{\boldsymbol{\varphi}}$*. The three equations above are called* ***equations of balance in the deformed configuration****.*

Equation (2.4) is a partial differential equation (PDE) for the tensor field of interest. Equation (2.6) describes a traction boundary condition, it guarantees that the Cauchy stress vector in the direction of the outer normal vector of the boundary agrees with the prescribed pressure or traction $\mathbf{g}^{\boldsymbol{\varphi}}$.

Definition 2.3 (Cauchy stress tensor)**.** The tensor field $\boldsymbol{\sigma}^{\boldsymbol{\varphi}}$ in the previous theorem is called *Cauchy stress tensor field* which may be abbreviated to *Cauchy stress tensor* where the dependence on $\mathbf{x}^{\boldsymbol{\varphi}} \in \overline{\Omega}^{\boldsymbol{\varphi}}$ is included implicitly.

To recapitulate: Since the deformation depends on the way that external forces propagate through the body, it cannot be determined directly but only by introducing the concept of the stress vector and tensor. The stress tensor serves as a secondary unknown of the computation. For the stress tensor field $\boldsymbol{\sigma}^{\boldsymbol{\varphi}}$ on Ω, we have a PDE (2.4) introduced by Cauchy's Theorem 2.2. The governing PDE expresses the fundamental principle of force and moment balance. The domain $\overline{\Omega}^{\boldsymbol{\varphi}} = \boldsymbol{\varphi}(\overline{\Omega})$ of the secondary unknown $\boldsymbol{\sigma}^{\boldsymbol{\varphi}}$ directly depends on the primary unknown $\boldsymbol{\varphi}$. The following Piola transform is used to transform integrals from the current back to the initial configuration Ω which will result in a modified PDE in the initial configuration.

Definition 2.4 (Piola transform, first Piola-Kirchhoff stress tensor)**.** Let $\boldsymbol{\sigma}^{\boldsymbol{\varphi}} : \overline{\Omega}^{\boldsymbol{\varphi}} \to \mathbb{R}^{3\times 3}$ be the Cauchy stress tensor field for the deformation $\boldsymbol{\varphi} : \overline{\Omega} \to \overline{\Omega}^{\boldsymbol{\varphi}}$. Then the mapping

$$\mathbf{P} : \overline{\Omega} \to \mathbb{R}^{3\times 3} \tag{2.7}$$

$$\mathbf{P}(\mathbf{x}) := \det\big(\mathbf{F}(\mathbf{x})\big)\boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x}^{\varphi})\mathbf{F}(\mathbf{x})^{-\mathrm{T}} \tag{2.8}$$

with the deformation gradient $\mathbf{F}$ as defined in Equation (2.1) is called *Piola transform* of the Cauchy stress tensor field $\boldsymbol{\sigma}^{\boldsymbol{\varphi}}$. The tensor field $\mathbf{P}$ is called *first Piola-Kirchhoff stress tensor* at $\mathbf{x} \in \overline{\Omega}$.

Definition 2.5 (Body and surface loads in the initial configuration)**.** Assume we know the deformation to be caused by the body force field

$$\mathbf{f}^{\boldsymbol{\varphi}} : \Omega^{\boldsymbol{\varphi}} \to \mathbb{R}^3$$

and the surface force field

$$\mathbf{g}^{\boldsymbol{\varphi}} : \Gamma_1^{\boldsymbol{\varphi}} \to \mathbb{R}^3.$$

Then the vector field

$$\begin{aligned}\mathbf{f} &: \Omega \to \mathbb{R}^3,\\ \mathbf{f}(\mathbf{x}) &:= \det(\mathbf{F}(\mathbf{x}))\mathbf{f}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}})\end{aligned}$$

is called *body load in the initial configuration.*

For the surface loads, we assume that $\det(\mathbf{F}) \neq 0$ for all $\mathbf{x}^{\boldsymbol{\varphi}} \in \Gamma_1^{\boldsymbol{\varphi}}$ and set

$$\Gamma_1 := \boldsymbol{\varphi}^{-1}(\Gamma_1^{\boldsymbol{\varphi}}).$$

We define the *surface load on the initial configuration* as

$$\begin{aligned}\mathbf{g} &: \Gamma_1 \to \mathbb{R}^3,\\ \mathbf{g}(\mathbf{x}) &:= \det(\mathbf{F})\|\mathbf{F}^{-\mathrm{T}}\mathbf{n}\|_2\ \mathbf{g}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}).\end{aligned}$$

Defining the body loads on the initial configuration as above leads for every $A \subset \Omega$ to the relation

$$\int_A \mathbf{f}(\mathbf{x})\ \mathrm{d}\mathbf{x} = \int_{A^{\boldsymbol{\varphi}}} \mathbf{f}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}})\ \mathrm{d}\mathbf{x}^{\boldsymbol{\varphi}}$$

because of the transformation theorem, see Equation (A.2) in Appendix A. According to [8, p. 74], for all $A \subset \Gamma_1$ there holds

$$\int_A \mathbf{g}(\mathbf{x})\ \mathrm{da} = \int_{\boldsymbol{\varphi}(A)} \mathbf{g}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}})\ \mathrm{da}^{\boldsymbol{\varphi}}$$

for the surface loads in the initial configuration.

The stress tensor and the body and surface loads are now defined on the original domain Ω before deformation, see Figure 2.3. Using the Piola transform (Definition 2.4) and Cauchy's Theorem 2.2, it is possible to express the relation between external forces $\mathbf{f}, \mathbf{g}$ and the stress tensor field $\mathbf{P}$ without the dependence of the integration domain on the unknown deformation $\boldsymbol{\varphi}$. The following Theorem 2.8 uses the Frobenius scalar product.

Definition 2.6 (Frobenius scalar product)**.** Let $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m\times n}$ and $\mathbf{B} = (b_{ij}) \in \mathbb{R}^{m\times n}$. Then the *Frobenius scalar product* is defined as

$$\mathbf{A} : \mathbf{B} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ij}. \tag{2.9}$$

Remark 2.7 (Use of the symbol ∇). In the following balance equation, the Jacobian of a test function $\boldsymbol{\eta} : \overline{\Omega} \to \mathbb{R}^3$ is denoted as $\nabla\boldsymbol{\eta}$, so

$$(\nabla\boldsymbol{\eta}(\mathbf{x}))_{ij} := \frac{\partial \eta_i}{\partial x_j}(\mathbf{x}).$$

Theorem 2.8 (Balance equation, [8, p. 75]). *The first Piola-Kirchhoff stress tensor field satisfies the following equations in the initial configuration $\bar{\Omega}$:*

$$-\operatorname{div}\mathbf{P}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega, \tag{2.10}$$

$$\mathbf{F}(\mathbf{x})\mathbf{P}(\mathbf{x})^{\mathrm{T}} = \mathbf{P}(\mathbf{x})\mathbf{F}(\mathbf{x})^{\mathrm{T}}, \tag{2.11}$$

$$\mathbf{P}(\mathbf{x})\mathbf{n} = \mathbf{g}(\mathbf{x}) \quad \forall \mathbf{x} \in \Gamma_1. \tag{2.12}$$

The first and third equation are together equivalent to the variational equations

$$\int_\Omega \mathbf{P}(\mathbf{x}) : \nabla\boldsymbol{\eta}(\mathbf{x}) \, \mathrm{d}\mathbf{x} = \int_\Omega \mathbf{f}(\mathbf{x}) \cdot \boldsymbol{\eta}(\mathbf{x}) \, \mathrm{d}\mathbf{x} + \int_{\Gamma_1} \mathbf{g}(\mathbf{x}) \cdot \boldsymbol{\eta}(\mathbf{x}) \, \mathrm{da} \tag{2.13}$$

for all test functions $\boldsymbol{\eta} : \overline{\Omega} \to \mathbb{R}^3$ that vanish on $\Gamma \setminus \Gamma_1$, where $\cdot$ denotes the standard scalar product. In the following, we will not explicitly write the dependence on $\mathbf{x}$ anymore, so the equation reads

$$\int_\Omega \mathbf{P} : \nabla\boldsymbol{\eta} \, \mathrm{d}\mathbf{x} = \int_\Omega \mathbf{f} \cdot \boldsymbol{\eta} \, \mathrm{d}\mathbf{x} + \int_{\Gamma_1} \mathbf{g} \cdot \boldsymbol{\eta} \, \mathrm{da}.$$

The dependence on $\mathbf{x}$ is included implicitly. Furthermore, the first Piola-Kirchhoff stress tensor $\mathbf{P}$ depends on the unknown deformation $\boldsymbol{\varphi}$.

Equation (2.13) is also called "Principle of Virtual Work" or the "Weak Form of Equilibrium in the Reference Configuration" of the balance equation, for example in [33].

Core Concept 2.1: The balance equation for a continuous body under deformation

Body and surface loads cause a continuous body to deform and induce a stress state in the continuum $\overline{\Omega}$. The stress during the deformation $\boldsymbol{\varphi} : \overline{\Omega} \to \mathbb{R}^3$ is expressed by the Cauchy stress tensor

$$\boldsymbol{\sigma}^{\boldsymbol{\varphi}} : \overline{\Omega}^{\boldsymbol{\varphi}} \to \mathbb{R}^{3\times 3}$$

which is uniquely determined by Cauchy's Theorem 2.2.
Because of the dependence of these equations on the deformation $\boldsymbol{\varphi}$, the Piola transform is introduced to map all quantities of interest back to the original domain $\overline{\Omega}$. The first Piola-Kirchhoff stress tensor $\mathbf{P} : \overline{\Omega} \to \mathbb{R}^{3\times 3}$ is an analogous stress measure on the original domain, which is uniquely determined by the balance equation in the initial configuration:

$$\int_\Omega \mathbf{P} : \nabla\boldsymbol{\eta} \, \mathrm{d}\mathbf{x} = \int_\Omega \mathbf{f} \cdot \boldsymbol{\eta} \, \mathrm{d}\mathbf{x} + \int_{\Gamma_1} \mathbf{g} \cdot \boldsymbol{\eta} \, \mathrm{da} \quad \forall \boldsymbol{\eta},$$

where $\mathbf{F}(\mathbf{x})\mathbf{P}(\mathbf{x})^{\mathrm{T}} = \mathbf{P}(\mathbf{x})\mathbf{F}(\mathbf{x})^{\mathrm{T}}$ with the deformation gradient $\mathbf{F}(\mathbf{x})$, see Equation (2.1).

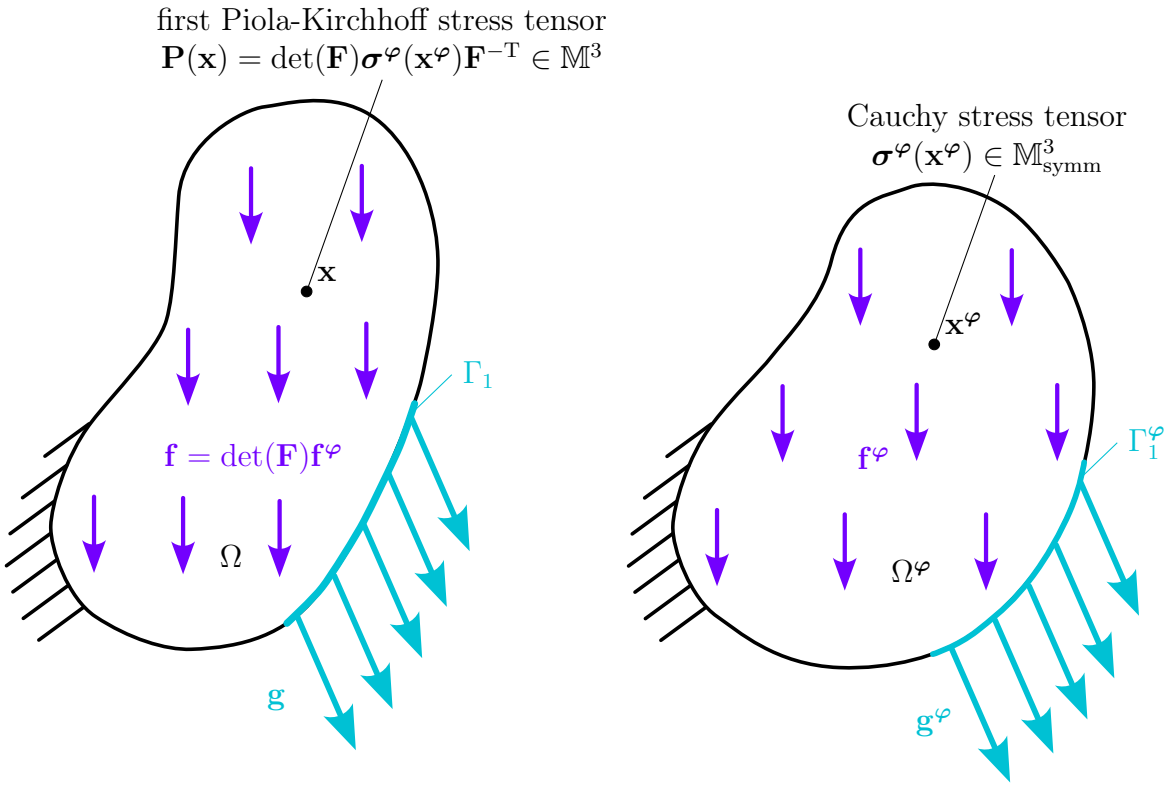


Figure 2.3: The stress tensor $\boldsymbol{\sigma}^{\varphi}$, surface and body loads $\mathbf{g}^{\varphi}, \mathbf{f}^{\varphi}$ in the deformed configuration, see the right sub-figure, are in balance (equilibrium). This is the fundamental statement of Cauchy's Theorem 2.2. Defining the first Piola-Kirchhoff stress tensor $\mathbf{P}$ and the body and surface loads $\mathbf{f}, \mathbf{g}$ creates the situation in the left sub-figure. Theorem 2.8 guarantees an equilibrium in the initial configuration.

2.4 Use of tensors

The theory of elasticity and the analysis of the balance equations use tensors. Until now, we used the word "tensor" as a synonym for matrix (e.g., for the stress tensor fields $\boldsymbol{\sigma}^{\varphi}$ and $\mathbf{P}$). Tensors can be seen as an extension of the matrix concept in the following sense. While a vector $\mathbf{v} \in \mathbb{R}^n$ has one index and a matrix $\mathbf{V} \in \mathbb{R}^{m \times n}$ has two indices, a general tensor has k indices:

$$\mathbf{T} \in \mathbb{R}^{n_1 \times \cdots \times n_k}, \quad n_1, \ldots, n_k \in \mathbb{N}, k \in \mathbb{N}.$$

To match the notation in [34], we use the following convention.

- The number of indices k is called the **order** of the tensor.
- A tensor with $k = 1$ is a vector.
- A tensor with $k = 2$ is a matrix.

A scalar can be seen as a tensor of order $k = 0$. We will write all tensors with order $k > 0$ in **bold**. Additionally, all tensors of order $k > 1$ are referred to with capital letters. The (scalar) components of a tensor $\mathbf{T}$ are written as $t_{i_1 i_2 \ldots i_k}$. To summarize, the convention for tensors is as follows:

	order	example symbol	components
scalar	$k = 0$	a	
vector	$k = 1$	$\mathbf{v}$	v_i
matrix	$k = 2$	$\mathbf{A}$	a_{ij}
general tensor	k	$\mathbf{T}$	$t_{i_1 i_2 \ldots i_k}$

Lemma 2.9 (Tensor space)**.** *The set $V = \mathbb{R}^{n_1 \times \cdots \times n_k}$ with scalar multiplication $\lambda \cdot \mathbf{T}, \lambda \in \mathbb{R}, \mathbf{T} \in V$ and component-wise addition is a vector space.*

We use the following definition to construct a basis of the tensor space.

Definition 2.10 (Tensor product)**.** Let $\mathbf{v}^{(j)} \in \mathbb{R}^{n_j}, 1 \leq j \leq k$. The *tensor product* $\otimes$ is defined as

$$\begin{aligned} \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k} &\to \mathbb{R}^{n_1 \times \cdots \times n_k} \\ (\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(k)}) &\mapsto \mathbf{v}^{(1)} \otimes \cdots \otimes \mathbf{v}^{(k)} := \mathbf{T} \end{aligned}$$

with $t_{i_1 i_2 \ldots i_k} = v_{i_1}^{(1)} \cdot \ldots \cdot v_{i_k}^{(k)}$.

Example 2.11. *Consider $k = 2, n_1 = 3$ and $n_2 = 2$. Then the tensor product is*

$$\mathbb{R}^3 \times \mathbb{R}^2 \to \mathbb{R}^{3\times 2}, \quad \mathbf{v} \otimes \mathbf{w} = \mathbf{v}\mathbf{w}^{\mathrm{T}}.$$

In particular, for $\mathbf{v} = \mathbf{e}_2^3 = (0,1,0)^{\mathrm{T}}$ and $\mathbf{w} = \mathbf{e}_1^2 = (1,0)^{\mathrm{T}}$ we obtain the rank 1 matrix

$$\mathbf{v}\mathbf{w}^{\mathrm{T}} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}$$

and for the remaining canonical basis vectors of $\mathbb{R}^3$ and $\mathbb{R}^2$

$$\mathbf{e}_1^3(\mathbf{e}_1^2)^{\mathrm{T}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{e}_3^3(\mathbf{e}_1^2)^{\mathrm{T}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix},$$

$$\mathbf{e}_1^3(\mathbf{e}_2^2)^{\mathrm{T}} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{e}_2^3(\mathbf{e}_2^2)^{\mathrm{T}} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{e}_3^3(\mathbf{e}_2^2)^{\mathrm{T}} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

We observe that

$$\operatorname{span}\{\mathbf{e}_1^3(\mathbf{e}_1^2)^{\mathrm{T}}, \mathbf{e}_2^3(\mathbf{e}_1^2)^{\mathrm{T}}, \mathbf{e}_3^3(\mathbf{e}_1^2)^{\mathrm{T}}, \mathbf{e}_1^3(\mathbf{e}_2^2)^{\mathrm{T}}, \mathbf{e}_2^3(\mathbf{e}_2^2)^{\mathrm{T}}, \mathbf{e}_3^3(\mathbf{e}_2^2)^{\mathrm{T}}\} = \mathbb{R}^{3\times 2}.$$

Remark 2.12. In the above example, the dimensions n_1, n_2 were included as superscripts to indicate which canonical basis vector was used. In the rest of the chapter, the dimensions are omitted because they are clear from the context.

Lemma 2.13 (Basis of the tensor space)**.** *The set of tensors*

$$\{\mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_k}, \quad 1 \le i_1 \le n_1, \ldots, 1 \le i_k \le n_k\},$$

where $\mathbf{e}_{i_j} \in \mathbb{R}^{n_j}$ is the i_j'th unit vector in $\mathbb{R}^{n_j}$, is a basis of $\mathbb{R}^{n_1\times\cdots\times n_k}$. The dimension of $\mathbb{R}^{n_1\times\cdots\times n_k}$ is

$$\prod_{j=1}^{k} n_j.$$

A tensor $\mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_k}$ is called a canonical tensor.

2.4.1 Tensor contractions

In the following, we introduce the multiplication of two tensors of different order, which is also called contraction. Later, in the linearization and discretization of the first Piola-Kirchhoff stress tensor, we will need such a contraction of tensors of order $k = 4$ and $k = 2$. All definitions in this subsection lead up to that definition. Since the tensor space $\mathbb{R}^{n_1\times\cdots\times n_k}$ has a basis of canonical tensors, see Lemma 2.13, it is sufficient to define the contraction on canonical tensors and by linearity extend to general tensors.

Definition 2.14 (Contraction of a matrix ($k = 2$) and a vector ($k = 1$))**.** For $\mathbf{x} \in \mathbb{R}^{n_2}$ and $\mathbf{e}_{i_1} \in \mathbb{R}^{n_1}, \mathbf{e}_{i_2} \in \mathbb{R}^{n_2}$, define

$$\mathbb{R}^{n_1 \times n_2} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_1}, \tag{2.14}$$

$$(\mathbf{e}_{i_1} \otimes \mathbf{e}_{i_2})\mathbf{x} := (\mathbf{x} \cdot \mathbf{e}_{i_2})\mathbf{e}_{i_1}. \tag{2.15}$$

Example 2.11 (continued). *For the contraction of the second-order canonical tensor* $\mathbf{e}_2 \otimes \mathbf{e}_1$ *with* $\mathbf{x} \in \mathbb{R}^2$ *we compute*

$$(\mathbf{e}_2 \otimes \mathbf{e}_1)\mathbf{x} = \left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ x_1 \\ 0 \end{pmatrix}.$$

Note that this agrees with the result of the matrix-vector product of $\mathbf{e}_2 \otimes \mathbf{e}_1$ *and* $\mathbf{x}$*.*

Furthermore, we can express any tensor $\mathbf{T} \in \mathbb{R}^{3\times 2}$ as a linear combination

$$\mathbf{T} = \sum_{i_1=1}^{3} \sum_{i_2=1}^{2} t_{i_1 i_2}\ \mathbf{e}_{i_1} \otimes \mathbf{e}_{i_2}, \quad t_{i_1 i_2} \in \mathbb{R}$$

and thus introduce the **linear extension** of the contraction as

$$\mathbf{T}\mathbf{x} := \sum_{i_1=1}^{3} \sum_{i_2=1}^{2} t_{i_1 i_2}\ (\mathbf{e}_{i_1} \otimes \mathbf{e}_{i_2})\mathbf{x}.$$

This definition agrees with the classical definition of the matrix-vector product.

Definition 2.15 (Contraction of a higher-order tensor ($k > 2$) and a vector)**.** For $\mathbf{x} \in \mathbb{R}^{n_k}$, we define the contraction with a tensor of order k as

$$\mathbb{R}^{n_1 \times \cdots \times n_k} \times \mathbb{R}^{n_k} \to \mathbb{R}^{n_1 \times \cdots \times n_{k-1}},$$
$$(\mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_k})\mathbf{x} := (\mathbf{x} \cdot \mathbf{e}_{i_k})(\mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_{k-1}}).$$

The contraction with arbitrary tensors is again obtained by linear extension.

Definition 2.16 (Contraction of a tensor of order $k = 4$ and a tensor of order $k = 2$)**.** Let $\mathcal{C}$ be a tensor of order $k = 4$. The contraction of $\mathcal{C}$ with a canonical tensor of order 2 is defined as

$$\mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4} \times \mathbb{R}^{n_3 \times n_4} \to \mathbb{R}^{n_1 \times n_2}$$
$$\mathcal{C}(\mathbf{e}_{i_3} \otimes \mathbf{e}_{i_4}) := (\mathcal{C}\mathbf{e}_{i_4})\mathbf{e}_{i_3}.$$

As before, the contraction is extended linearly to all second-order tensors.

2.4.2 Euclidean sum convention / Einstein convention

Any vector $\mathbf{v} \in \mathbb{R}^n$ and any matrix $\mathbf{S} \in \mathbb{R}^{m\times n}$ can be written as

$$\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{e}_i, \quad \mathbf{S} = \sum_{i=1}^{m}\sum_{j=1}^{n} s_{ij}\ \mathbf{e}_i \otimes \mathbf{e}_j.$$

We will sometimes use an equivalent short version of this notation:

$$\mathbf{v} = v_i \mathbf{e}_i, \quad \mathbf{S} = s_{ij}\ \mathbf{e}_i \otimes \mathbf{e}_j. \tag{2.16}$$

In this convention, an index appearing twice on the right-hand side of the equation means that we sum over that index. If not stated otherwise, the index ranges are $1 \leq i, j \leq 3$ since we work in three-dimensional space.

2.4.3 Frobenius scalar product

We highlight some properties of the Frobenius scalar product (Definition 2.6) that will be useful when working with stress and strain tensors in the next chapter.

A short computation shows that for any second-order tensors $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of appropriate sizes it holds that

$$\mathbf{AB} : \mathbf{C} = \mathbf{A}^{\mathrm{T}}\mathbf{C} : \mathbf{B}. \tag{2.17}$$

Let $\mathbf{u}, \mathbf{v}, \mathbf{x}, \mathbf{y} \in \mathbb{R}^3$. Then there holds

$$(\mathbf{u} \otimes \mathbf{v}) : (\mathbf{x} \otimes \mathbf{y}) = (\mathbf{u} \cdot \mathbf{x})(\mathbf{v} \cdot \mathbf{y}). \tag{2.18}$$

Lemma 2.17. *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ and $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$. Then it holds that*

$$\mathbf{A} : \mathbf{B} = \mathbf{A} : \frac{1}{2}(\mathbf{B} + \mathbf{B}^{\mathrm{T}}). \tag{2.19}$$

Proof.

$$\begin{aligned}
\mathbf{A} : \mathbf{B} &= \mathbf{A} : \left(\frac{1}{2}(\mathbf{B} + \mathbf{B}^{\mathrm{T}}) + \frac{1}{2}(\mathbf{B} - \mathbf{B}^{\mathrm{T}})\right) \\
&= \mathbf{A} : \frac{1}{2}(\mathbf{B} + \mathbf{B}^{\mathrm{T}}) + \frac{1}{2}\underbrace{(\mathbf{A} : \mathbf{B} - \mathbf{A} : \mathbf{B}^{\mathrm{T}})}_{=0 \text{ since } \mathbf{A}=\mathbf{A}^{\mathrm{T}}} \\
&= \mathbf{A} : \frac{1}{2}(\mathbf{B} + \mathbf{B}^{\mathrm{T}}).
\end{aligned}$$

□

Chapter 3

Iterative Methods for the Balance Equation

Finding the deformation $\boldsymbol{\varphi}$ of a body under given forces $\mathbf{g}, \mathbf{f}$ means solving the equation of balance (2.13). Note that the equation involves the Piola-Kirchhoff stress tensor $\mathbf{P}$ which depends on $\boldsymbol{\varphi}$. The relation between $\mathbf{P}$ and $\boldsymbol{\varphi}$ is generally nonlinear. In simulations of complex structures such as bridges or airplanes, analytical solutions for the deformation are rarely available. Instead, approximate solutions are obtained using nonlinear finite element methods. Compared to standard textbooks in nonlinear finite element methods such as [34], this chapter provides a newly structured overview of the linearization of the balance equation in the style of an abstract Galerkin discretization of a nonlinear variational problem as in [16]. The process is as follows:

1. Derive the equations of balance and write them as a root-finding problem $\mathbf{R}(\mathbf{x}) = \mathbf{0}$.

2. Apply Newton's method in a function space of admissible displacements. For this step, we need a directional derivative of the respective function $\mathbf{R}$. This step results in an update to $\boldsymbol{\varphi}$ in form of a continuous function and is also called linearization.

3. Discretize the continuous equations with finite elements and obtain a discrete update.

3.1 Newton's method

Consider a nonlinear root-finding problem

$$\mathbf{R}(\mathbf{x}) = \mathbf{0} \quad \text{for } \mathbf{R} : D \subset \mathbb{R}^n \to \mathbb{R}^n,\ n \in \mathbb{N}. \tag{3.1}$$

Given that $\mathbf{R}$ is differentiable and $\mathbf{x}_0 \in D$, we define its linearization at $\mathbf{x}_0$

$$\tilde{\mathbf{R}}(\mathbf{x}) := \mathbf{R}(\mathbf{x}_0) + \mathrm{D}\mathbf{R}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0), \tag{3.2}$$

where $\mathrm{D}\mathbf{R}(\mathbf{x}_0)$ denotes the Jacobian of $\mathbf{R}$ evaluated at $\mathbf{x}_0$. If $\mathrm{D}\mathbf{R}(\mathbf{x}_0)$ is regular, the root of the affine-linear function $\tilde{\mathbf{R}}$ is

$$\mathbf{x}_1 := \mathbf{x}_0 - (\mathrm{D}\mathbf{R}(\mathbf{x}_0))^{-1}\mathbf{R}(\mathbf{x}_0).$$

We hope that the root of the affine-linear function $\tilde{\mathbf{R}}(\mathbf{x})$ is closer to the solution of the original problem $\mathbf{R}(\mathbf{x}) = \mathbf{0}$ than $\mathbf{x}_0$. We can then define a new affine-linear function for $\mathbf{x}_1$, for $\mathbf{x}_2$, and so on. We define a sequence $(\mathbf{x}_k)_{k\in\mathbb{N}}$, given that $\mathrm{D}\mathbf{R}(\mathbf{x}_k)$ is regular for all $\mathbf{x}_k$:

$$\mathbf{x}_{k+1} := \mathbf{x}_k - (\mathrm{D}\mathbf{R}(\mathbf{x}_k))^{-1}\mathbf{R}(\mathbf{x}_k)$$

or

$$\text{Find } \mathbf{u}_k \text{ such that } \mathrm{D}\mathbf{R}(\mathbf{x}_k)\mathbf{u}_k = -\mathbf{R}(\mathbf{x}_k). \tag{3.3}$$

$$\text{Set } \mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{u}_k. \tag{3.4}$$

The method defined by this is called *Newton's method*.

3.2 Application to the balance equation

In this section, we demonstrate how to perform Newton's method in an appropriate function space. The method computes a sequence of deformations of our solid body $\overline{\Omega}$. The goal is for this sequence to converge to a deformation, $\boldsymbol{\varphi}$, for which the Piola-Kirchhoff stress tensor satisfies the variational equation (2.13).

Cauchy's Theorem 2.2 uses only one type of boundary condition: the traction/pressure boundary condition. For elastic body simulations, however, another type of boundary condition is essential. Typically, the deformation on one or several parts of the boundary is known a priori. It is physically impossible to define both the traction and the deformation on any part of the boundary. The boundary Γ of $\overline{\Omega}$ is therefore split into three parts:

- the fixed boundary Γ_0
- the traction/pressure boundary Γ_1
- and the free boundary $\Gamma \setminus (\Gamma_0 \cup \Gamma_1)$.

Let $\Gamma_0 \subset \Gamma$ be the finite union of connected subsets of the boundary:

$$\Gamma_0 = \bigcup_{k=2}^{n} \Gamma_k, \quad n \in \mathbb{N}, \Gamma_k \subset \Gamma \setminus \Gamma_1, \Gamma_k \text{ connected.}$$

Then, we can apply deformation boundary conditions on the fixed part of the boundary Γ_0 by defining a function

$$\overline{\boldsymbol{\varphi}} : \Gamma_0 \to \mathbb{R}^3.$$

In the remainder of this section, we assume that $\overline{\boldsymbol{\varphi}}$ is given. By construction, the fixed part of the boundary is distinct from the traction/pressure boundary Γ_1:

$$\Gamma_0 \cap \Gamma_1 = \varnothing.$$

The variational problem is posed in the Sobolev space $H^1(\overline{\Omega}; \mathbb{R}^3)$ as defined in [14, p. 122]. The formulation for deformation problems in solid mechanics can be found in [3, p. 3]. The set of admissible deformations is

$$V = \{\boldsymbol{\varphi} \in H^1(\overline{\Omega}; \mathbb{R}^3) \mid \boldsymbol{\varphi} = \overline{\boldsymbol{\varphi}} \text{ on } \Gamma_0\}. \tag{3.5}$$

The set V is not a vector space because it is not closed under addition. Let $\overline{\boldsymbol{\varphi}_0} \in V$ be an initial deformation that is admissible, i.e., it fulfills the deformation boundary conditions. Then, since

$$V = \overline{\boldsymbol{\varphi}_0} + \{\boldsymbol{\varphi} \in H^1(\overline{\Omega}; \mathbb{R}^3) \mid \boldsymbol{\varphi} = 0 \text{ on } \Gamma_0\}, \tag{3.6}$$

$$=: \overline{\boldsymbol{\varphi}_0} + U, \tag{3.7}$$

the set V is an affine space of the vector space U. The elements of U can be seen as corrections to the initial deformation $\overline{\boldsymbol{\varphi}_0}$. In practice, deformation boundary conditions are often rewritten as displacement boundary conditions.

Definition 3.1 (Displacement)**.** Let $\boldsymbol{\varphi} : \overline{\Omega} :\to \mathbb{R}^3$ be a deformation of a solid body $\overline{\Omega}$. Then the *displacement* of $\overline{\Omega}$ is defined as

$$\mathbf{u} : \overline{\Omega} \to \mathbb{R}^3, \ \mathbf{x} \mapsto \boldsymbol{\varphi}(\mathbf{x}) - \mathbf{x}.$$

Analogously, the displacement boundary condition is

$$\mathbf{u} : \Gamma_0 \to \mathbb{R}^3, \ \mathbf{x} \mapsto \overline{\boldsymbol{\varphi}}(\mathbf{x}) - \mathbf{x}.$$

Consider now the nonlinear variational problem (2.13). Defining the set of admissible test functions as

$$V_0 = \{\boldsymbol{\eta} \in H^1(\overline{\Omega}; \mathbb{R}^3) \mid \boldsymbol{\eta} = \mathbf{0} \text{ on } \boldsymbol{\Gamma} \setminus \boldsymbol{\Gamma_1}\} \tag{3.8}$$

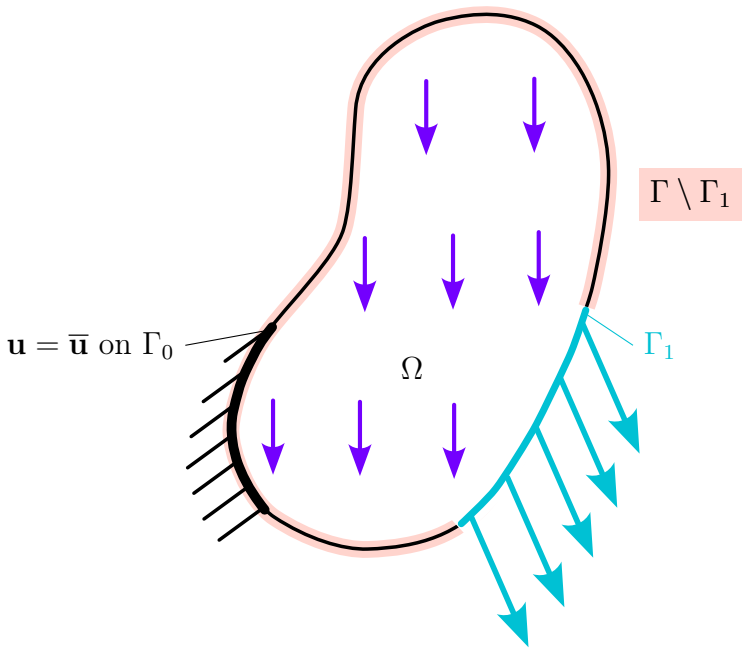


Figure 3.1: The displacement boundary condition enforces the displacement $\mathbf{u} = \bar{\mathbf{u}}$ (or equivalently, the deformation $\boldsymbol{\varphi} = \bar{\boldsymbol{\varphi}}$) on Γ_0, a part of the boundary that does not overlap with the traction boundary Γ_1. The test functions $\boldsymbol{\eta} \in V_0$ vanish on $\Gamma \setminus \Gamma_1$.

allows us to write the left-hand side of Equation (2.13) as

$$a : V \times V_0 \to \mathbb{R}, \quad a(\boldsymbol{\varphi}, \boldsymbol{\eta}) := \int_\Omega \mathbf{P} : \nabla \boldsymbol{\eta} \, \mathrm{d}\mathbf{x} \tag{3.9}$$

and the right-hand side as

$$\ell : V_0 \to \mathbb{R}, \quad \ell(\boldsymbol{\eta}) := \int_\Omega \mathbf{f} \cdot \boldsymbol{\eta} \, \mathrm{d}\mathbf{x} + \int_{\Gamma_1} \mathbf{g} \cdot \boldsymbol{\eta} \, \mathrm{da}. \tag{3.10}$$

Then the problem posed by the nonlinear variational equation is expressed as:

$$\text{Find } \boldsymbol{\varphi} \in V \text{ such that } a(\boldsymbol{\varphi}, \boldsymbol{\eta}) = \ell(\boldsymbol{\eta}) \ \forall \boldsymbol{\eta} \in V_0.$$

Note that the mapping a is linear in the second argument and ℓ is linear. We rewrite this as a root-finding problem for $\boldsymbol{\varphi}$:

$$\text{Find } \boldsymbol{\varphi} \in V \text{ such that } a(\boldsymbol{\varphi}, \boldsymbol{\eta}) - \ell(\boldsymbol{\eta}) = 0 \ \forall \boldsymbol{\eta} \in V_0. \tag{3.11}$$

To apply Newton's method, we must generalize the meaning of $\mathrm{D}\mathbf{R}(\mathbf{x})$ for functionals.

Directional derivatives measure the rate of change of a function

- at one point and
- in a given direction.

Definition 3.2 (Directional derivative)**.** Let V, W be vector spaces and $F : V \to W$. If the limit exists, we define the *directional derivative* of F at $\mathbf{x} \in V$ in the direction of $\mathbf{u} \in V$ as

$$\mathrm{D}F(\mathbf{x})[\mathbf{u}] := \lim_{\varepsilon \to 0} \frac{F(\mathbf{x} + \varepsilon \mathbf{u}) - F(\mathbf{x})}{\varepsilon} = \frac{\partial}{\partial \varepsilon} F(\mathbf{x} + \varepsilon \mathbf{u})|_{\varepsilon = 0}.$$

Lemma 3.3 (Linearity of the directional derivative [5, pp. 48])**.** *Let V, W be vector spaces and $\mathbf{x}, \mathbf{u}$ be elements of V. Let $F_1, F_2 : V \to W$. If the directional derivatives exist, there holds*

$$\mathrm{D}(F_1 + F_2)(\mathbf{x})[\mathbf{u}] = \mathrm{D}F_1(\mathbf{x})[\mathbf{u}] + \mathrm{D}F_2(\mathbf{x})[\mathbf{u}].$$

In Section 3.3, we will compute the directional derivative of the function $a(\boldsymbol{\varphi}, \boldsymbol{\eta})$ in Equation (3.9) in direction of an update $\mathbf{u}$. There, the integrand $\mathbf{P} : \nabla \boldsymbol{\eta}$ is rewritten using products of other second-order tensors. We therefore introduce a product rule for the directional derivative in the following Lemma 3.4. The first Piola-Kirchhoff stress tensor $\mathbf{P}$ is a tensor field on Ω. It depends on $\mathbf{x} \in \Omega$ and on the current deformation $\boldsymbol{\varphi}$. The factors in the matrix-matrix products are also second-order tensors depending on $\mathbf{x}$ and $\boldsymbol{\varphi}$. We use the short notation $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{n \times p}$ for the tensor fields. The dependence on $\mathbf{x}$ and $\boldsymbol{\varphi}$ is included implicitly.

Lemma 3.4 (Product rule for the matrix-matrix product)**.** *Let $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{n \times p}$. The directional derivative of the matrix-matrix product is*

$$\mathrm{D}\mathbf{A}\mathbf{B}[\mathbf{u}] = \mathbf{A}\ \mathrm{D}\mathbf{B}[\mathbf{u}] + \mathrm{D}\mathbf{A}[\mathbf{u}]\mathbf{B}.$$

Proof. Writing $\mathbf{A}$ and $\mathbf{B}$ in Euclidean sum convention (see Equation (2.16)), we have

$$\mathbf{A} = a_{ij}\ \mathbf{e}_i \otimes \mathbf{e}_j, \quad \mathbf{B} = b_{k\ell}\ \mathbf{e}_k \otimes \mathbf{e}_\ell.$$

The matrix-matrix product is

$$\begin{aligned} \mathbf{A}\mathbf{B} &= a_{ij} b_{k\ell} (\mathbf{e}_i \otimes \mathbf{e}_j)(\mathbf{e}_k \otimes \mathbf{e}_\ell) && (3.12) \\ &= a_{ij} b_{k\ell} \delta_{jk} (\mathbf{e}_i \otimes \mathbf{e}_\ell) && (3.13) \end{aligned}$$

where δ_{jk} denotes the Kronecker delta. Since $a_{ij} b_{k\ell} \delta_{jk} = a_{ik} b_{k\ell}$ holds, the matrix-matrix product simplifies to

$$\mathbf{A}\mathbf{B} = a_{ik} b_{k\ell}\ \mathbf{e}_i \otimes \mathbf{e}_\ell.$$

Therefore, the directional derivative of the matrix-matrix product is

$$\begin{aligned}\mathrm{D}\mathbf{AB}[\mathbf{u}] &= \mathrm{D}a_{ik}b_{k\ell}[\mathbf{u}]\ \mathbf{e}_i \otimes \mathbf{e}_\ell \\ &\overset{*}{=} (a_{ik}\mathrm{D}b_{k\ell}[\mathbf{u}] + \mathrm{D}a_{ik}[\mathbf{u}]b_{k\ell})\ \mathbf{e}_i \otimes \mathbf{e}_\ell \\ &= a_{ik}\ \mathrm{D}b_{k\ell}[\mathbf{u}]\ \mathbf{e}_i \otimes \mathbf{e}_\ell + \mathrm{D}a_{ik}[\mathbf{u}]b_{k\ell}\ \mathbf{e}_i \otimes \mathbf{e}_\ell\end{aligned}$$

where $(*)$ uses the scalar product rule for directional derivatives, which follows directly from the product rule for the derivative. We write the result as matrix-matrix products

$$\mathrm{D}\mathbf{AB}[\mathbf{u}] = \mathbf{A}\ \mathrm{D}\mathbf{B}[\mathbf{u}] + \mathrm{D}\mathbf{A}[\mathbf{u}]\mathbf{B}.$$

□

For $\mathbf{R}$ as defined in Equation (3.1), the directional derivative is computed by multiplication with the Jacobian:

$$\mathrm{D}\mathbf{R}(\mathbf{x})[\mathbf{u}] = \mathrm{D}\mathbf{R}(\mathbf{x})\mathbf{u}.$$

In this sense, the directional derivative is a generalization of the Jacobian to functions between arbitrary vector spaces. This motivates the use of a directional derivative in Newton's method, see Equation (3.3), for the function space V.

Applied to our root-finding problem in Equation (3.11) and with an initial guess $\boldsymbol{\varphi}_0 \in V$ given, we obtain for $k = 0, 1, \ldots$:

$$\text{Find } \mathbf{u}_k \in U \text{ such that } a(\boldsymbol{\varphi}_k, \boldsymbol{\eta}) + \mathrm{D}a(\boldsymbol{\varphi}_k, \boldsymbol{\eta})[\mathbf{u}_k] = \ell(\boldsymbol{\eta})\ \forall \boldsymbol{\eta} \in V_0, \tag{3.14}$$

$$\text{Set } \boldsymbol{\varphi}_{k+1} := \boldsymbol{\varphi}_k + \mathbf{u}_k. \tag{3.15}$$

This provides an iterative method and results in a sequence of admissible functions $\boldsymbol{\varphi}_k \in V$ if each subproblem, Equation (3.14), has a solution. The sub-problems are linear variational problems in $\mathbf{u}$:

$$\begin{aligned}&\text{Find } \mathbf{u} \in U \text{ such that } b(\mathbf{u}, \boldsymbol{\eta}) = h(\boldsymbol{\eta})\ \forall \boldsymbol{\eta} \in V_0 \\ &\text{with } b(\mathbf{u}, \boldsymbol{\eta}) := \mathrm{D}a(\boldsymbol{\varphi}_k, \boldsymbol{\eta})[\mathbf{u}], \quad h(\boldsymbol{\eta}) := \ell(\boldsymbol{\eta}) - a(\boldsymbol{\varphi}_k, \boldsymbol{\eta}).\end{aligned}$$

We use a Galerkin discretization for each subproblem. Let $V_0^N \subset V_0, U^N \subset U$ be subspaces with finite dimensions and $V^N = \overline{\boldsymbol{\varphi}_0} + U^N$. With $\boldsymbol{\varphi}_0^N \in V^N$ given, Newton's method in V^N for $k = 0, 1, \ldots$ is defined as follows:

$$\text{Find } \mathbf{u}_k^N \in U^N \text{such that } a(\boldsymbol{\varphi}_k^N, \boldsymbol{\eta}^N) + \mathrm{D}a(\boldsymbol{\varphi}_k^N, \boldsymbol{\eta}^N)[\mathbf{u}_k^N] = \ell(\boldsymbol{\eta}^N)\ \forall \boldsymbol{\eta}^N \in V_0^N, \tag{3.16}$$

$$\text{Set } \boldsymbol{\varphi}_{k+1}^N := \boldsymbol{\varphi}_k^N + \mathbf{u}_k^N. \tag{3.17}$$

Core Concept 3.1: Iterative method for the balance equation

The admissible deformations are differentiable and fulfill a displacement boundary condition that is given on $\Gamma_0 \subset \Gamma$. With $\overline{\boldsymbol{\varphi}_0} \in H^1(\overline{\Omega}; \mathbb{R}^3)$ given and satisfying the displacement boundary condition, the admissible deformations are $V = \overline{\boldsymbol{\varphi}_0} + U$ with a vector space U defined in Equation (3.6).
The space of admissible test functions V_0 is defined in Equation (3.8). Following Theorem 2.8, the test functions vanish on the part of the boundary where no traction (or pressure) boundary condition is given. With

$$a(\boldsymbol{\varphi}, \boldsymbol{\eta}) := \int_\Omega \mathbf{P} : \nabla \boldsymbol{\eta} \, \mathrm{d}\mathbf{x}, \quad \ell(\boldsymbol{\eta}) := \int_\Omega \mathbf{f} \cdot \boldsymbol{\eta} \, \mathrm{d}\mathbf{x} + \int_{\Gamma_1} \mathbf{g} \cdot \boldsymbol{\eta} \, \mathrm{da}$$

and by using a Galerkin discretization with finite-dimensional subspaces $V_0^N \subset V_0, U^N \subset U$ and $V^N = \overline{\boldsymbol{\varphi}_0} + U^N$, Newton's method in V^N is defined as follows: Let an admissible initial deformation $\boldsymbol{\varphi}_0^N \in V^N$ be given. Then the k'th Newton update is computed by:

Find $\mathbf{u}_k^N \in U^N$ such that $a(\boldsymbol{\varphi}_k^N, \boldsymbol{\eta}^N) + \mathrm{D}a(\boldsymbol{\varphi}_k^N, \boldsymbol{\eta}^N)[\mathbf{u}_k^N] = \ell(\boldsymbol{\eta}^N) \; \forall \boldsymbol{\eta}^N \in V_0^N$.
Set $\boldsymbol{\varphi}_{k+1}^N := \boldsymbol{\varphi}_k^N + \mathbf{u}_k^N$.

3.3 Directional derivative for the Newton step

While computing the directional derivative of a, Equation (3.9), needed in Newton's method, we introduce four important tensors: the right Cauchy-Green tensor, the Green-Lagrange strain tensor, the second Piola-Kirchhoff stress tensor, and the elasticity tensor. If possible, we give an intuition for their physical meanings.

Definition 3.5 (Right Cauchy-Green tensor)**.** The tensor

$$\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$$

with the deformation gradient $\mathbf{F}$ is called *right Cauchy-Green tensor.*

We consider the following example for intuition, illustrated in Figure 3.2.

Example 3.6. *Let the deformation be*

$$\begin{aligned} &\boldsymbol{\varphi} : [0,1]^2 \to \mathbb{R}^2, \\ &\boldsymbol{\varphi} : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_1 + \alpha x_2 \\ x_2 + \beta x_1 \end{pmatrix}. \end{aligned}$$

The deformation gradient is

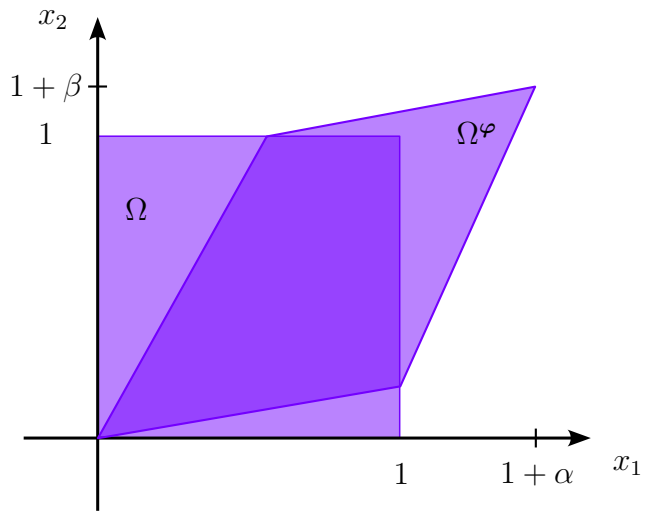


Figure 3.2: The domain and image of the deformation $\boldsymbol{\varphi}$ (initial and current configuration).

$$\mathbf{F} = \begin{pmatrix} 1 & \alpha \\ \beta & 1 \end{pmatrix}$$

and the change in volume from $[0,1]^2$ *to* $\boldsymbol{\varphi}([0,1]^2)$ *is* $\det \mathbf{F} = 1 - \alpha\beta$. *The right Cauchy-Green tensor is*

$$\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F} = \begin{pmatrix} \beta^2 + 1 & \alpha + \beta \\ \alpha + \beta & \alpha^2 + 1 \end{pmatrix}.$$

We observe that

$$\mathbf{x}^{\mathrm{T}}\mathbf{C}\mathbf{x} = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \mathbf{C} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (x_1 + \alpha x_2)^2 + (x_2 + \beta x_1)^2 = \|\boldsymbol{\varphi}(\mathbf{x})\|_2^2.$$

In this sense, $\mathbf{C}$ *measures the length of the vector* $\boldsymbol{\varphi}(\mathbf{x})$. *This is because the deformation is linear.*

Even for a nonlinear deformation, the term $\mathbf{x}^T\mathbf{C}\mathbf{x}$ still coincides with $\|\boldsymbol{\varphi}(\mathbf{x})\|_2^2$ in the first order term, see for example [7, p. 277]. The strain tensor indicates how vectors or line elements are strained when subjected to a deformation. Considering $\mathbf{C}$ for the example above, we observe a behavior that does not agree with our intuition of strain. In the trivial case $\alpha = \beta = 0$ we would expect the strain to be zero. This motivates the final definition of a strain tensor:

Definition 3.7 (Green-Lagrange strain tensor)**.** The tensor

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{I}\right)$$

with the identity tensor $\mathbf{I}$ is called *Green-Lagrange strain tensor.*

We illustrate how directional derivatives are computed using $\mathrm{D}\mathbf{E}[\boldsymbol{\eta}]$ as an example. We have

$$\mathrm{D}\mathbf{E}[\boldsymbol{\eta}] = \mathrm{D}\left(\frac{1}{2}(\mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{I})\right)[\boldsymbol{\eta}] = \frac{1}{2}\left(\mathrm{D}\mathbf{F}^{\mathrm{T}}[\boldsymbol{\eta}]\mathbf{F} + \mathbf{F}^{\mathrm{T}}\mathrm{D}\mathbf{F}[\boldsymbol{\eta}]\right)$$

using the linearity of the directional derivative and the product rule, see Lemma 3.4. Writing

$$\mathrm{D}\mathbf{F}(\boldsymbol{\varphi})[\boldsymbol{\eta}] = \frac{\partial}{\partial\varepsilon}\mathbf{F}(\boldsymbol{\varphi} + \varepsilon\boldsymbol{\eta})|_{\varepsilon=0}$$

shows that since $\mathbf{F}(\boldsymbol{\varphi} + \varepsilon\boldsymbol{\eta})$ is the Jacobian of the deformation $\boldsymbol{\varphi} + \varepsilon\boldsymbol{\eta}$,

$$\mathrm{D}\mathbf{F}(\boldsymbol{\varphi})[\boldsymbol{\eta}] = \nabla\boldsymbol{\eta}. \tag{3.18}$$

For the derivative of the Green-Lagrange strain in direction of $\boldsymbol{\eta}$, also called **variation of E**, we obtain

$$\mathrm{D}\mathbf{E}[\boldsymbol{\eta}] = \frac{1}{2}((\nabla\boldsymbol{\eta})^{\mathrm{T}}\mathbf{F} + \mathbf{F}^{\mathrm{T}}\nabla\boldsymbol{\eta}). \tag{3.19}$$

The variation is sometimes denoted as $\delta\mathbf{E}$, for example in [33]. The derivative in the direction of $\mathbf{u}$ is

$$\mathrm{D}\mathbf{E}[\mathbf{u}] = \frac{1}{2}((\nabla\mathbf{u})^{\mathrm{T}}\mathbf{F} + \mathbf{F}^{\mathrm{T}}\nabla\mathbf{u}). \tag{3.20}$$

Definition 3.8 (Second Piola-Kirchhoff stress tensor)**.** The tensor

$$\mathbf{S} = \mathbf{F}^{-1}\mathbf{P} \tag{3.21}$$

is called *second Piola-Kirchhoff stress tensor*.

Lemma 3.9. *The second Piola-Kirchhoff stress tensor is symmetric.*

Proof. With the definition of $\mathbf{P}$, Equation (2.7), we have

$$\begin{aligned}
\mathbf{S}(\mathbf{x})^{\mathrm{T}} &= (\mathbf{F}^{-1}(\mathbf{x})\mathbf{P}(\mathbf{x}))^{\mathrm{T}} \\
&= \det(\nabla\boldsymbol{\varphi}(\mathbf{x}))\left(\boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x})\mathbf{F}(\mathbf{x})^{-\mathrm{T}}\right)^{\mathrm{T}}\mathbf{F}^{-\mathrm{T}}(\mathbf{x}) \\
&= \det(\nabla\boldsymbol{\varphi}(\mathbf{x}))\mathbf{F}(\mathbf{x})^{-1}\boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x})\mathbf{F}^{-\mathrm{T}}(\mathbf{x}) \\
&= \mathbf{S}(\mathbf{x}),
\end{aligned}$$

where we have used the symmetry of $\boldsymbol{\sigma}^{\boldsymbol{\varphi}}$, see Equation (2.4). □

While the first Piola-Kirchhoff stress tensor relates the forces in the deformed state to those in the initial configuration, the second Piola-Kirchhoff stress tensor does not have a physical meaning. Nevertheless, it has two important roles:

- Since it is symmetric, using it to replace the first Piola-Kirchhoff stress saves memory in an implementation.
- To compute the directional derivative of $\mathbf{P}$, we decompose it into $\mathbf{F}$ and $\mathbf{S}$.

We compute the directional derivative of a, Equation (3.9), in the direction of $\mathbf{u}$. For the elastic deformation there holds[1]

$$\mathrm{D}a(\boldsymbol{\varphi}, \boldsymbol{\eta})[\mathbf{u}] = \mathrm{D} \int_\Omega \mathbf{P} : \nabla \boldsymbol{\eta} \, \mathrm{d}\mathbf{x} \, [\mathbf{u}] = \int_\Omega \mathrm{D}\mathbf{P}[\mathbf{u}] : \nabla \boldsymbol{\eta} \, \mathrm{d}\mathbf{x}.$$

For the directional derivative of the second Piola-Kirchhoff stress tensor we obtain

$$\mathrm{D}\mathbf{P}[\mathbf{u}] = \mathrm{D}\mathbf{F}\mathbf{S}[\mathbf{u}] = \mathrm{D}\mathbf{F}[\mathbf{u}] \; \mathbf{S} + \mathbf{F} \; \mathrm{D}\mathbf{S}[\mathbf{u}]$$

using the product rule, see Lemma 3.4. The directional derivative of $\mathbf{F}$ is $\nabla \mathbf{u}$, see Equation (3.18), so we obtain

$$\mathrm{D}\mathbf{P}[\mathbf{u}] = \nabla \mathbf{u} \; \mathbf{S} + \mathbf{F} \; \mathrm{D}\mathbf{S}[\mathbf{u}].$$

We compute the directional derivative of the second Piola-Kirchhoff stress tensor as

$$\begin{aligned} \mathrm{D}\mathbf{S}_{ij}[\mathbf{u}] &= \frac{\partial \mathbf{S}_{ij}(\mathbf{E}(\boldsymbol{\varphi} + \varepsilon \mathbf{u}))}{\partial \varepsilon}\Big|_{\varepsilon=0} \overset{(\mathrm{A}.3)}{=} \sum_{k=1}^{3} \sum_{l=1}^{3} \frac{\partial \mathbf{S}_{ij}}{\partial \mathbf{E}_{kl}} \frac{\partial \mathbf{E}_{kl}(\boldsymbol{\varphi} + \varepsilon \mathbf{u})}{\partial \varepsilon}\Big|_{\varepsilon=0} \\ &= \sum_{k=1}^{3} \sum_{l=1}^{3} \frac{\partial \mathbf{S}_{ij}}{\partial \mathbf{E}_{kl}} \mathrm{D}\mathbf{E}_{kl}[\mathbf{u}] \end{aligned}$$

where $\mathbf{E}$ is the Green-Lagrange strain, see [5, p. 160]. The directional derivative of $\mathbf{S}$ can therefore be written as a double contraction of the directional derivative of the Green-Lagrange strain $\mathrm{D}\mathbf{E}[\mathbf{u}]$ with the fourth-order *elasticity tensor* $\mathcal{C}$ defined over its components:

$$\mathcal{C}_{ijkl} := \frac{\partial \mathbf{S}_{ij}}{\partial \mathbf{E}_{kl}}.$$

Since $\mathbf{E}$ directly depends on the right Cauchy-Green tensor $\mathbf{C}$, we compute

$$\mathcal{C}_{ijkl} = \frac{\partial \mathbf{S}_{ij}}{\partial \mathbf{C}_{kl}} \frac{\partial \mathbf{C}_{kl}}{\partial \mathbf{E}_{kl}} = 2 \frac{\partial \mathbf{S}_{ij}}{\partial \mathbf{C}_{kl}}.$$

In conclusion, there holds

$$\mathrm{D}\mathbf{S}[\mathbf{u}] = \mathcal{C}\mathrm{D}\mathbf{E}[\mathbf{u}] \text{ with } \mathcal{C} = 2 \frac{\partial \mathbf{S}}{\partial \mathbf{C}}.$$

[1] In the case of conservative loads, reversing the order of the integral and the directional derivative is allowed, see [34, Equation 3.338]. Conservative loads preserve deformation energy.

For the directional derivative of a, we obtain

$$\begin{aligned} \mathrm{D}a(\boldsymbol{\varphi}, \boldsymbol{\eta})[\mathbf{u}] &= \int_\Omega (\nabla \mathbf{u}\mathbf{S} + \mathbf{F}\mathcal{C}\mathrm{D}\mathbf{E}[\mathbf{u}]) : \nabla \boldsymbol{\eta} \, \mathrm{d}\mathbf{x} \\ &= \int_\Omega \nabla \mathbf{u}\mathbf{S} : \nabla \boldsymbol{\eta} + \mathbf{F}\mathcal{C}\mathrm{D}\mathbf{E}[\mathbf{u}] : \nabla \boldsymbol{\eta} \, \mathrm{d}\mathbf{x} \\ &\overset{(2.17)}{=} \int_\Omega \nabla \mathbf{u}\mathbf{S} : \nabla \boldsymbol{\eta} + \mathcal{C}\mathrm{D}\mathbf{E}[\mathbf{u}] : \mathbf{F}^{\mathrm{T}} \nabla \boldsymbol{\eta} \, \mathrm{d}\mathbf{x}. \end{aligned}$$

The tensor contraction $\mathcal{C}\mathrm{D}\mathbf{E}[\mathbf{u}]$ yields a symmetric matrix. This follows from a short calculation, using that $\mathbf{S}$ is symmetric. With Lemma 2.17, we rewrite the integral as

$$\mathrm{D}a(\boldsymbol{\varphi}, \boldsymbol{\eta})[\mathbf{u}] = \int_\Omega \nabla \mathbf{u}\mathbf{S} : \nabla \boldsymbol{\eta} + \mathcal{C}\mathrm{D}\mathbf{E}[\mathbf{u}] : \frac{1}{2}(\mathbf{F}^{\mathrm{T}} \nabla \boldsymbol{\eta} + (\nabla \boldsymbol{\eta})^{\mathrm{T}} \mathbf{F}) \, \mathrm{d}\mathbf{x}. \tag{3.22}$$

The last term is the directional derivative of $\mathbf{E}$, see Equation (3.20):

$$\mathrm{D}a(\boldsymbol{\varphi}, \boldsymbol{\eta})[\mathbf{u}] = \int_\Omega \nabla \mathbf{u}\mathbf{S} : \nabla \boldsymbol{\eta} + \mathcal{C}\mathrm{D}\mathbf{E}[\mathbf{u}] : \mathrm{D}\mathbf{E}[\boldsymbol{\eta}] \, \mathrm{d}\mathbf{x}. \tag{3.23}$$

This concludes the computation of the directional derivative.

Chapter 4

Discretization

We use high-order hierarchical finite elements to discretize the displacement problem in space. In the previous chapter, we saw that the admissible displacements are functions in $H^1(\overline{\Omega};\mathbb{R}^3)$ that vanish on Γ_0, the part of the boundary where displacement boundary conditions are given, see Equation (3.6). We now assume that for each component of the displacement, which is a function in $H^1(\overline{\Omega};\mathbb{R})$, a finite basis

$$\mathcal{B} = \{N_i : \overline{\Omega} \to \mathbb{R}, \quad i = 1, \dots, n \text{ with } N_i = 0 \text{ on } \Gamma_0\} \tag{4.1}$$

is given. Then, the space of admissible displacements is

$$U^N = \operatorname{span}\{N_i \mathbf{e}_j \mid i = 1, \dots, n \text{ and } j = 1, 2, 3\} \tag{4.2}$$

and the dimension of U^N is $\dim(U^N) = 3n$, which implies that for $\mathbf{u} \in U^N$ there exist coefficients $u_{ij} \in \mathbb{R}, i = 1, \dots, n$ and $j = 1, 2, 3$, such that

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{n} u_{i,1} N_i(\mathbf{x}) \mathbf{e}_1 + \sum_{i=1}^{n} u_{i,2} N_i(\mathbf{x}) \mathbf{e}_2 + \sum_{i=1}^{n} u_{i,3} N_i(\mathbf{x}) \mathbf{e}_3$$

or

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{n} \begin{pmatrix} u_{i,1} \\ u_{i,2} \\ u_{i,3} \end{pmatrix} N_i(\mathbf{x}) =: \sum_{i=1}^{n} \mathbf{u}_i N_i(\mathbf{x}). \tag{4.3}$$

The vectors $\mathbf{u}_i \in \mathbb{R}^3, i = 1, \dots, n$ hold the coefficients from the linear combination of the basis. The coefficients are also called degrees of freedom (DOF).

In Section 3.2, we derived a formula for discrete Newton steps for the balance equation, see Core Concept 3.1 on page 23. Assuming that both the displacement and the test function are linear combinations of a finite number of basis functions, we analyze the expressions $a(\boldsymbol{\varphi}_k^N, \boldsymbol{\eta}^N)$, $\mathrm{D}a(\boldsymbol{\varphi}_k^N, \boldsymbol{\eta}^N)[\mathbf{u}_k^N]$ and $\ell(\boldsymbol{\eta}^N)$. This will lead to a system of nonlinear equations.

The main idea of the Finite Element Method (FEM) is to break Ω down into small elements of the same type of shape and limit the support of a basis function to one element (and potentially its neighbors). Usually, we choose triangles (tetrahedra in three dimensions) or rectangles (hexahedra in three dimensions) as elements.

Definition 4.1 (Mesh)**.** Let $\mathcal{M} = \{K_j\}_{i=1}^n, n \in \mathbb{N}$ be a numbered set of hexahedra K_j with the following properties.

1. All hexahedra K_j are open.
2. The union of the (closed) hexahedra yields the entire domain:
$$\bigcup_{i=1}^{n} \overline{K_j} = \overline{\Omega}.$$
3. The hexahedra have disjoint interiors:
$$K_i \cap K_j = \varnothing \text{ for } i, j \in \{1, \dots, n\} \text{ with } i \neq j.$$
4. Every intersection $\overline{K_i} \cap \overline{K_j}, i \neq j$ is exactly one of the following:
   - An edge of $\overline{K_i}$ and an edge of $\overline{K_j}$.
   - A vertex of $\overline{K_i}$ and a vertex of $\overline{K_j}$.
   - A face of $\overline{K_i}$ and a face of $\overline{K_j}$.
   - Empty.

Then $\mathcal{M}$ is called a *mesh* of the domain Ω. The hexahedra are called *elements* of the mesh.

We use sets of basis functions N_i that have a small support. In Section 4.3, we construct candidates for basis functions that are local to one element. Later, we will see that since this conflicts with inter-element continuity, basis functions should be allowed to be non-zero on neighboring elements. Therefore, the number of degrees of freedom that have an impact on the element depends on the geometry of the mesh. For an element $e \in \mathcal{M}$, let the indices of the non-vanishing basis functions be

$$N_e := \{i = 1, \dots, n \mid \operatorname{supp}(N_i) \cap e \neq \varnothing\}$$

such that

$$\mathbf{u}(\mathbf{x})|_e = \sum_{i \in N_e} N_i(\mathbf{x}) \mathbf{u}_i.$$

To obtain formulas for the discrete Newton step which we can use in an implementation, we plug the representations of $\mathbf{u}$ and $\boldsymbol{\eta}$ as linear combinations into Equation (3.16). To indicate that the displacement and test function now lie in finite-dimensional subspaces, we write $\mathbf{u}^N$ for the displacement and $\boldsymbol{\eta}^N$ for the test function. We closely follow the notation in [34, pp. 123].

4.1 Galerkin discretization of the balance equation

We assume that the displacement is given by the linear combination in Equation (4.3). Selecting differentiable basis functions ensures that

$$\nabla \mathbf{u}^N(\mathbf{x}) = \sum_{i=1}^{n} \nabla \left(N_i(\mathbf{x}) \mathbf{u}_i \right) = \sum_{i=1}^{n} \nabla \left(N_i(\mathbf{x}) \begin{pmatrix} u_{i,1} \\ u_{i,2} \\ u_{i,3} \end{pmatrix} \right)$$

with

$$\nabla N_i(\mathbf{x}) \mathbf{u}_i = \begin{pmatrix} \frac{\partial N_i(\mathbf{x})}{\partial x_1} u_{i,1} & \frac{\partial N_i(\mathbf{x})}{\partial x_2} u_{i,1} & \frac{\partial N_i(\mathbf{x})}{\partial x_3} u_{i,1} \\ \frac{\partial N_i(\mathbf{x})}{\partial x_1} u_{i,2} & \frac{\partial N_i(\mathbf{x})}{\partial x_2} u_{i,2} & \frac{\partial N_i(\mathbf{x})}{\partial x_1} u_{i,2} \\ \frac{\partial N_i(\mathbf{x})}{\partial x_1} u_{i,3} & \frac{\partial N_i(\mathbf{x})}{\partial x_2} u_{i,3} & \frac{\partial N_i(\mathbf{x})}{\partial x_1} u_{i,3} \end{pmatrix},$$

so $\nabla \mathbf{u}^N(\mathbf{x})$ can be written as a dyadic product

$$\nabla N_i(\mathbf{x}) \mathbf{u}_i = \mathbf{u}_i \otimes \nabla N_i(\mathbf{x}). \tag{4.4}$$

4.1.1 Discretization of the internal virtual work

The term $a(\boldsymbol{\varphi}_k^N, \boldsymbol{\eta}^N)$ in Core Concept 3.1 represents the internal virtual work of the deformation. We discretize the term by plugging in the linear combination of $\boldsymbol{\varphi}_k^N$, Equation (4.3), and the analogously obtained linear combination

$$\boldsymbol{\eta}^N(\mathbf{x}) = \sum_{i=1}^{n} N_i(\mathbf{x}) \boldsymbol{\eta}_i, \tag{4.5}$$

where $\boldsymbol{\eta}_i \in \mathbb{R}^3$ for $i = 1, \ldots, n$. We will observe that the coefficient vectors can be pulled out of the equation. This will help us to eliminate them in a root-finding problem. We first observe that

$$\begin{aligned} \mathbf{P} : \nabla \boldsymbol{\eta} &\overset{(3.21)}{=} \mathbf{F}\mathbf{S} : \nabla \boldsymbol{\eta} \overset{(2.17)}{=} \mathbf{F}^{\mathrm{T}} \nabla \boldsymbol{\eta} : \mathbf{S} \\ &\overset{(2.19)}{=} \mathbf{S} : \frac{1}{2} \big(\mathbf{F}^{\mathrm{T}} \nabla \boldsymbol{\eta} + \nabla \boldsymbol{\eta}^{\mathrm{T}} \mathbf{F} \big), \end{aligned}$$

where we have used the symmetry of $\mathbf{S}$ in the last step. The last term is the variation of the Green-Lagrange strain $\delta \mathbf{E}$, see Equation (3.19), so

$$\mathbf{P} : \nabla \boldsymbol{\eta} = \mathbf{S} : \delta \mathbf{E}.$$

Analogous to the derivative of the displacement in Equation (4.4), the derivatives of the test functions are

$$\nabla \boldsymbol{\eta}^N(\mathbf{x}) = \sum_{i=1}^{n} \boldsymbol{\eta}_i \otimes \nabla N_i(\mathbf{x}).$$

For the variation of the Green-Lagrange strain, we conclude

$$\delta \mathbf{E}^N = \frac{1}{2} \sum_{i=1}^{n} \Big((\mathbf{F}^N)^{\mathrm{T}} (\boldsymbol{\eta}_i \otimes \nabla N_i) + (\nabla N_i \otimes \boldsymbol{\eta}_i) \mathbf{F}^N \Big). \tag{4.6}$$

The components of the tensor are retrieved as follows. For $1 \leq k, \ell \leq 3$,

$$\delta \mathbf{E}^N_{k\ell} = \mathbf{e}_k^{\mathrm{T}} \delta \mathbf{E}^N \mathbf{e}_\ell = \frac{1}{2} \sum_{i=1}^{n} \Big(\mathbf{e}_k^{\mathrm{T}} (\mathbf{F}^N)^{\mathrm{T}} (\boldsymbol{\eta}_i \otimes \nabla N_i) \mathbf{e}_\ell + \mathbf{e}_k^{\mathrm{T}} (\nabla N_i \otimes \boldsymbol{\eta}_i) \mathbf{F}^N \mathbf{e}_\ell \Big).$$

For the first part of the sum, using the definition of the tensor contraction, Equation (2.14), gives

$$\mathbf{e}_k^{\mathrm{T}} (\mathbf{F}^N)^{\mathrm{T}} (\boldsymbol{\eta}_i \otimes \nabla N_i) \mathbf{e}_\ell = \mathbf{e}_k^{\mathrm{T}} (\mathbf{F}^N)^{\mathrm{T}} (\mathbf{e}_\ell \cdot \nabla N_i) \boldsymbol{\eta}_i = (\mathbf{F}^N \mathbf{e}_k \cdot \boldsymbol{\eta}_i)(\mathbf{e}_\ell \cdot \nabla N_i),$$

and similarly the second term is simplified to

$$(\mathbf{F}^N \mathbf{e}_\ell \cdot \boldsymbol{\eta}_i)(\mathbf{e}_k \cdot \nabla N_i).$$

Using the Euclidean sum convention, we write the components of $\delta \mathbf{E}^N$ as

$$\delta \mathbf{E}^N_{k\ell} = \frac{1}{2} \sum_{i=1}^{n} \Big(\nabla N_i \mathbf{e}_\ell \mathbf{F}^N_{mk} + \nabla N_i \mathbf{e}_k \mathbf{F}^N_{m\ell} \Big) (\boldsymbol{\eta}_i)_m.$$

The components of $\mathbf{F}^N$ are computed as follows. In each Newton step, see Core Concept 3.1, the deformation $\boldsymbol{\varphi}_k^N$ is a sum of displacements $\mathbf{u}_k^N$ and the initial deformation $\boldsymbol{\varphi}_0^N$. We therefore write

$$\boldsymbol{\varphi}_k^N(\mathbf{x}) = \boldsymbol{\varphi}_0^N(\mathbf{x}) + \sum_{j=1}^{k-1} \mathbf{u}_j^N(\mathbf{x}) = \boldsymbol{\varphi}_0^N(\mathbf{x}) + \sum_{i=1}^{n} N_i(\mathbf{x}) \mathbf{x}_i,$$

i.e., all displacements are in V^N, so their finite sum is in V^N and is expressed by a linear combination of the basis. We compute the components of $\mathbf{F}^N$ for the simple case $\boldsymbol{\varphi}_0^N(\mathbf{x}) = \mathbf{0}$. The general case additionally requires the derivative of $\boldsymbol{\varphi}_0^N$.

In case that $\boldsymbol{\varphi}_0^N(\mathbf{x}) = \mathbf{0}$, we conclude

$$\mathbf{F}^N = \nabla \boldsymbol{\varphi}_k^N = \sum_{i=1}^{n} \mathbf{x}_i \otimes \nabla N_i \tag{4.7}$$

with components

$$\mathbf{F}^N_{mk} = \mathbf{e}_m^{\mathrm{T}}\Big(\sum_{i=1}^{n} \mathbf{x}_i \otimes \nabla N_i\Big)\mathbf{e}_k = \sum_{i=1}^{n} (\mathbf{e}_k \cdot \nabla N_i)(\mathbf{e}_m \cdot \mathbf{x}_i).$$

This explicit formula for the deformation gradient complements the discrete version of $\delta\mathbf{E}$. Since the variation of the Green-Lagrange tensor is a symmetric tensor, it has six independent components. In an implementation, storing the upper triangular part of $\delta\mathbf{E}$ is sufficient. For this purpose, the components are rewritten into a vector of length 6.

Definition 4.2 (Voigt notation)**.** Let $\mathbf{E}$ be the Green-Lagrange tensor or its variation,

$$\mathbf{E} = \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{pmatrix} \in \mathbb{R}^{3\times 3}_{symm}, \text{ and } V(\mathbf{E}) = \begin{pmatrix} E_{11} \\ E_{22} \\ E_{33} \\ 2E_{12} \\ 2E_{23} \\ 2E_{13} \end{pmatrix}.$$

The vector $V(\mathbf{E})$ is called $\mathbf{E}$ *in Voigt notation.*

Let $\mathbf{S}$ be the second Piola-Kirchhoff tensor,

$$\mathbf{S} = \begin{pmatrix} S_{11} & S_{12} & S_{13} \\ S_{12} & S_{22} & S_{23} \\ S_{13} & S_{23} & S_{33} \end{pmatrix} \in \mathbb{R}^{3\times 3}_{symm}, \text{ and } V(\mathbf{S}) = \begin{pmatrix} S_{11} \\ S_{22} \\ S_{33} \\ S_{12} \\ S_{23} \\ S_{13} \end{pmatrix}.$$

The vector $V(\mathbf{S})$ is called $\mathbf{S}$ *in Voigt notation.*

The reason for the different definition is that

$$\mathbf{E} : \mathbf{S} = V(\mathbf{E}) \cdot V(\mathbf{S}).$$

The dependence of $\delta\mathbf{E}$ on the test function degrees of freedom is

$$V(\delta\mathbf{E}) = \sum_{i=1}^{n} \mathbf{B}_i \boldsymbol{\eta}_i, \quad \mathbf{B}_i \in \mathbb{R}^{6\times 3}$$

with $\mathbf{B}_i =$

$$\begin{pmatrix} F_{11}\nabla N_i \cdot \mathbf{e}_1 & F_{21}\nabla N_i \cdot \mathbf{e}_1 & F_{31}\nabla N_i \cdot \mathbf{e}_1 \\ F_{12}\nabla N_i \cdot \mathbf{e}_2 & F_{22}\nabla N_i \cdot \mathbf{e}_2 & F_{32}\nabla N_i \cdot \mathbf{e}_2 \\ F_{13}\nabla N_i \cdot \mathbf{e}_3 & F_{23}\nabla N_i \cdot \mathbf{e}_3 & F_{33}\nabla N_i \cdot \mathbf{e}_3 \\ F_{11}\nabla N_i \cdot \mathbf{e}_2 + F_{12}\nabla N_i \cdot \mathbf{e}_1 & F_{21}\nabla N_i \cdot \mathbf{e}_2 + F_{22}\nabla N_i \cdot \mathbf{e}_1 & F_{31}\nabla N_i \cdot \mathbf{e}_2 + F_{32}\nabla N_i \cdot \mathbf{e}_1 \\ F_{12}\nabla N_i \cdot \mathbf{e}_3 + F_{12}\nabla N_i \cdot \mathbf{e}_2 & F_{22}\nabla N_i \cdot \mathbf{e}_3 + F_{23}\nabla N_i \cdot \mathbf{e}_2 & F_{32}\nabla N_i \cdot \mathbf{e}_3 + F_{33}\nabla N_i \cdot \mathbf{e}_2 \\ F_{11}\nabla N_i \cdot \mathbf{e}_3 + F_{13}\nabla N_i \cdot \mathbf{e}_1 & F_{21}\nabla N_i \cdot \mathbf{e}_3 + F_{23}\nabla N_i \cdot \mathbf{e}_1 & F_{31}\nabla N_i \cdot \mathbf{e}_3 + F_{33}\nabla N_i \cdot \mathbf{e}_1 \end{pmatrix},$$

see [34, p. 124]. Computing the contribution of each element to the integral equation, using Voigt notation for $\delta\mathbf{E}$ and $\mathbf{S}$ leads to

$$\int_e \delta\mathbf{E} : \mathbf{S} \, \mathrm{d}\mathbf{x} = \int_e V(\delta\mathbf{E}) \cdot V(\mathbf{S}) \, \mathrm{d}\mathbf{x}$$
$$= \int_e \sum_{i \in N_e} (\mathbf{B}_i \boldsymbol{\eta}_i) \cdot V(\mathbf{S}) \, \mathrm{d}\mathbf{x} = \sum_{i \in N_e} \boldsymbol{\eta}_i^{\mathrm{T}} \int_e \mathbf{B}_i^{\mathrm{T}} \cdot V(\mathbf{S}) \, \mathrm{d}\mathbf{x}$$

which defines the internal work of each degree of freedom on element level,

$$\mathbf{E}_{int}^i = \int_e \mathbf{B}_i^T V(\mathbf{S}) \, \mathrm{d}\mathbf{x}.$$

The integral is evaluated by numerical quadrature on the reference element. The global equation is

$$\int_\Omega \delta\mathbf{E} : \mathbf{S} \, \mathrm{d}\mathbf{x} = \bigcup_{e \in \mathcal{M}} \sum_{i \in N_e} \boldsymbol{\eta}_i^{\mathrm{T}} \mathbf{E}_{int}^i =: \boldsymbol{\eta}^{\mathrm{T}} \mathbf{E}_{int} \tag{4.8}$$

which defines two vectors $\boldsymbol{\eta}, \mathbf{E}_{int}$ of length $3n$. The operator $\cup$ stands for the finite element assembly operation. The assembly is performed like in linear FEM and is for example described in [31, pp. 62]. The vector $\boldsymbol{\eta}$ holds all the coefficients of the test functions. The vector $\mathbf{E}_{int}$ is called **internal load vector**. It depends on the displacement $\mathbf{u}$.

4.1.2 Discretization of the external virtual work

In the next step, we compute the term $\ell(\boldsymbol{\eta}^N)$ by inserting the linear combination, Equation (4.5), into the load term of the variational problem, see Equation (3.10). On each element, we have

$$\ell(\boldsymbol{\eta}^N) = \sum_{i \in N_e} \Big(\int_e (N_i(\mathbf{x}) \boldsymbol{\eta}_i)^{\mathrm{T}} \mathbf{f}(\mathbf{x}) \, \mathrm{d}\mathbf{x} + \int_{\Gamma_r} (N_i(\mathbf{x}) \boldsymbol{\eta}_i)^{\mathrm{T}} \mathbf{g}(\mathbf{x}) \, \mathrm{d}a \Big)$$

where Γ_r is the part of the surface of element e subjected to surface loading. The factors in the linear combination of $\boldsymbol{\eta}^N$ are independent of $\mathbf{x}$ so that we can write

$$\ell(\boldsymbol{\eta}^N) = \sum_{i \in N_e} \boldsymbol{\eta}_i^{\mathrm{T}} \underbrace{\int_e N_i(\mathbf{x}) \mathbf{f}(\mathbf{x}) \, \mathrm{d}\mathbf{x}}_{=: \mathbf{E}_{ext}^i} + \sum_{i \in N_e} \boldsymbol{\eta}_i^{\mathrm{T}} \underbrace{\int_{\Gamma_r} N_i(\mathbf{x}) \mathbf{g}(\mathbf{x}) \, \mathrm{d}a}_{=: \mathbf{E}_{ext}^{i,\Gamma}} .$$

Assembly with respect to the global ordering of the degrees of freedom gives

$$\ell(\boldsymbol{\eta}^N) = \bigcup_{e \in \mathcal{M}} \sum_{i \in N_e} \boldsymbol{\eta}_i^{\mathrm{T}} (\mathbf{E}_{ext}^i + \mathbf{E}_{ext}^{i,\Gamma}) =: \boldsymbol{\eta}^{\mathrm{T}} \mathbf{E}_{ext}. \tag{4.9}$$

The vector $\mathbf{E}_{ext}$ is called **external load vector** and depends only on $\mathbf{f}$ and $\mathbf{g}$.

4.1.3 Resulting system of nonlinear equations

Summarizing the two previous sections, we computed the internal load vector $\mathbf{E}_{int}$ and the external load vector $\mathbf{E}_{ext}$. Their scalar product with the global vector of coefficients in the linear combination of test functions gives the internal and external energy terms:

$$\boldsymbol{\eta}^{\mathrm{T}}\mathbf{E}_{int} = a(\boldsymbol{\varphi}^N, \boldsymbol{\eta}^N),$$
$$\boldsymbol{\eta}^{\mathrm{T}}\mathbf{E}_{ext} = \ell(\boldsymbol{\eta}^N).$$

Recalling Equation (3.11), we see that for the body in equilibrium, there must hold

$$\boldsymbol{\eta}^{\mathrm{T}}(\mathbf{E}_{int} - \mathbf{E}_{ext}) = 0 \quad \forall \boldsymbol{\eta} \in V_0.$$

We therefore require

$$\mathbf{E}_{int}(\mathbf{u}) - \mathbf{E}_{ext} = \mathbf{0},$$

which is a system of nonlinear equations. The dependence on $\mathbf{u}$ is stated explicitly since only the internal load vector has this dependence. This concludes the discretization of the nonlinear problem. In order to iterate towards the solution of this system of equations, in the next section we compute the discretization of the directional derivative to be used in the discrete Newton step.

Core Concept 4.1: System of nonlinear equations

Writing the test function and displacement as linear combinations

$$\boldsymbol{\eta}^N(\mathbf{x}) = \sum_{i=1}^{n} N_i(\mathbf{x})\boldsymbol{\eta}_i, \quad \mathbf{u}^N(\mathbf{x}) = \sum_{i=1}^{n} N_i(\mathbf{x})\mathbf{u}_i,$$

with respect to basis functions in Equation (4.2) and $\boldsymbol{\eta}_i, \mathbf{u}_i \in \mathbb{R}^3$ leads to discretized versions of the first derivatives $\nabla\mathbf{u}^N(\mathbf{x})$ (Equation (4.4)) and $\nabla\boldsymbol{\eta}^N(\mathbf{x})$, of the deformation tensor, Equation (4.7), and the variation of the Green-Lagrange tensor, Equation (4.6).
For the discrete internal virtual work, in Section 4.1.1 we derived a representation as a scalar product of two vectors of length $3n$:

$$a(\boldsymbol{\varphi}^N, \mathbf{u}^N) = \boldsymbol{\eta}^T\mathbf{E}_{int},$$

see Equation (4.8), and likewise for the discrete external energy in Section 4.1.2:

$$\ell(\mathbf{u}^N) = \boldsymbol{\eta}^T\mathbf{E}_{ext},$$

see Equation (4.9). We therefore concluded in Section 4.1.3 that our Newton update $\mathbf{u}$ is a solution of the system of nonlinear equations

$$\mathbf{E}_{int}(\mathbf{u}) - \mathbf{E}_{ext} = 0.$$

4.2 Galerkin discretization of the directional derivative

In order to find a solution to the nonlinear system of equations we just derived, we want to apply Newton's method to iteratively approach a root. For this purpose, we compute the directional derivative of $\mathbf{E}_{int}(\mathbf{u}) - \mathbf{E}_{ext} =: \mathbf{R}(\mathbf{u})$. Since the ansatz functions yield a continuous displacement, we can equivalently discretize the directional derivative of $a(\boldsymbol{\varphi}, \boldsymbol{\eta})$, which has already been computed in Section 3.3:

$$\mathrm{D}a(\boldsymbol{\varphi}, \boldsymbol{\eta})[\mathbf{u}] = \int_\Omega \underbrace{\nabla \mathbf{u}\mathbf{S} : \nabla \boldsymbol{\eta}}_{\text{term 1}} + \underbrace{\mathcal{C}\mathrm{D}\mathbf{E}[\mathbf{u}] : \mathrm{D}\mathbf{E}[\boldsymbol{\eta}]}_{\text{term 2}} \, \mathrm{d}\mathbf{x}. \tag{4.10}$$

The following paragraph is based on [34, pp. 128]. We compute the contribution of each element to the equation. Recall from Equation (4.4) that

$$\nabla \mathbf{u}^N = \sum_{k=1}^{n} \mathbf{u}_k \otimes \nabla N_k, \quad \nabla \boldsymbol{\eta}^N = \sum_{i=1}^{n} \boldsymbol{\eta}_i \otimes \nabla N_i.$$

Inserting these into the first term of Equation (4.10) gives

$$\left(\sum_{k \in N_e} \mathbf{u}_k \otimes \nabla N_k \right) \mathbf{S} : \left(\sum_{i \in N_e} \boldsymbol{\eta}_i \otimes \nabla N_i \right) = \left(\sum_{k \in N_e} \mathbf{u}_k \otimes \mathbf{S} \nabla N_k \right) : \left(\sum_{i \in N_e} \boldsymbol{\eta}_i \otimes \nabla N_i \right)$$

because of the rules for tensor contraction and the symmetry of $\mathbf{S}$. Linearity of the Frobenius scalar product yields

$$\sum_{k \in N_e} \sum_{i \in N_e} \left((\mathbf{u}_k \otimes \mathbf{S} \nabla N_k) : (\boldsymbol{\eta}_i \otimes \nabla N_i) \right) = \sum_{k \in N_e} \sum_{i \in N_e} \left((\mathbf{u}_k \cdot \boldsymbol{\eta}_i) \left(\mathbf{S} \nabla N_k \cdot \nabla N_i \right) \right),$$

see Equation (2.18). Finally, we obtain

$$\sum_{k \in N_e} \sum_{i \in N_e} \left(\mathbf{u}_k^{\mathrm{T}} \boldsymbol{\eta}_i \nabla N_k^{\mathrm{T}} \mathbf{S} \nabla N_i \right)$$

for the first term. Applying Voigt notation to the second term in Equation (4.10) gives

$$V(\delta \mathbf{E}) \cdot V(\mathcal{C} D\mathbf{E}[\mathbf{u}]).$$

The material model of the body characterizes the connection between the strain tensor (e.g., the right Cauchy-Green tensor $\mathbf{C}$) and the stress tensor $\mathbf{P}$. They are introduced in the following Chapter 5. We will limit our analysis to hyperelastic material models there. In this case, the tensor $\mathcal{C}$ has only 12 non-zero components, see [34, p. 79], therefore we can introduce a matrix $\mathbf{D} \in \mathbb{R}^{6 \times 6}$ such that

$$V(\mathcal{C} D\mathbf{E}[\mathbf{u}]) = \mathbf{D} V(D\mathbf{E}[\mathbf{u}]).$$

Substituting the discrete versions of $\delta\mathbf{E}$ and $D\mathbf{E}[\mathbf{u}]$ gives

$$\sum_{i\in N_e} \mathbf{B}_i\boldsymbol{\eta}_i \cdot \mathbf{D}\left(\sum_{k\in N_e} \mathbf{B}_k\mathbf{u}_k\right) = \sum_{i\in N_e}\sum_{k\in N_e} \boldsymbol{\eta}_i^{\mathrm{T}}\mathbf{B}_i^{\mathrm{T}}\mathbf{D}\mathbf{B}_k\mathbf{u}_k.$$

Summarizing, we have

$$\begin{aligned}
\mathrm{D}a(\boldsymbol{\varphi},\boldsymbol{\eta})[\mathbf{u}] &= \bigcup_{e\in\mathcal{M}} \int_e \left(\sum_{k\in N_e}\sum_{i\in N_e} \left(\mathbf{u}_k^{\mathrm{T}}\boldsymbol{\eta}_i\nabla N_k^{\mathrm{T}}\mathbf{S}\nabla N_i\right) + \boldsymbol{\eta}_i^{\mathrm{T}}\mathbf{B}_i^{\mathrm{T}}\mathbf{D}\mathbf{B}_k\mathbf{u}_k\right) \\
&= \bigcup_{e\in\mathcal{M}} \sum_{k\in N_e}\sum_{i\in N_e} \int_e \left(\mathbf{u}_k^{\mathrm{T}}\boldsymbol{\eta}_i\nabla N_k^{\mathrm{T}}\mathbf{S}\nabla N_i + \boldsymbol{\eta}_i^{\mathrm{T}}\mathbf{B}_i^{\mathrm{T}}\mathbf{D}\mathbf{B}_k\mathbf{u}_k\right) \\
&= \bigcup_{e\in\mathcal{M}} \sum_{k\in N_e}\sum_{i\in N_e} \int_e \boldsymbol{\eta}_i^{\mathrm{T}} \underbrace{\left(\nabla N_k^{\mathrm{T}}\mathbf{S}\nabla N_i + \mathbf{B}_i^{\mathrm{T}}\mathbf{D}\mathbf{B}_k\right)}_{=:\mathbf{K}_{ik}} \mathbf{u}_k.
\end{aligned}$$

Again, we assemble with respect to the global order of the degrees of freedom and obtain

$$\mathrm{D}a(\boldsymbol{\varphi},\boldsymbol{\eta})[\mathbf{u}] =: \boldsymbol{\eta}^T\mathbf{K}\mathbf{u} \tag{4.11}$$

with the **global stiffness matrix K** and the global vector of degrees of freedom **u**.

4.3 Shape functions and the hierarchical basis

In the previous section, the choice of the basis functions in Equation (4.1) was left unspecified, we only assumed that a mesh is available on our domain $\overline{\Omega}$ and that each basis function is non-zero on only one element and its neighbors. The properties of the global stiffness matrix, see Equation (4.11), in particular

- the sparsity pattern (see Section 6.4) and
- the condition number,

depend on the choice of basis functions, which is also essential for the modification of the nonlinear solver introduced in Chapter 7. Therefore, this section includes an in-depth discussion and visualization of the hierarchical basis functions.

In the following, we will consider simple domains in $\mathbb{R}$ and $\mathbb{R}^3$. Later, we will transform every element of the mesh to this simple domain ("reference element") using mappings, see Section 4.4. Since the deformation should be continuous across element boundaries, we have a special interest in the values of our basis function on the boundary of our simple domain. The basis functions should be constructed with the implementation in mind. This means that the continuity conditions should be

easy to impose, the basis functions should be easy to evaluate at different points in space and, as stated before, they should have local support. We start this section by switching to the case of Ω being an open interval in $\mathbb{R}$. We will construct piecewise polynomials that meet the requirements stated above. Then, we expand the definition of those piecewise polynomials from intervals to hexahedra.

Definition 4.3 (Legendre polynomial)**.** For $k \in \mathbb{N}$, the polynomial

$$P_k(x) = \begin{cases} 1 & \text{if } k = 0, \\ x & \text{if } k = 1, \\ \displaystyle\sum_{i=0}^{\lfloor k/2 \rfloor} (-1)^i \frac{(2k-2i)!}{(k-i)!\ (k-2i)!\ i!\ 2^k} x^{k-2i} & \text{else,} \end{cases}$$

is called *Legendre polynomial* of degree k.

The Legendre polynomials are pairwise orthogonal with respect to the scalar product

$$\langle P_k, P_\ell \rangle = \int_{-1}^{1} P_k(x) P_\ell(x)\ \mathrm{d}x,$$

and satisfy

$$(2n+1)P_n(x) = P'_{n+1}(x) - P'_{n-1}(x) \quad \text{for } n = 1, 2, \ldots \tag{4.12}$$

see [30, p. 317-318]. There holds

$$P_k(1) = 1,$$
$$P_k(-1) = (-1)^k.$$

The first few Legendre polynomials are visualized in Figure 4.1. Since $P_k(-1) \neq 0$ and $P_k(1) \neq 0$, a continuity condition on the boundary needs to take all basis functions into consideration. On a large scale, this is computationally expensive. We therefore move on to another set of polynomials which achieve the desired property of easy continuity enforcement by combining the Legendre polynomials such that most of them are zero when evaluated on the boundary.

Definition 4.4 (Hierarchical shape function)**.** For $k \in \mathbb{N}$, the polynomial N_k defined as

$$N_k(x) = \begin{cases} \frac{1-x}{2}, & \text{if } k = 0, \\ \frac{1+x}{2}, & \text{if } k = 1, \\ \frac{1}{\sqrt{4k-2}}(P_k(x) - P_{k-2}(x)) & \text{else,} \end{cases}$$

with the Legendre polynomials P_k, P_{k-2} of degree k and $k-2$ is called *hierarchical shape function*. The polynomial N_k has degree $\deg(N_k) = \max\{1, k\}$.

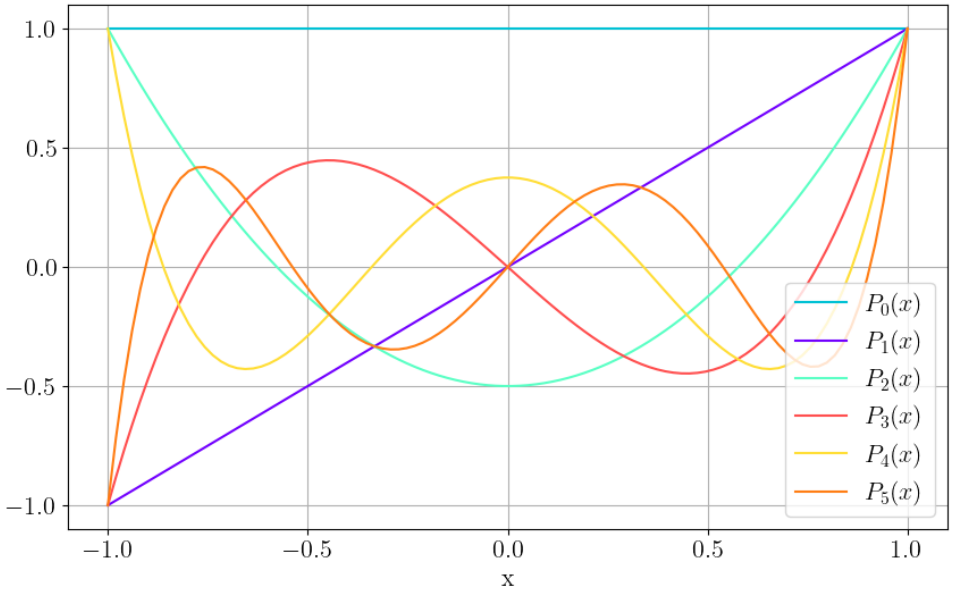


Figure 4.1: Legendre polynomials.

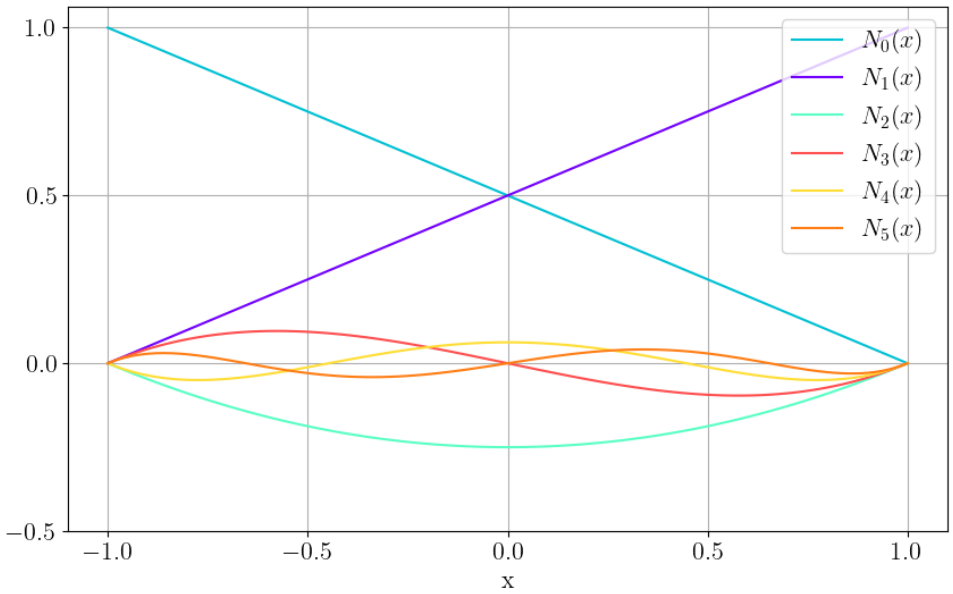


Figure 4.2: Hierarchical shape functions.

The first few hierarchical shape functions are visualized in Figure 4.2.

Lemma 4.5. *Let $k \in \mathbb{N}, k > 1$ and N_k be a hierarchical shape function. Then*

$$N_k(-1) = N_k(1) = 0.$$

Proof. Follows directly from $P_k(1) = 1$ and $P_k(-1) = (-1)^k$. □

Lemma 4.6. *Let $k, \ell \in \mathbb{N}, k, \ell > 1$ and N_k and N_ℓ be hierarchical shape functions. Then if $k \neq \ell$, it holds that*

$$\langle N_k, N_\ell \rangle := \int_{-1}^{1} N_k'(x) N_\ell'(x) \, \mathrm{d}x = 0. \tag{4.13}$$

Proof. For $k, \ell > 1, k \neq \ell$,

$$\begin{aligned}\int_{-1}^{1} N_k'(x) N_\ell'(x) \, \mathrm{d}x &= \frac{1}{\sqrt{4k-2}} \frac{1}{\sqrt{4\ell-2}} \int_{-1}^{1} \left(P_k(x) - P_{k-2}(x)\right)' \left(P_\ell(x) - P_{\ell-2}(x)\right)' \, \mathrm{d}x \\ &\overset{(4.12)}{=} \frac{(2k-1)}{\sqrt{4k-2}} \frac{(2\ell-1)}{\sqrt{4\ell-2}} \int_{-1}^{1} P_{k-1}(x) P_{\ell-1}(x) \, \mathrm{d}x = 0\end{aligned}$$

because the Legendre polynomials are pairwise orthogonal. □

Remark 4.7. Let $k \in \mathbb{N}$ and $\mathcal{B}_k = (N_2, \ldots, N_k)$. Then, $\langle \cdot, \cdot \rangle$ in Equation (4.13) is a scalar product on $\operatorname{span}(\mathcal{B}_k)$, and $\mathcal{B}_k$ is an orthogonal basis with respect to it.

We now consider a simple three-dimensional domain $\Omega_{st} = (-1, 1)^3$ with nodes, edges and faces denoted as in Figure 4.3. In the following section, we will see that the products of the hierarchical shape functions yield candidates for a basis of V_0 that makes the enforcement of continuity conditions easy, even in the three-dimensional case.

Definition 4.8 (Shape function product, maximum and total degree of a polynomial)**.** For every $i, k, \ell \in \mathbb{N}$, the shape function product $N_{i,k,\ell} : \overline{\Omega_{st}} \to \mathbb{R}$ is defined by

$$N_{i,k,\ell}(\xi, \eta, \zeta) = N_i(\xi) N_k(\eta) N_\ell(\zeta), \tag{4.14}$$

where N_i, N_j, N_k are hierarchical shape functions (see Definition 4.4).

The *maximum degree* of the multivariate polynomial $N_{i,k,\ell}$ is

$$\deg_{max}(N_{i,k,\ell}) := \max\{\deg(N_i), \deg(N_k), \deg(N_\ell)\}.$$

The *total degree* of $N_{i,k,\ell}$ is

$$\deg(N_{i,k,\ell}) := \deg(N_i) + \deg(N_k) + \deg(N_\ell).$$

Node modes

Consider the shape function product

$$N_{1,1,1}(\xi,\eta,\zeta) = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta),$$

on $\overline{\Omega_{st}}$ visualized in Figure 4.4. Since this shape function product takes the value 1 only on node V_7 and is zero for all other nodes, it is called "node mode". In total, there are 8 node modes:

Shape function product	**Non-zero on node**
$N_{0,0,0}$	V_1
$N_{1,0,0}$	V_2
$N_{1,1,0}$	V_3
$N_{0,1,0}$	V_4
$N_{0,0,1}$	V_5
$N_{1,0,1}$	V_6
$N_{1,1,1}$	V_7
$N_{0,1,1}$	V_8

Also, note that the shape function products are non-zero on the edges adjacent to their node and zero on all other edges. In particular, restricting one of the node modes to an edge gives an affine-linear function, e.g.

$$N_{1,1,1}(\xi,\eta,\zeta)|_{\xi=1,\zeta=1} = \frac{1}{2}(1+\eta).$$

In linear FEM, the basis is constructed using the node modes. In the example above, we can see that the deformation is then limited to be affine-linear between nodes. In high-order FEM, shape function products $N_{i,k,\ell}$ with $i,k,\ell > 1$ are permitted. In the following section, we will see that they can be classified as edge, face, or internal modes, based on the values of i,k and ℓ.

Edge modes

The shape function products $N_{i,k,\ell}$ where exactly one of the three indices i,k and ℓ is greater than 1 are called **edge modes**. The edge modes

- vanish on all nodes of Ω_{st} and
- do not vanish on exactly one of the edges.

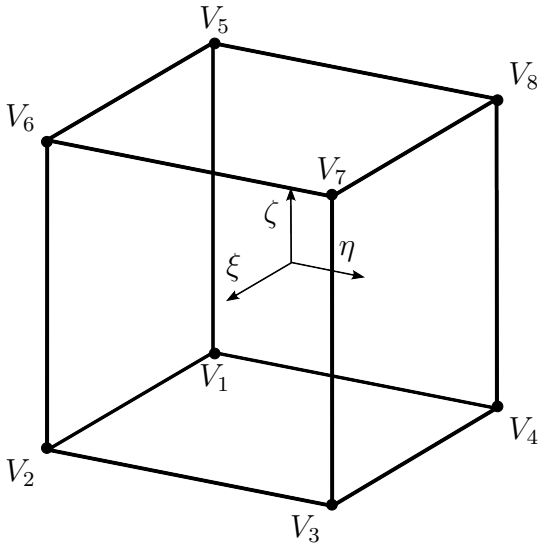


Figure 4.3: The standard or reference element $\Omega_{st} = (-1, 1)^3$.

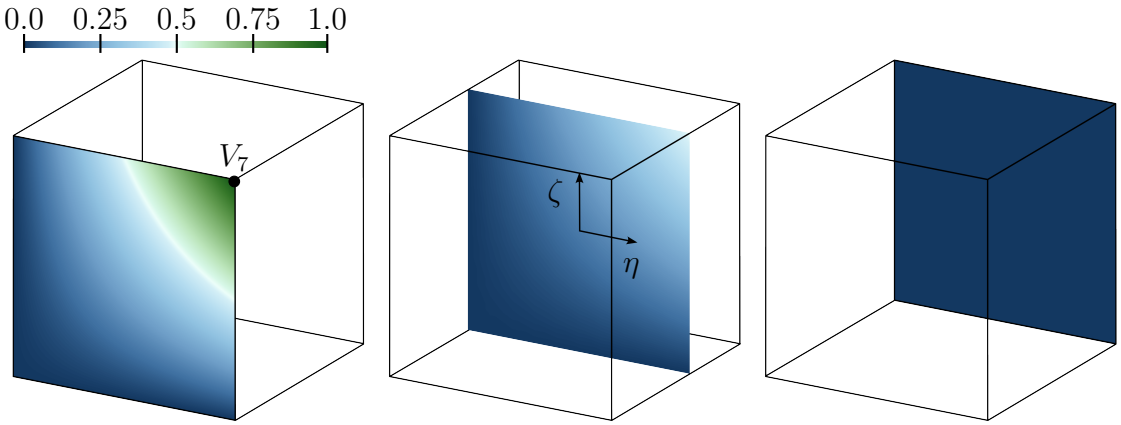


Figure 4.4: Three slices (left: $\xi = 1$, middle: $\xi = 0$, right: $\xi = -1$) of the shape function product $N_{1,1,1} : \Omega^{st} \to \mathbb{R}$. We observe that $N_{1,1,1}(1, 1, 1) = 1$. On all other nodes, $N_{1,1,1} = 0$ holds. Furthermore, on $\partial\Omega$, $N_{1,1,1}$ is non-zero only on the neighbor edges and faces of V_7.

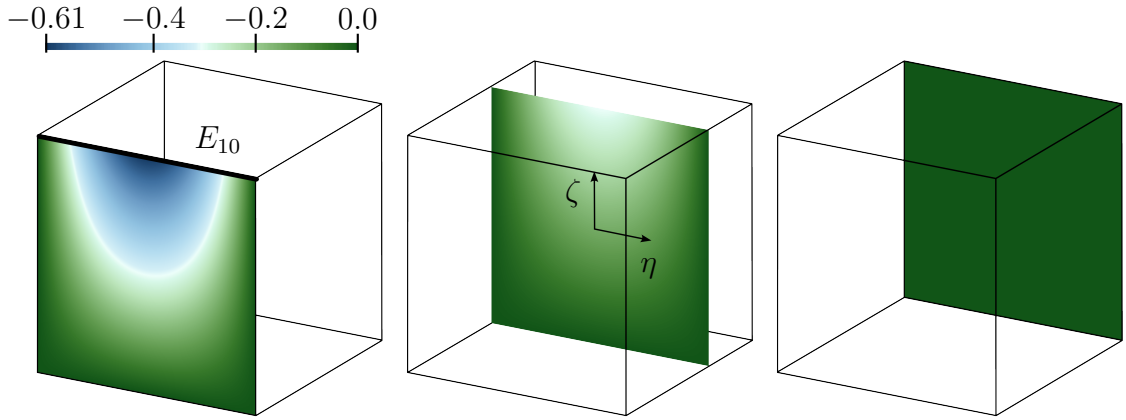


Figure 4.5: The shape function product $N_{1,2,1}$ is a quadratic polynomial on edge E_{10} and zero on all other edges.

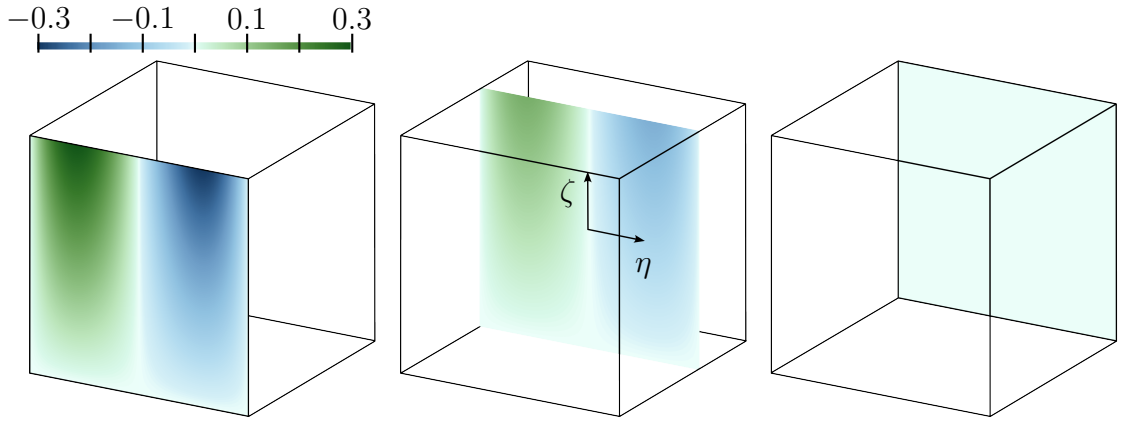


Figure 4.6: Shape function product $N_{1,3,1}$.

For example, the shape function product

$$N_{1,2,1}(\xi,\eta,\zeta) = \frac{1}{4}(1+\xi)N_2(\eta)(1+\zeta),$$

see Figure 4.5, vanishes on every edge except E_{10}. Restricted to E_{10}, it is a polynomial of degree 2:

$$N_{1,2,1}(\xi,\eta,\zeta)|_{\xi=1,\zeta=1} = N_2(\eta).$$

Replacing $N_2(\eta)$ by $N_k(\eta)$ for $k > 2$ gives shape function products with analogous properties, see Figures 4.6 and 4.7. With a maximal shape function degree $p \in \mathbb{N}$ (meaning $i, k, \ell \leq p$), the number of possible edge modes is $12(p-1)$.

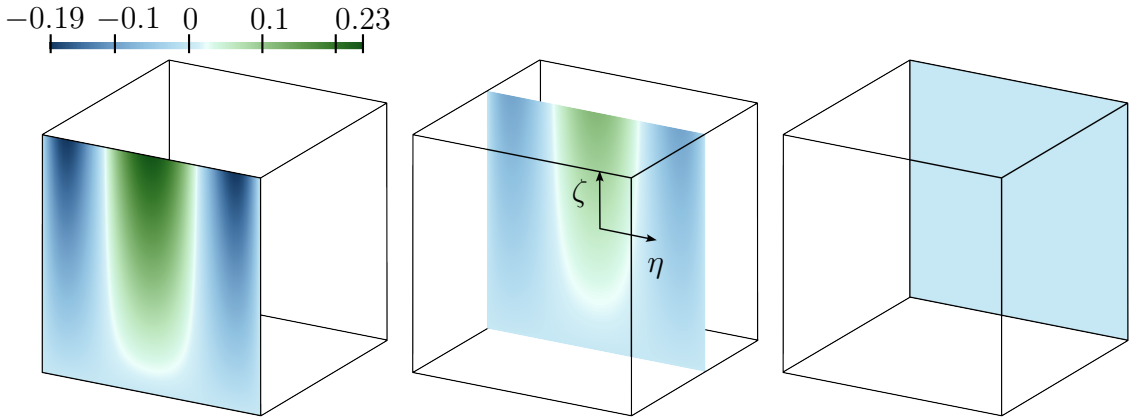


Figure 4.7: Shape function product $N_{1,4,1}(\xi, \eta, \zeta)$.

Face modes

The shape function products $N_{i,k,\ell}$ with exactly one of the indices smaller than 2 are called **face modes**. The face modes

- vanish on all nodes and edges of $\overline{\Omega_{st}}$ and
- are non-zero on exactly one of the faces.

For example,

$$N_{1,2,2}(\xi, \eta, \zeta) = \frac{1}{2}(1 + \xi)N_2(\eta)N_2(\zeta),$$

see Figure 4.8, is non-zero on F_3 and vanishes on all other faces. Assuming a maximal polynomial degree p, the number of face modes is $6(p-1)^2$.

Internal modes

The remaining shape function products are the **internal modes**

$$N_{i,k,\ell}(\xi, \eta, \zeta) \text{ with } 2 \leq i, k, \ell \leq p,$$

assuming a maximal polynomial degree p. There are $(p-1)^3$ internal modes. Because the factors N_i, N_j and N_ℓ vanish on the interval boundary, there holds

$$\operatorname{supp}(N_{i,k,\ell}) \cap \partial\Omega_{st} = \varnothing.$$

The internal modes vanish on all nodes, edges, and faces of $\overline{\Omega_{st}}$. Therefore, they are omitted from the continuity conditions.

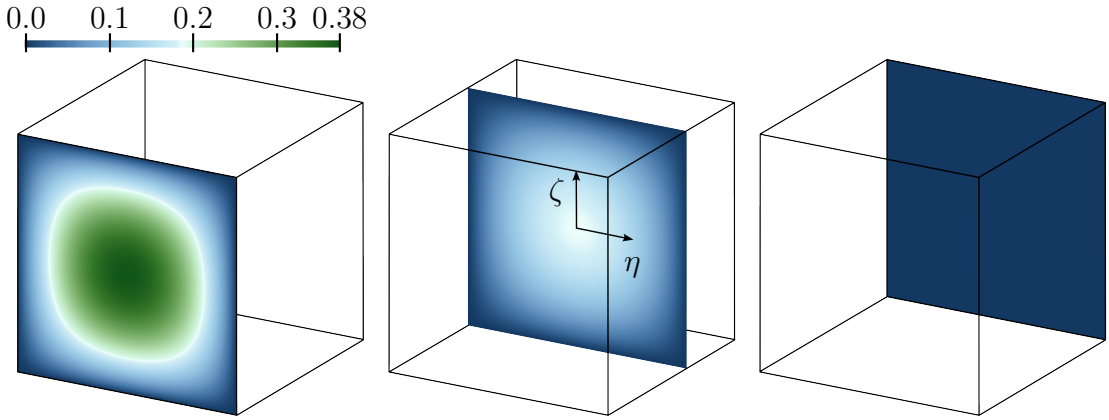
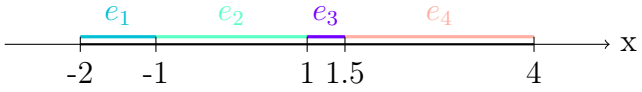


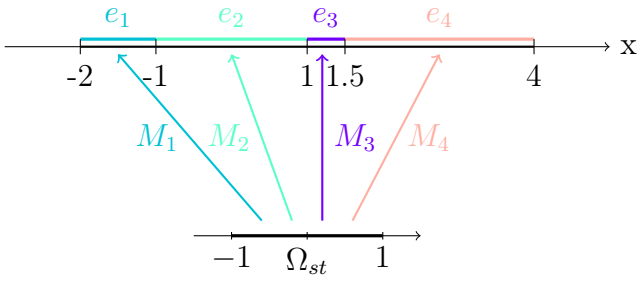
Figure 4.8: The shape function product $N_{1,2,2}(\xi, \eta, \zeta)$ is non-zero on the front face and vanishes on all other faces.

4.4 Mapping and selection of the basis

In the last section, we constructed shape functions on the simple domain Ω_{st}. In the next step, we use Ω_{st} as a reference element and map it to an arbitrary element of our mesh (also called domain element). Again, we start with an example of a one-dimensional domain. Consider the following mesh on the domain $\Omega = (-2, 4)$.



The reference element is $\Omega_{st} = (-1, 1)$. We consider affine-linear mappings M_i from the reference element to the domain elements:



We evaluate integrals over Ω by transforming to the reference element, e.g. since

$$M_1(\xi) = \frac{1}{2}\xi - \frac{3}{2}$$

by using Equation (A.2) from the Appendix, we get a factor of $\frac{1}{2}$ during change of variables,

$$\int_{e_1} u(x)\ \mathrm{d}x = \int_{-2}^{-1} u(x)\ \mathrm{d}x = \frac{1}{2}\int_{-1}^{1} u(M_1(\xi))\ \mathrm{d}\xi \approx \frac{1}{2}\sum_{i=1}^{k} w_i\ u(M_1(\xi_i)),$$

where w_i are the weights of the quadrature formula and ξ_i are the quadrature nodes in $[-1, 1]$. They can be tabulated on $\overline{\Omega_{st}}$ and reused for all integrals.

Analogously, for an element $e = (a, b)$, the affine-linear mapping is

$$M_e : \Omega_{st} \to e, \quad M_e(\xi) = \frac{b-a}{2}\xi + \frac{a+b}{2}$$

and there holds

$$\int_a^b u(x)\ \mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} u(M_e(\xi))\ \mathrm{d}\xi.$$

The affine-linear mappings M_e connect the reference element with a domain element e. They are used to define the basis functions on the domain Ω. Consider the hierarchical shape functions $N_k, 0 \le k \le 3$, see Definition 4.4 and Figure 4.2. For any shape function and element, consider

$$M_e \circ N_k : e \to \mathbb{R}$$

and extend them to the neighbor element with 0, see Figure 4.9. Our goal is to construct continuous basis functions, which rules out $M_{e_1} \circ N_0$ and $M_{e_2} \circ N_1$. Instead, we use their sum, see Figure 4.10. Then

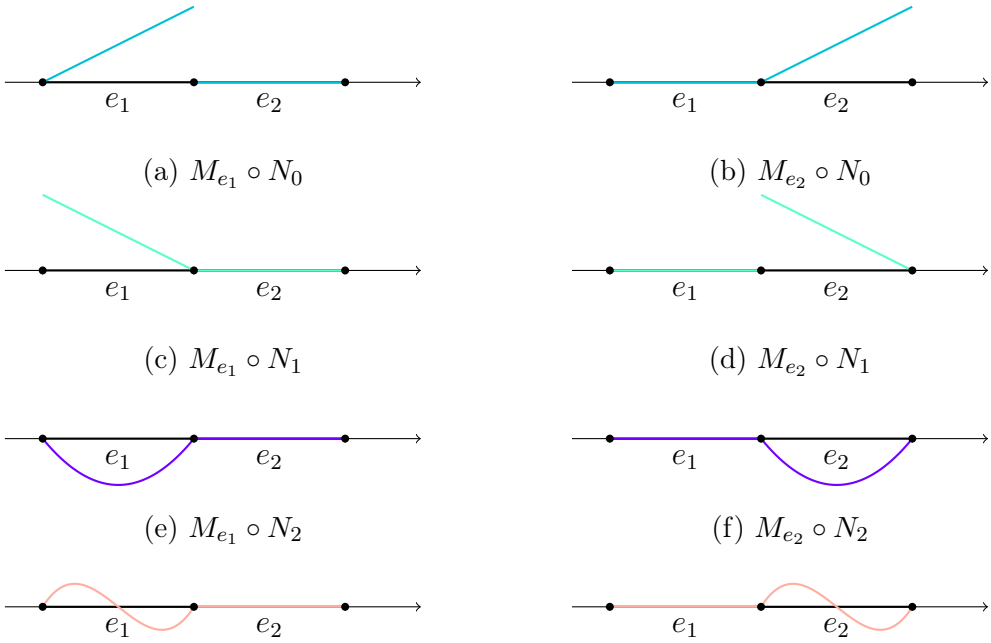
$$f \in \operatorname{span}\{\underbrace{M_{e_1} \circ N_1, M_{e_2} \circ N_0}_{\substack{\text{ramp functions}\\ \text{for boundary nodes}}}, \underbrace{M_{e_1} \circ N_0 + M_{e_2} \circ N_1}_{\substack{\text{hat function}\\ \text{for inner node}}},$$
$$M_{e_1} \circ N_2, M_{e_2} \circ N_2, M_{e_1} \circ N_3, M_{e_2} \circ N_3\}$$

has the following properties:

- The function f is continuous.
- On every element (e_1 or e_2), it is a polynomial of degree ≤ 3.

The generalization to a partition of an interval $(a, b) \subset \mathbb{R}$ with $n \in \mathbb{N}$ subintervals $e_i, 1 \le i \le n$ and polynomials of degree $p \in \mathbb{N}$ is straightforward. Assuming that the elements are numbered from left to right,

$$V := \operatorname{span}\Big(\underbrace{\{M_{e_1} \circ N_1, M_{e_n} \circ N_0\}}_{\substack{\text{ramp functions}\\ \text{for boundary nodes}}} \cup \underbrace{\{M_{e_i} \circ N_0 + M_{e_{i+1}} \circ N_1,\ 1 \le i \le n-1\}}_{\substack{\text{hat functions}\\ \text{for inner nodes}}}$$



(a) $M_{e_1} \circ N_0$ (b) $M_{e_2} \circ N_0$

(c) $M_{e_1} \circ N_1$ (d) $M_{e_2} \circ N_1$

(e) $M_{e_1} \circ N_2$ (f) $M_{e_2} \circ N_2$

(g) $M_{e_1} \circ N_3$ (h) $M_{e_2} \circ N_3$

Figure 4.9: Mapping the hierarchical shape functions N_0, N_1, N_2, N_3 to the domain elements e_1, e_2 and extending them with 0 to the whole domain yields two discontinuous and six continuous functions.

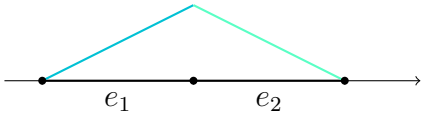


Figure 4.10: The point-wise sum of $M_{e_1} \circ N_0$ and $M_{e_2} \circ N_1$ is a continuous function. Due to the shape of its graph, it is sometimes called a hat function.

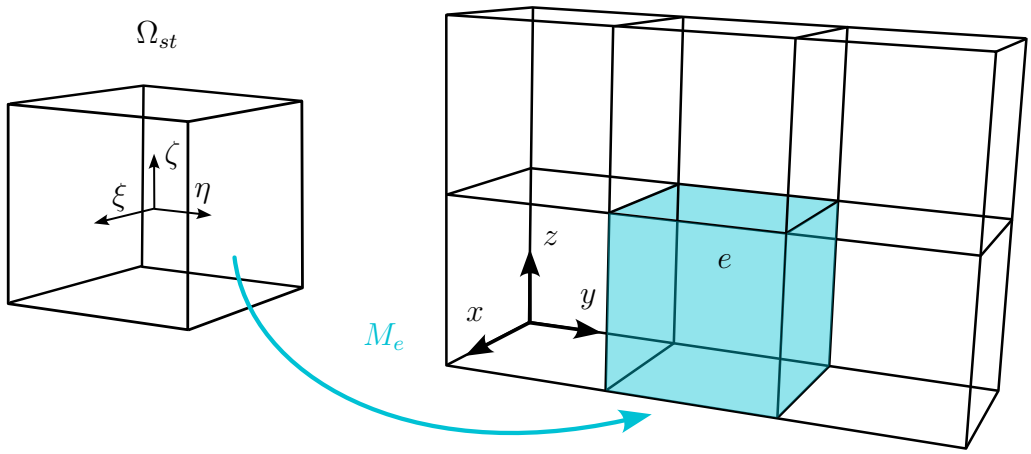


Figure 4.11: The domain $\Omega = (0,1) \times (0,3) \times (0,2)$ is discretized using a uniform mesh of six elements. Each domain element is the image of a bijective mapping $M_e : \Omega_{st} \to e$.

$$\cup \{M_{e_i} \circ N_k,\ 1 \leq i \leq n, 2 \leq k \leq p\}\Big)$$

yields a vector space of continuous functions that are polynomials of degree $\leq p$ when restricted to an element.

To summarize, mapping from the reference element to the domain elements has the following advantages.

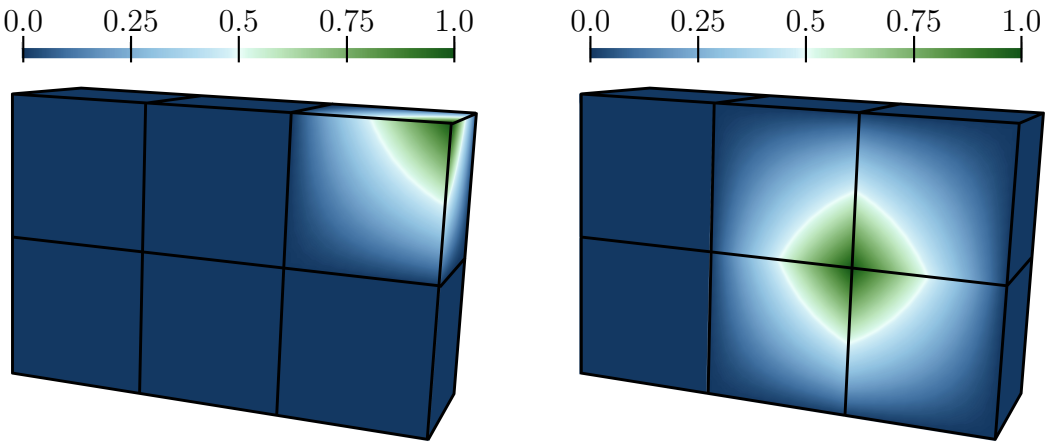
1. Integrals are pulled back to the reference element and evaluated there, using quadrature.

2. The basis functions for the entire mesh are assembled from the basis defined on the reference element, mapped to the mesh.

We generalize the concept to three-dimensional domains.

Remark 4.9. Combining a hexahedral mesh and linear mapping yields a computational domain consisting of hexahedra. For many problems, this is a poor representation, especially if the element size is large (compared to the size of the actual physical domain). When a precise representation of the geometry and large element sizes are required, blending techniques can be used. The basic idea is to describe parts of the boundary by a parametrized function, add them to the mapping function, and smooth them out in the direction of other edges and faces, such that the influence vanishes there, see [20].

Remark 4.10. As in the one-dimensional case, integrals over the domain element are evaluated by transforming back to the reference element (Equation A.2 in the Appendix) and using the tabulated weights and nodes for the quadrature formula.



(a) (b)

Figure 4.12: Node basis function for a single-element node of $\mathcal{M}$ (left) and a multi-element node (right).

Consider the reference element $\Omega_{st} = (-1, 1)^3$ and a mesh $\mathcal{M}$, see Definition 4.1. As each element $e \in \mathcal{M}$ is a hexahedron, the reference element can be mapped onto each domain element e by an affine-linear function M_e, see Figure 4.11. Analogous to the one-dimensional domain, the shape function products defined in the section before Equation (4.14) on the reference domain are mapped to domain elements. The candidates for basis functions on the domain Ω are

$$M_e \circ N_{i,k,\ell} : e \to \mathbb{R}.$$

Constructing a basis for a space of continuous piecewise polynomial functions in three dimensions presents a new challenge. The global displacement should be continuous not only in each node, but also across edges and faces. This is achieved by combining the shape function products of neighbor elements to shared basis functions for each node, edge, and face. For a one-dimensional domain, shared basis functions (hat functions) were introduced for the internal nodes. Consider the computational domain $\Omega = (0, 2) \times (0, 3) \times (0, 1)$, discretized with 6 finite elements, see Figure 4.11. In the following, we will visualize the respective basis functions that ensure the continuity of the numerical solution.

Shared node basis functions

For a node that intersects with the closure of exactly one element, the basis function is the respective shape function product mapped to the domain element and extended with 0 to all other elements, see Figure 4.12(a).

A node shared by at least two elements is assigned a basis function by adding the respective shape function products (mapped to the domain) for all neighbor

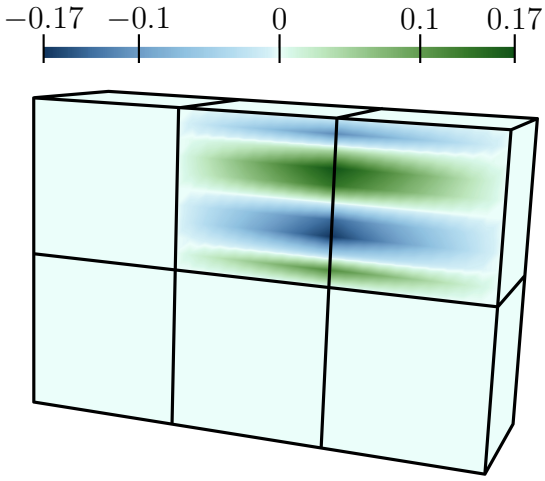


Figure 4.13: Shared edge basis function.

elements, see Figure 4.12(b).

The point-wise sum of the mapped shape function products yields a combined basis function which is non-zero on all neighbor elements of the node. In total, this approach yields 24 (shared) node basis functions, one for each node of the example mesh in Figure 4.11.

Shared edge basis functions

Next, we consider the edges of the mesh. Let $p \in \mathbb{N}$ be the maximal polynomial degree chosen for the basis functions. Analogous to the construction of the shared node basis functions, for each edge and each polynomial degree $j \leq p$, the respective edge modes of all neighbor elements are mapped to the domain. Their sum yields a combined basis function for the respective edge, see Figure 4.13. In the case of an odd j, special care is needed during the implementation to ensure the continuity of the combined edge basis function. For odd shape functions, $N_j(x) = -N_j(-x)$ holds for $x \in [-1, 1]$. If the local orientations of the mapping of two neighbouring elements do not match, the combined basis function will not be continuous (sometimes called "sign mismatch"). This issue can be easily corrected by adjusting the mapping.

For the mesh shown above which has 46 edges, there are $46(p-1)$ shared edge basis functions.

Shared face basis functions

For the 29 faces of the example mesh, there are $29(p-1)^2$ shared face basis functions. As for the edge basis functions, possible sign mismatches on shared faces need to be taken care of during implementation.

Internal basis functions

Mapping the internal modes from the reference element to each domain element gives us $6(p-1)^3$ internal basis functions. Since they vanish on the element boundary, they can be used as standalone basis functions.

In total, we retrieve $6(p-1)^3+29(p-1)^2+46(p-1)+24$ possible (shared) basis functions and each linear combination yields a continuous function $\mathbf{u}:\Omega\to\mathbb{R}$.

In a final step, we show which of the possible shared node, shared edge, shared face and internal basis functions are typically selected for the high-order discretization. This choice is consistent with [12].

Definition 4.11 (Product space)**.** Given a hexahedral mesh $\mathcal{M}$ (see Definition 4.1) on a domain Ω, the *product space* of order $p\in\mathbb{N}$ is the span of all (shared) node, edge and face basis functions with $\deg_{max}(N_{i,k,\ell})\leq p$.

Definition 4.12 (Trunk space)**.** Given a hexahedral mesh $\mathcal{M}$ on a domain Ω, the *trunk space* of order $p\in\mathbb{N}$ is the span of all (shared) node basis functions, (shared) edge basis functions of total degree $\deg(N_{i,k,\ell})\leq p+2$, (shared) face basis functions of total degree $\deg(N_{i,k,\ell})\leq p+1$ and internal basis functions of total degree $\deg(N_{i,k,\ell})\leq p$.

The permitted basis functions for the trunk space are given for $p=1,\dots 5$ in Table 4.1. Note that for $p=1$, only node modes are permitted. For $p=2$, the edge modes based on the shape function products $N_{i,k,\ell}$ of total degree $\deg(N_i)+\deg(N_k)+\deg(N_\ell)=4$ are added. Since the degree of each factor is at least 1, exactly one of the factors has degree 2. Therefore, for $p=2$, the edge modes that are quadratic in one component are added. Likewise, for $p=5$, the basis functions based on the shape function product with exactly two quadratic components are included.

Remark 4.13. In this thesis, we limit ourselves to isotropic high-order finite element spaces, meaning that basis functions are selected according to their degree, which is invariant under rotation in space. In the case of thin-walled structures, the deformation might behave differently depending on the direction in space, giving reason to use anisotropic high-order finite elements spaces.

Table 4.1: Permitted basis functions depending on the order p of the trunk space.

	node modes	**edge modes**	**face modes**	**internal modes**
	✓	total degree $\leq p+2$	total degree $\leq p+1$	total degree $\leq p$
$p = \mathbf{1}$	✓	×	×	×
$p = \mathbf{2}$	✓	≤ 4	×	×
$p = \mathbf{3}$	✓	≤ 5	×	×
$p = \mathbf{4}$	✓	≤ 6	≤ 5	×
$p = \mathbf{5}$	✓	≤ 7	≤ 6	×
$p = \mathbf{6}$	✓	≤ 8	≤ 7	≤ 6

Chapter 5

Nonlinear Elasticity

The primary variable of our simulation is the unknown deformation of a solid body under the influence of external forces and body forces. The available differential equation is given by Cauchy's Theorem 2.2 and is set up for the stress tensor $\mathbf{P}$. Cauchy's Theorem states the fundamental physical law that a closed system's external and internal forces must be in balance. During the analysis of the Newton step in Chapter 3, we introduced the concept of strain (see Definition 3.5). Deformation, displacement and strain are directly related, as shown in Figure 5.1. The missing connection of these three with the external forces and the stress is characterized with a material model.

Material models describe how we expect a given material to behave in the presence of strains. They are designed to model certain real-world behaviors of materials that are observed in experiments. In contrast to the balance equation, a material model does not express a law of nature — it is an assumption. Therefore, the simulation results are always to be interpreted with respect to the material model used. Here, we limit ourselves to elastic material models.

Elasticity

Elasticity is the property of a material to

- deform under the influence of external forces and
- return to its original state after the external force is removed.

This assumption makes sense for materials like rubber or organic tissue. A metallic material is usually modeled with an elastic material model for small stresses. As the external forces and therefore the stresses get larger, the deformation becomes more significant, and the second assumption (that the material will return to its initial state) is no longer valid. The critical stress, after which the deformation becomes irreversible, is called flow stress. Another limitation of elastic material

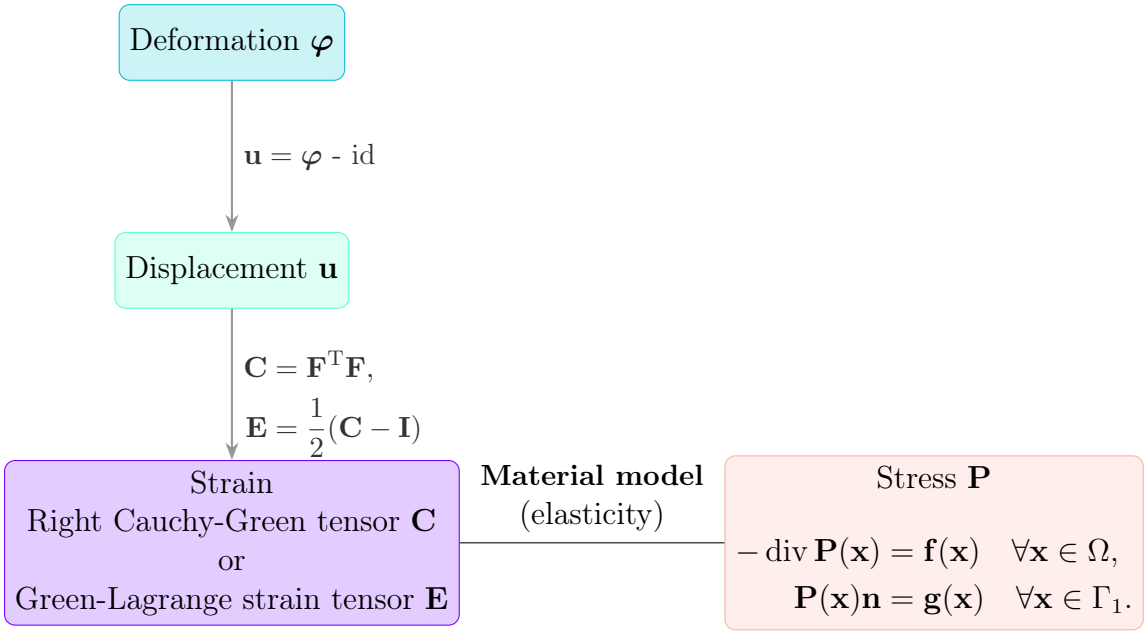


Figure 5.1: Connection of the primary variable, the deformation, to the partial differential equation from Cauchy's theorem. The deformation gradient is $\mathbf{F} = J_{\boldsymbol{\varphi}}$, see Chapter 2. The identity mapping is $\mathrm{id} : \Omega \to \Omega$, $\mathrm{id}(\mathbf{x}) = \mathbf{x}$ and the identity tensor is $\mathbf{I} \in \mathbb{R}^{3\times 3}$.

models is that they are not able to predict tearing. To simulate situations involving damage of the object under consideration, sophisticated methods such as phase field models are used, see for example [17].

The following sections will use the notation:

$$\begin{aligned}
\mathbb{M}^3 &:= \mathbb{R}^{3\times 3},\\
\mathbb{M}^3_+ &:= \{\mathbf{A} \in \mathbb{R}^{3\times 3} \mid \det(\mathbf{A}) > 0\},\\
\mathbb{M}^3_{\mathrm{symm}} &:= \{\mathbf{A} \in \mathbb{R}^{3\times 3} \mid \mathbf{A}^{\mathrm{T}} = \mathbf{A}\}\\
\mathbb{M}^3_{\mathrm{symm},+} &:= \{\mathbf{A} \in \mathbb{R}^{3\times 3} \mid \mathbf{A}^{\mathrm{T}} = \mathbf{A},\ \det(\mathbf{A}) > 0\}
\end{aligned}$$

Definition 5.1 (Invariant of a matrix)**.** For $\mathbf{A} \in \mathbb{M}^3$, a scalar-valued function $f : \mathbb{M}^3 \to \mathbb{R}$ is called an *invariant* of $\mathbf{A}$ if

$$f(\mathbf{Q}^{\mathrm{T}}\mathbf{A}\mathbf{Q}) = f(\mathbf{A})$$

for all orthogonal $\mathbf{Q} \in \mathbb{M}^3$.

A short computation shows that all elements in the tripel

$$\mathbf{i}_{\mathbf{A}} = (\operatorname{trace}(\mathbf{A}), \det(\mathbf{A})\operatorname{trace}(\mathbf{A}^{-1}), \det(\mathbf{A})) \tag{5.1}$$

are invariants of $\mathbf{A}$.

5.1 The response function of an elastic material

To formalize the concept of elasticity, we introduce the following definition of an elastic material, see [8, p. 91].

Definition 5.2 (Elastic material)**.** A material is called *elastic* if there exists a mapping

$$\widehat{\boldsymbol{\sigma}} : \bar{\Omega} \times \mathbb{M}^3_+ \to \mathbb{M}^3_{symm}$$

such that

$$\boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}) = \widehat{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{F}(\mathbf{x})) \quad \forall \mathbf{x} \in \Omega,$$

where $\mathbf{F}$ is the deformation gradient of $\boldsymbol{\varphi}$. The equation above is called a *constitutive equation* and $\widehat{\boldsymbol{\sigma}}$ is called the *response function* of the material.

The definition states that the Cauchy stress tensor at each point of the deformed configuration only depends on the respective point in the initial configuration and the deformation gradient. Note that all positive definite matrices from $\mathbb{R}^{3\times3}$ are permitted as deformation gradients. Therefore, internal constraints for the deformation gradient conflict with this definition. For example, since $\det(\mathbf{F})$ measures the change of volume during deformation, incompressible materials with $\det(\mathbf{F}) = 1$ are not elastic according to the definition above.

To see that this definition of an elastic material agrees with the colloquial definition before, we have to keep in mind which cases we want to exclude:

1. A material not deforming.
2. A material undergoing a deformation that is not reversed when removing the force.

Situation 1 is already ruled out by considering Cauchy's theorem and modeling with continuous objects. Situation 2 would mean that the stress field must depend on time (or past deformations), which is not allowed according to Definition 5.2.

The limitations of the definition are the following:

- The stress field is assumed to depend only on the first derivative of the deformation and
- the dependence is only local (i.e., there is no direct dependence on the value of $\mathbf{F}$ at other points $\mathbf{y} \in \Omega$).

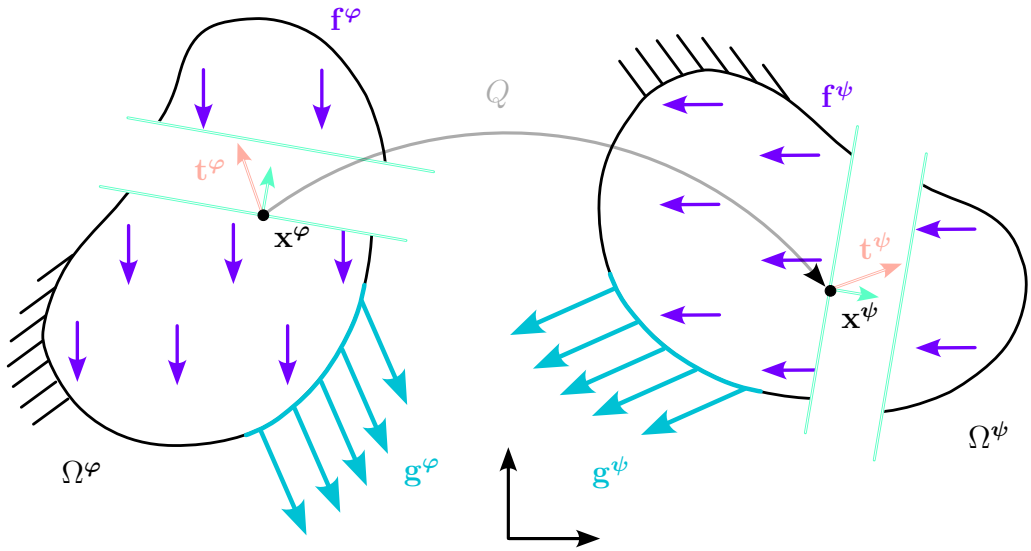


Figure 5.2: The deformed domain Ω^{φ} is rotated around the origin using the rotation matrix Q. The normal vector of the cut surface is rotated accordingly. Then, the Cauchy stress vector at $\mathbf{x}^{\psi} = Q\mathbf{x}^{\varphi}$ is the rotated stress vector at $\mathbf{x}^{\varphi}$.

Alternatively, the response function can be stated for the first and second Piola-Kirchhoff stress tensors since they are related to the Cauchy stress tensor by the Piola transform, see Equations (2.7) and (3.21). The response function for the first Piola-Kirchhoff stress tensor is

$$\widehat{\mathbf{P}}(\mathbf{x}, \mathbf{F}) = \det(\mathbf{F})\widehat{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{F})\mathbf{F}^{-\mathrm{T}}$$

and for the second Piola-Kirchhoff stress

$$\widehat{\mathbf{S}}(\mathbf{x}, \mathbf{F}) = \det(\mathbf{F})\mathbf{F}^{-1}\widehat{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{F})\mathbf{F}^{-\mathrm{T}}. \tag{5.2}$$

In practice, not all response functions properly represent an elastic object. For example, from a physical point of view, rotating Ω^{φ}, $\mathbf{f}^{\varphi}$ and $\mathbf{g}^{\varphi}$ around the origin should result in an analogously rotated stress vector, see Figure 5.2. In the following, we impose this property of stress states in continuous objects which will restrict the class of constitutive equations.

Axiom 5.3. *Consider the deformation $\boldsymbol{\varphi}$ on Ω and the surface load $\mathbf{g}^{\varphi}$ and body load $\mathbf{f}^{\varphi}$, resulting in the deformed configuration Ω^{φ}. Let $\mathbf{Q} \in \mathbb{R}^{3\times 3}$ be an orthogonal matrix with $\det(\mathbf{Q}) = 1$ and $\boldsymbol{\psi}$ be the rotated deformation*

$$\boldsymbol{\psi} := \mathbf{Q}\boldsymbol{\varphi}.$$

Then the stress vector $\mathbf{t}^{\psi}$ of the rotated configuration is the rotated stress vector of the original configuration:

$$\mathbf{t}^{\psi}(\mathbf{x}^{\psi}, \mathbf{Q}\mathbf{n}) = \mathbf{Q}\mathbf{t}^{\varphi}(\mathbf{x}^{\varphi}, \mathbf{n}) \quad \forall \mathbf{x} \in \bar{\Omega}, \mathbf{n} \in S^2.$$

The property is called ***(material) frame-indifference.***

The following theorem provides characterizations of frame-indifferent materials.

Theorem 5.4. *Let $\widehat{\boldsymbol{\sigma}}$ be a response function for a given material. Then, the following statements are equivalent to each other:*

1. *The material is frame-indifferent.*
2. *The response function satisfies $\widehat{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{QF}) = \mathbf{Q}\widehat{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{F})\mathbf{Q}^{\mathrm{T}}$ for all $\mathbf{F} \in \mathbb{M}^3_+$ and all orthogonal matrices $\mathbf{Q} \in \mathbb{R}^{3\times 3}$ with $\det(\mathbf{Q}) = 1$.*
3. *There exists a mapping $\tilde{\mathbf{S}} : \bar{\Omega} \times \mathbb{M}^3_{symm,+} \to \mathbb{M}^3_{symm}$ such that $\widehat{\mathbf{S}}(\mathbf{x}, \mathbf{F}) = \tilde{\mathbf{S}}(\mathbf{x}, \mathbf{F}^{\mathrm{T}}\mathbf{F})$ for all $\mathbf{F} \in \mathbb{M}^3_+$ where $\widehat{\mathbf{S}}$ is the response function for the second Piola-Kirchhoff stress tensor, see Equation* (5.2).

Proof. See [8, pp. 102]. □

The class of possible response functions can be restricted even further in the case of isotropic materials.

Definition 5.5 (Isotropic material)**.** If the response function of a material satisfies

$$\widehat{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{FQ}) = \widehat{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{F})$$

for all $\mathbf{F} \in \mathbb{M}^3_+, \mathbf{Q} \in \mathbb{M}^3_+$ with $\det(\mathbf{Q}) = 1$, we call the material isotropic.

The intuitive idea behind this definition is that the material behaves the same locally, regardless of direction. The equivalent definition for the second Piola-Kirchhoff stress tensor is

$$\widehat{\mathbf{S}}(\mathbf{x}, \mathbf{FQ}) = \mathbf{Q}^{\mathrm{T}}\widehat{\mathbf{S}}(\mathbf{x}, \mathbf{F})\mathbf{Q},$$

see [8, p. 106]. Taking the axiom of material frame-indifference into consideration, the response functions of isotropic materials have a special form:

Theorem 5.6 (Response function of an isotropic elastic material [8, pp. 115])**.** *For an elastic isotropic material and assuming that the response function is frame-indifferent, for an arbitrary deformation $\boldsymbol{\varphi} : \bar{\Omega} \to \mathbb{R}^3$, the Cauchy stress tensor at $\mathbf{x}^{\boldsymbol{\varphi}}$ is*

$$\boldsymbol{\sigma}^{\boldsymbol{\varphi}}(\mathbf{x}^{\boldsymbol{\varphi}}) = \widehat{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{F}) = \bar{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{F}\mathbf{F}^{\mathrm{T}})$$

where $\bar{\boldsymbol{\sigma}}$ takes the form

$$\bar{\boldsymbol{\sigma}}(\mathbf{x}, \mathbf{B}) = \beta_0(\mathbf{x}, \mathbf{i_B})\mathbf{I} + \beta_1(\mathbf{x}, \mathbf{i_B})\mathbf{B} + \beta_2(\mathbf{x}, \mathbf{i_B})\mathbf{B}^2 \quad \forall \mathbf{B} \in \mathbb{M}^3_{symm},$$

and $\beta_0, \beta_1, \beta_2$ are real-valued functions of $\mathbf{x}$ and the invariants of $\mathbf{B}$, see Equation (5.1).

The second Piola-Kirchhoff stress tensor is given by

$$\mathbf{S}(\mathbf{x}) = \widehat{\mathbf{S}}(\mathbf{x}, \mathbf{F}) = \bar{\mathbf{S}}(\mathbf{x}, \mathbf{F}^{\mathrm{T}}\mathbf{F}),$$

where $\bar{\mathbf{S}}$ *takes the form*

$$\bar{\mathbf{S}}(\mathbf{x}, \mathbf{C}) = \gamma_0(\mathbf{x}, \mathbf{i}_{\mathbf{C}})\mathbf{I} + \gamma_1(\mathbf{x}, \mathbf{i}_{\mathbf{C}})\mathbf{C} + \gamma_2(\mathbf{x}, \mathbf{i}_{\mathbf{C}})\mathbf{C}^2 \quad \forall \mathbf{C} \in \mathbb{M}^3_{symm}.$$

The real-valued functions $\gamma_1, \gamma_2, \gamma_3$ *depend on* $\mathbf{x}$ *and the invariants of* $\mathbf{C}$*, see Equation* (5.1)*.*

If $\bar{\boldsymbol{\sigma}}$ *or* $\bar{\mathbf{S}}$ *are of the above form, the material is isotropic and the axiom of frame-indifference is satisfied.*

Core Concept 5.1: Characterization of isotropic elastic materials

An elastic material has a response function that assigns each $\mathbf{x} \in \Omega$ and each possible deformation gradient $\mathbf{F}(\mathbf{x})$ to a stress tensor, either at the respective point of the deformed configuration $\mathbf{x}^{\boldsymbol{\varphi}} = \boldsymbol{\varphi}(\mathbf{x}) \in \boldsymbol{\varphi}(\Omega)$ (Cauchy stress) or at $\mathbf{x}$ (Piola-Kirchhoff stress, initial configuration). Assuming that the material is isotropic ("behaves independent of the orientation") and as a result of the axiom of material frame-indifference, the response function for the second Piola-Kirchhoff stress tensor is of the form

$$\bar{\mathbf{S}}(\mathbf{x}, \mathbf{C}) = \gamma_0(\mathbf{x}, \mathbf{i}_{\mathbf{C}})\mathbf{I} + \gamma_1(\mathbf{x}, \mathbf{i}_{\mathbf{C}})\mathbf{C} + \gamma_2(\mathbf{x}, \mathbf{i}_{\mathbf{C}})\mathbf{C}^2 \quad \forall \mathbf{C} \in \mathbb{M}^3_{symm},$$

so the tensor can be computed using the right Cauchy-Green tensor $\mathbf{C}$ and the response function which depends only on $\mathbf{C}$ and $\mathbf{x}$:

$$\mathbf{S}(\mathbf{x}) = \bar{\mathbf{S}}(\mathbf{x}, \mathbf{C}).$$

5.2 Hyperelastic materials

In the previous section, we introduced the response function of an elastic material. Hyperelastic material models are a subclass of elastic material models. The properties of these models are defined by a *strain energy function*. It will be demonstrated that the partial derivatives of the strain energy function yield the second Piola-Kirchhoff stress tensor. This property facilitates the analysis and implementation of hyperelastic materials.

Definition 5.7 (Hyperelastic material)**.** An elastic material with response function $\widehat{\mathbf{P}}$ for the first Piola-Kirchhoff stress tensor is called *hyperelastic* if there exists a function

$$\widehat{\psi} : \bar{\Omega} \times \mathbb{M}^3_+ \to \mathbb{R}$$

which is differentiable with respect to every component of $\mathbf{F} \in \mathbb{M}^3_+$ for each $\mathbf{x} \in \bar{\Omega}$, such that

$$(\widehat{\mathbf{P}}(\mathbf{x}, \mathbf{F}))_{ij} = \frac{\partial \widehat{\psi}}{\partial \mathbf{F}_{ij}}(\mathbf{x}, \mathbf{F}), \quad i, j = 1, 2, 3,$$

or in compact notation

$$\widehat{\mathbf{P}}(\mathbf{x}, \mathbf{F}) = \frac{\partial \widehat{\psi}}{\partial \mathbf{F}}(\mathbf{x}, \mathbf{F}).$$

The function $\widehat{\psi}$ is called the *stored energy function* or *strain energy function*.

One can show that under the assumption of a hyperelastic material model, solving the boundary value problem in the equations of balance, see Equation (2.4), is formally equivalent to finding a stationary point of a total energy functional, see [8, p. 142]. Furthermore, the properties of the response function for frame-indifferent, isotropic and homogeneous materials translate to the strain energy function as follows.

- Assuming a frame-indifferent material, $\widehat{\psi}$ is a function of $\mathbf{x}$ and $\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$.
- Additionally assuming an isotropic material, $\widehat{\psi}$ depends only on $\mathbf{x}$ and the invariants $\det(\mathbf{C}), \operatorname{trace}(\mathbf{C})$ and $\det(\mathbf{C}) \operatorname{trace}(\mathbf{C}^{-1})$ of $\mathbf{C}$.

Following [29], we select the strain energy function

$$\psi(I_{\mathbf{C}}, J) := \frac{\mu}{2}(I_{\mathbf{C}} - 3) + \frac{\lambda}{4}\left(J^2 - 1\right) - \left(\frac{\lambda}{2} + \mu\right) \ln J$$

with $I_{\mathbf{C}} = \operatorname{trace}(\mathbf{C}), J = \det(\mathbf{F})$, where the Lamé constants $\lambda = 432.099$ MPa and $\mu = 185.185$ MPa are material parameters. The Lamé parameters of a material are obtained experimentally, as explained in [8, pp. 120]. Since

$$\det(\mathbf{C}) = \det(\mathbf{F})^2 = J^2$$

where $\det(\mathbf{F}) > 0$, the strain energy density function depends only on the invariants of $\mathbf{C}$.

Lemma 5.8. *For a hyperelastic material whose strain energy function is frame-indifferent, the response function for the second Piola-Kirchhoff stress is given by*

$$\tilde{\mathbf{S}}(\mathbf{x}, \mathbf{C}) = 2\frac{\partial \psi}{\partial \mathbf{C}}(\mathbf{x}, \mathbf{C}) \quad \forall \mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}, \mathbf{F} \in \mathbb{M}^3_+.$$

Proof. See [8, pp. 149]. □

Theorem 5.9. *For the strain energy function ψ as defined above, the second Piola-Kirchhoff stress tensor is*

$$\mathbf{S} = \frac{\lambda}{2}(J^2 - 1)\mathbf{C}^{-1} + \mu(\mathbf{I} - \mathbf{C}^{-1}).$$

Proof. The strain energy function is a linear combination of three terms:

$$\psi = \frac{\mu}{2}\underbrace{(I_{\mathbf{C}} - 3)}_{\psi_1} + \frac{\lambda}{4}\underbrace{(J^2 - 1)}_{\psi_2} - \left(\frac{\lambda}{2} + \mu\right)\underbrace{\ln J}_{\psi_3}$$

For the first term ψ_1, we obtain

$$\frac{\partial \psi_1}{\partial \mathbf{C}} = \mathbf{I}$$

without computation. For the second term,

$$\frac{\partial J^2}{\partial c_{ij}} = \frac{\partial \det(C)}{\partial c_{ij}} = \det(\mathbf{C})(\mathbf{C}^{-1})_{ji} = \det(\mathbf{C})(\mathbf{C}^{-1})_{ij}$$

because $\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$ is symmetric. The components of the third term are

$$\begin{aligned}\frac{\partial \ln(\det(\mathbf{C})^{1/2})}{\partial c_{ij}} &= \frac{\partial \det(\mathbf{C})^{1/2}}{\partial c_{ij}} \cdot \frac{1}{\det(\mathbf{C})^{1/2}} \\ &= \det(\mathbf{C})(\mathbf{C}^{-1})_{ij} \cdot \frac{1}{2}(\det(\mathbf{C}))^{-\frac{1}{2}} \cdot \frac{1}{\det(\mathbf{C})^{1/2}} \\ &= \frac{1}{2}(\mathbf{C}^{-1})_{ij}.\end{aligned}$$

By using Lemma 5.8, we obtain

$$\begin{aligned}\frac{1}{2}\tilde{\mathbf{S}} = \frac{\partial \psi}{\partial \mathbf{C}} &= \frac{\mu}{2}\mathbf{I} + \frac{\lambda}{4}\det(\mathbf{C})\mathbf{C}^{-1} - \frac{1}{2}\left(\frac{\lambda}{2} + \mu\right)\mathbf{C}^{-1} \\ &= \frac{\lambda}{4}(\det(\mathbf{C}) - 1)\mathbf{C}^{-1} + \frac{\mu}{2}(\mathbf{I} - \mathbf{C}^{-1}).\end{aligned}$$

□

In Chapter 4, we derived an iterative method for the balance equation which led to the definition of the internal and external load vectors and the stiffness matrix. Taking the hyperelastic material model into account makes these quantities computable: The internal load vector depends on the second Piola-Kirchhoff stress tensor, the deformation gradient and the gradient of the finite element basis functions. With the relation between strain tensor $\mathbf{C}$ and second Piola-Kirchhoff stress tensor $\mathbf{S}$ stated above, all of the discrete versions of these quantities can be computed for a given deformation. The external load vector only depends on the external loads and the gradient of the finite element basis functions. For the stiffness matrix, in

addition to the quantities already mentioned, the elasticity tensor $\mathcal{C}$ is needed. With $\mathbf{S} = 2\frac{\partial \psi}{\partial \mathbf{C}}$, this simplifies to

$$\mathcal{C}_{ijkl} = \frac{\partial \mathbf{S}_{ij}}{\partial \mathbf{E}_{kl}} = \frac{4\partial^2 \psi}{\partial \mathbf{C}_{ij} \partial \mathbf{C}_{kl}},$$

see [5, p. 160].

Core Concept 5.2: Hyperelastic material model under consideration

A hyperelastic material has a strain energy function $\psi : \bar{\Omega} \times \mathbb{M}^3_+ \to \mathbb{R}$ that has the response function of the material as its first derivative:

$$\widehat{\sigma}(\mathbf{x}, \mathbf{F}) = \frac{\partial \widehat{\psi}}{\partial \mathbf{F}}(\mathbf{x}, \mathbf{F}).$$

For our analysis, we use the strain energy function

$$\psi(I_{\mathbf{C}}, J) := \frac{\mu}{2}\left(I_{\mathbf{C}} - 3\right) + \frac{\lambda}{4}\left(J^2 - 1\right) - \left(\frac{\lambda}{2} + \mu\right) \ln J$$

with $I_{\mathbf{C}} = \operatorname{trace}(\mathbf{C}), J = \det(\mathbf{F})$ and Lamé constants $\lambda = 432.099$ MPa and $\mu = 185.185$ MPa.
The second Piola-Kirchhoff stress tensor is

$$\begin{aligned} \mathbf{S}(\mathbf{x}) = \tilde{\mathbf{S}}(\mathbf{x}, \mathbf{C}) &= 2\frac{\partial \psi}{\partial \mathbf{C}}(\mathbf{x}, \mathbf{C}) \\ &= \frac{\lambda}{2}(J^2 - 1)\mathbf{C}^{-1} + \mu(\mathbf{I} - \mathbf{C}^{-1}). \end{aligned}$$

Chapter 6

A Benchmark Problem in Nonlinear Elasticity

Since analytical solutions are rarely available, benchmark problems play a significant role in material modelling. Benchmark problems are boundary value problems with a prescribed geometry, boundary conditions and material parameters that are used to compare the performance of numerical methods. Often, we aim to show-case a specific mechanical effect or a numerical challenge with one benchmark problem. In this chapter, we introduce such a benchmark problem, reference state-of-the-art discretization methods and discuss the numerical solution process. We will later (in Chapter 8) use Cook's membrane to numerically test the new approach for the nonlinear solver.

6.1 Cook's membrane and the locking effect

Cook's membrane is a benchmark problem in computational mechanics. It is used for verification of simulation software and comparison of numerical methods, see for example [3, 29, 23, 32]. The computational domain $\overline{\Omega}$ for Cook's membrane is illustrated in Figure 6.1. All following lengths are in millimeters. The displacement boundary condition $\boldsymbol{\varphi}_0 = \mathbf{0}$ is given on the left side $\Gamma_0 = \{0\} \times (0, 44) \times (-0.5, 0.5)$.

On the right side $\Gamma_1 = \{48\} \times (44, 60) \times (-0.5, 0.5)$, the traction on the boundary is prescribed by

$$\mathbf{g}(\mathbf{x}) = \rho \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^{\mathrm{T}} \mathrm{MPa}, \tag{6.1}$$

where $\rho \in \mathbb{R}$ is the magnitude of the traction. Since the load on the right-hand side introduces a reaction force in the opposite direction at the clamped left-hand side, Cook's membrane is a shearing problem. The elastic material is characterized by the strain energy function that was introduced in the previous chapter, see Core Concept 5.2.

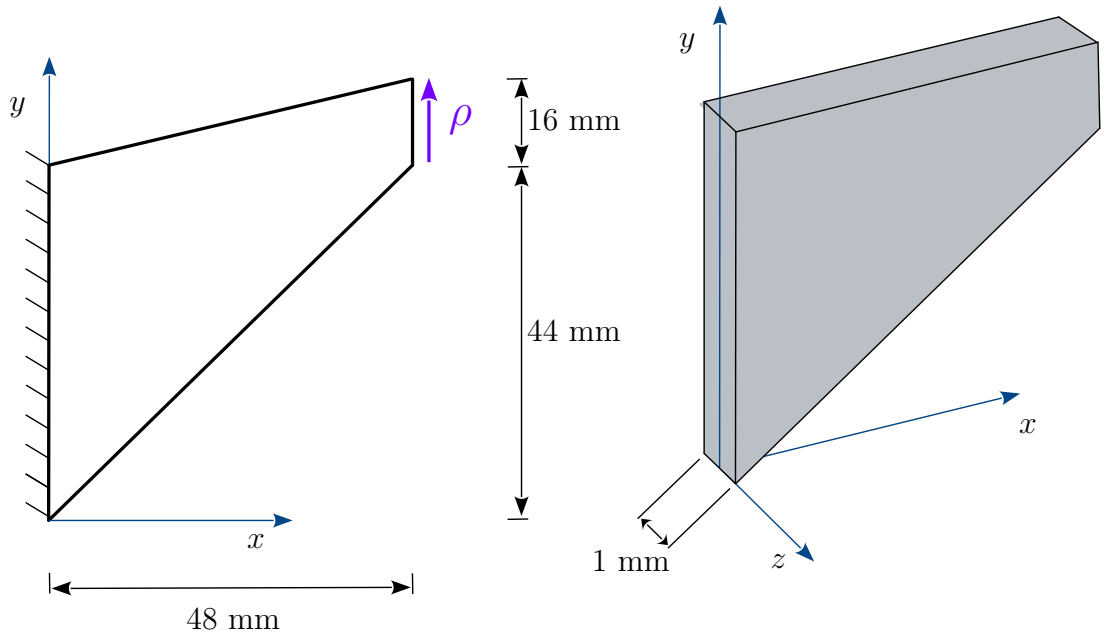


Figure 6.1: Initial geometry and boundary conditions for Cook's membrane. The elastic block is clamped to a wall on the left-hand side, we set a zero displacement boundary condition here. On the right-hand side, a traction load is applied in y-direction.

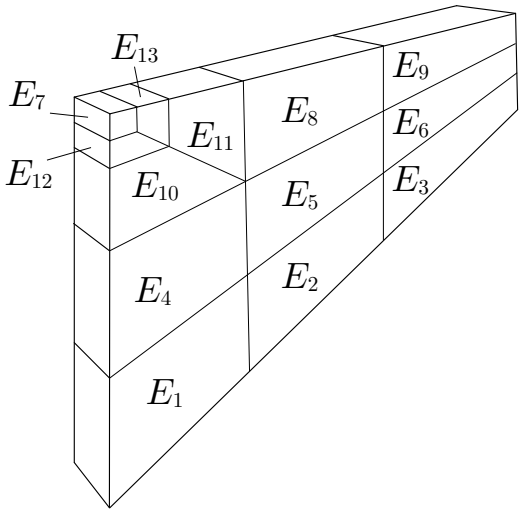


Figure 6.2: A mesh for the Cook's membrane benchmark problem, consisting of 13 finite elements that are refined towards the upper left corner.

Locking. Cook's membrane is used as an example of volumetric locking, see for example [32]. Locking describes the convergence to a numerical solution that exhibits significantly smaller displacement than what is physically reasonable or determined by experimental evidence. According to [7, p. 302], locking can be observed in computational problems with a thin structure or an almost incompressible material discretized with conforming finite elements and a low polynomial degree. If locking is caused by an almost incompressible material, it is referred to as volumetric locking. For thin structures, the term "membrane locking" is used. A finite element method is called *conforming* if the basis functions are admissible deformations, meaning they are included in the set V that was introduced in Equation (3.5). For non-conforming FEM, discontinuous basis functions are permitted.

Locking problems can be overcome by one of the following strategies:

1. Use non-conforming elements.

2. Use a high maximum polynomial degree (high order) for the basis functions.

We discuss discretizations that were proposed to prevent locking for Cook's membrane.

Discontinuous Galerkin. Non-conforming elements are used in the discontinuous Galerkin (DG) method. Several variants of DG methods have been applied to this benchmark problem, see for example [27] and the references therein. In [35], a DG method with low-order elements is used to prevent locking for Cook's membrane. In [18], a variant of DG is used to find the displacement of Cook's membrane. There, the combination of DG with higher order basis functions (e.g., $5 \leq p$) gives higher accuracy (for the same number of DOF) than using cubic basis functions ($p = 3$) with DG, see [18, Figure 18].

High-order FEM. A collection of numerical results for benchmark problems in computational mechanics is given in [29]. In Section 2.3, the authors explore Cook's membrane, the geometry and material selection align with our choice. In [29, Figure 3a], high-order (p-FEM) and low-order discretizations with quadratic shape functions are compared with respect to degrees of freedom and accuracy which is measured by the displacement of the upper right tip of Cook's membrane. We see that high-order elements give the best accuracy when the number of degrees of freedom is less than 10^4. For higher numbers of DOF, the methods give approximately the same tip displacement.

We conclude that the selection of a high-order discretization of hyperelastic problems can prevent the undesirable locking effect. As an alternative, discontinuous basis functions can be used. The combination of both approaches has also been explored, and it has been demonstrated that a high polynomial degree of the basis functions can support accuracy in DG methods.

Mixed FEM. Another recent alternative to traditional FEM are mixed Finite Element methods. Here, additional quantities such as the deformation gradient are treated as independent variables. The physical connection, e.g., between deformation and deformation gradient, is enforced by a Lagrange multiplier, see [24, Chapter 12] for an overview of constrained optimization using Lagrange multipliers. In [23], a mixed FEM with the deformation gradient $\mathbf{F}$ or the Cauchy-Green tensor as an additional independent variable are introduced and numerical tests on Cook's membrane are shown. In [23, Figure 5.6], the accuracy of the $\mathbf{F}$-based method is demonstrated.

Minimization problem. For hyperelastic materials, the equations of balance can be written as a minimization problem since the response function is the component-wise derivative of the stored energy function, see Definition 5.7. We again refer to [8, pp. 142] for a detailed analysis.

Conclusion. For our numerical tests, we select high-order conforming finite elements. High-order finite elements, that is, finite elements with $p > 2$, prevent locking. Comparing to a low-order model with the same number of degrees of freedom shows that the high-order models exhibit better accuracy. DG methods are not applied, but since high-order elements improved the accuracy of the DG method in [18], we expect that our results can be extended to DG methods. Mixed formulations require adaption of the variational problem and are therefore out of the scope of this thesis. To keep the nonlinear solver applicable to a wide range of materials, we maintain the formulation as a system of nonlinear equations. It should nevertheless be noted that a formulation as a minimization problem typically leads to better convergence results.

6.2 Load step Newton method for Cook's membrane

In the previous chapter, we concluded that the quantities needed in Newton's method as described in Core Concept 3.1 are computable if the stored energy function of the material is given. We use hierarchical, continuous high-order basis functions (Chapter 4.3). We select the basis functions according to the trunk space, see Definition 4.12.

Using the basis functions in a Galerkin ansatz yields the discrete equations in Core Concept 4.1 and the global stiffness matrix in Equation (4.11). Given a mesh on $\overline{\Omega}$ (Figure 6.2), the discretized Newton's method for the balance equation is described in Algorithm 6.1. As a termination criterion, we use the normalized norm of the (negative) residual, see Line 9 of Algorithm 6.1. We terminate the Newton

iteration if

$$\frac{\|\mathbf{R}\|_2}{\|\mathbf{E}_{ext}\|_2} < tol_{Newton}, \tag{6.2}$$

where $\mathbf{E}_{ext}$ is the external energy vector, see Core Concept 4.1. For part of the numerical tests, the absolute value of the scalar product of the negative residual and the Newton update $\Delta\mathbf{u}$ (see Line 10 of Algorithm 6.1) is used as a termination criterion:

$$|\langle \mathbf{R}, \Delta\mathbf{u} \rangle| < tol_{Newton}. \tag{6.3}$$

Implementation. The algorithm is implemented in C++ using the finite element framework `AdhoC++` (Advanced high-order finite element Code) that was developed at the Chair for Computation in Engineering at the Technical University Munich (TUM) and has since been continuously expanded and optimized at TUM and TUHH. A brief description of the code structure is given in [19, Section 3.3]. A more comprehensive overview of the implementation of high-order finite element and finite cell methods is available in [37] for the Matlab framework `FCMLab`, the structure of the code is closely related to that of `AdhoC++`. C++ is an object-oriented programming language. The algorithms presented in this and the next chapter rely on the definition of an `initialBoundaryValueProblem` object that is initialized with the geometry, mesh, material model and boundary conditions for Cook's membrane. The `initialBoundaryValueProblem` object then provides methods that retrieve the global stiffness matrix and external and internal energy vectors which can then be used in the nonlinear solver.

Displacement boundary conditions. In `AdhoC++`, the displacement boundary conditions are applied once the mesh, basis functions and degrees of freedom have already been generated. This allows for large flexibility, for example with time-dependent boundary conditions. However, it leads to degrees of freedom whose values are already determined by the displacement boundary conditions. Options for accounting for these boundary conditions are:

1. Remove the row and columns that correspond to pre-determined DOF from the global stiffness matrix.
2. Modify the matrix and right-hand side to ensure that the solution of the linear system has the prescribed values.
3. Add a penalty term for degrees of freedom that violate the boundary condition.

The first approach produces a smaller linear system, but it has the disadvantage of requiring the size of the global stiffness matrix to be modified at runtime, which is computationally inefficient. The third approach results in a large condition number

of the global stiffness matrix [4, p. 349]. Therefore, the global stiffness matrix and the right-hand side (the vectors $\mathbf{E}_{ext}$ and $\mathbf{E}_{int}$) are modified directly. A detailed description is available in Appendix B. In the following, $\mathbf{K}(\mathbf{u})$, $\mathbf{E}_{ext}$ and $\mathbf{E}_{int}$ are implicitly modified to account for the displacement boundary conditions, unless stated otherwise.

Newton's method. The number of Newton iterations for different choices of maximum polynomial degree and applied load is shown in Figure 6.3. We observe that the algorithm fails to converge for some combinations, for example for traction $\rho = 50$ and polynomial degree $p = 5$. Also note that the number of Newton iterations is, for all choices of the applied traction, larger in the high-order models compared to the low-order model with $p = 1$. Therefore, constructing effective alternatives to Newton's method is especially relevant for high-order discretizations.

Load step Newton method. We introduce the load step Newton (LSN) method, which is a standard approach in computational mechanics, see [34, p. 158] and [6, Section 2.4]. The load is applied in increments: Instead of the full traction magnitude in Equation (6.1), we use a scaled boundary condition $\lambda_1 \mathbf{g}(\mathbf{x}), \lambda_1 \in (0, 1]$. A small value of λ_1 facilitates the convergence of Newton's method. The numerical solution for the scaled boundary condition is used as a starting vector for a new run of Newton's method, for which we increase the magnitude of the traction boundary condition to $\lambda_2 \rho$, where $\lambda_2 \in (\lambda_1, 1]$. This process is repeated until the full magnitude of the boundary condition is applied.

An advantage of the load step Newton method is that the solutions of the subproblems have physical meaning: They are the solutions to displacement problems with a scaled external load. In an engineering application, the stress and displacement states for a growing load may be of interest. For problems involving a more general material (e.g., metals modeled with elasto-plastic materials), the load-displacement curve may even be necessary to find the current load since the deformation then depends on previous stress states because previous deformations might not be reversible.

The load step Newton method introduces an additional loop over the load increments and facilitates the convergence of all Newton's methods. We write the load factors $\lambda_1, \lambda_2, \ldots$ into an array

$$\Lambda = (\lambda_1, \lambda_2, \ldots, 1). \tag{6.4}$$

The load step Newton method is shown in Algorithm 6.2. If the termination criterion (6.2) is used, it has to be adapted, since the external load vector is scaled with the load factor λ in the computation of the negative residual (Line 11 of Algorithm 6.2). The new termination criterion is

$$\frac{\|\mathbf{R}\|_2}{\|\lambda \mathbf{E}_{ext}\|_2} < tol_{Newton}. \tag{6.5}$$

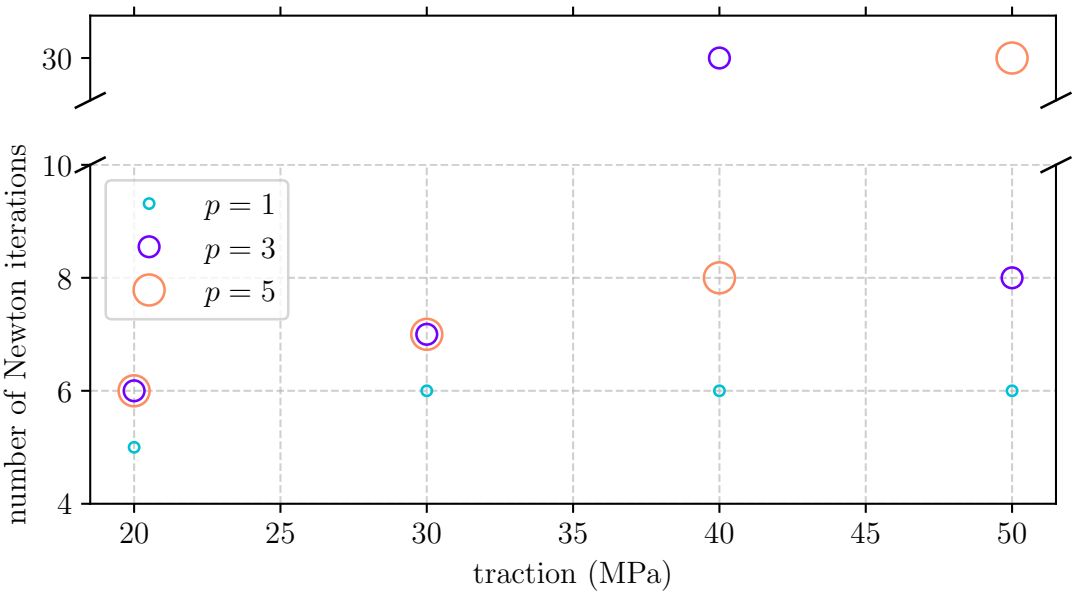


Figure 6.3: The convergence of Newton's method depends on the applied load (traction) and on the polynomial degree of the discretization. Increasing the traction on the right-hand side of Cook's membrane leads to more Newton iterations or even divergence of the method, indicated by $k_{max} = 30$ Newton iterations.

For a traction of $\rho = 50$ and a maximum polynomial degree $p = 5$, we set the load factors to

$$\Lambda = \begin{pmatrix} 0.25, & 0.5, & 0.75, & 1 \end{pmatrix},$$

which is sufficient to reach convergence for all four Newton runs. The numerical solutions to the load steps are shown in Figure 6.4. We observe that the applied load causes the solid body to bend upwards.

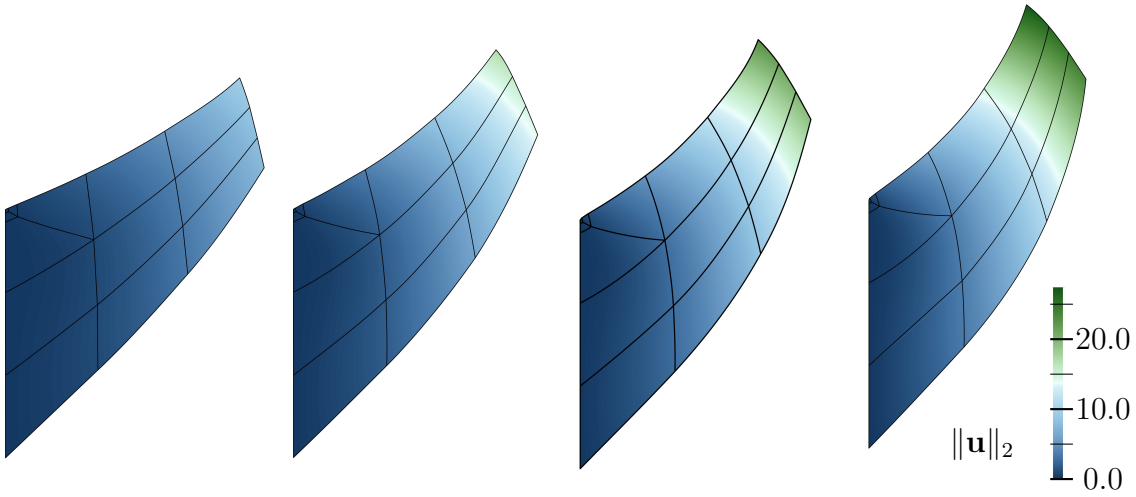
The classical load-stepping method is intuitive since deformations do not happen "at once" but the position of a particle in time changes in a continuous way. From a numerical point of view, the new starting vectors are close to the solution and therefore facilitate the convergence of Newton's method. However, the computational effort is high, especially in cases where the load increments that lead to convergence need to be small. In Section 6.1, we concluded that high-order discretizations can prevent locking. In Table 6.1, the number of degrees of freedom for maximum polynomial degrees $p = 2, \ldots, 5$ is shown. The choice of high-order basis functions increases computational costs significantly. This is due to the fact that the number of degrees of freedom determines the size of the vectors $\mathbf{E}_{ext}$ and $\mathbf{E}_{int}(\mathbf{u})$ and the matrix $\mathbf{K}(\mathbf{u})$. In Figure 6.5, we see that for high-order discretizations of Cook's membrane, the computation of the matrices $\mathbf{K}(\mathbf{u})$ dominates the computation time. For a maximum polynomial degree $p = 3$, 90% of the CPU time

Algorithm 6.1 Discretized Newton's method for the balance equation

Require: Maximum Newton steps $k_{\max}$, tol_{Newton}, order of the trunk space p
1: Set $\mathbf{u} = \mathbf{0}$. ▷ Initial displacement
2: Initialize an `adhocpp::initialBoundaryValueProblem IVP` with mesh $\mathcal{M}$ and trunk space basis functions of maximum degree p.
3: Set traction on Γ_1 in direction $(0, 1, 0)^{\mathrm{T}}$ for the `IVP`.
4: Initialize zero displacement boundary condition on Γ_0 for the `IVP`.
5: Compute external energy vector $\mathbf{E}_{\text{ext}}$ from the `IVP`. ▷ See Equation (4.9)
6: **for** $k = 0, 1, \ldots, k_{\max}$ **do**
7: Compute global stiffness matrix $\mathbf{K}(\mathbf{u})$ from the `IVP`. ▷ See Equation (4.11)
8: Compute internal energy vector $\mathbf{E}_{\text{int}}(\mathbf{u})$ from the `IVP`. ▷ See Equation (4.8)
9: $\mathbf{R} \leftarrow \mathbf{E}_{\text{ext}} - \mathbf{E}_{\text{int}}(\mathbf{u})$
10: $\Delta\mathbf{u} \leftarrow \text{linearSolver}(\mathbf{K}(\mathbf{u}), \mathbf{R})$. ▷ Solves linear system $\mathbf{K}(\mathbf{u})\Delta\mathbf{u} = \mathbf{R}$.
11: $\mathbf{u} \leftarrow \mathbf{u} + \Delta\mathbf{u}$
12: **if** termination criterion $< \text{tol}_{Newton}$ **then** ▷ See Equations (6.2) and (6.3)
13: break
14: **end if**
15: **end for**

Algorithm 6.2 Load step Newton method **(LSN)**

Require: Maximum Newton steps $k_{\max}$, tol_{Newton}, order of the trunk space p
Require: Load step array Λ of length $i_{\max}$, $0 < \Lambda[i] < \Lambda[i+1]$ and $\Lambda[i_{\max}] = 1$
1: Set $\mathbf{u} = \mathbf{0}$.
2: Initialize an `adhocpp::initialBoundaryValueProblem IVP` with mesh $\mathcal{M}$ and trunk space basis functions of maximum degree p.
3: Set traction on Γ_1 in direction $(0, 1, 0)^{\mathrm{T}}$ for the `IVP`.
4: Initialize zero displacement boundary condition on Γ_0 for the `IVP`.
5: Compute $\mathbf{E}_{\text{ext}}$ from the `IVP`.
6: **for** $i = 1, \ldots, i_{\max}$ **do**
7: $\lambda = \Lambda[i]$
8: **for** $k = 0, 1, \ldots, k_{\max}$ **do** ▷ i'th run of Newton's method
9: Compute global stiffness matrix $\mathbf{K}(\mathbf{u})$ from the `IVP`.
10: Compute internal energy vector $\mathbf{E}_{\text{int}}(\mathbf{u})$ from the `IVP`.
11: $\mathbf{R} \leftarrow \lambda\mathbf{E}_{\text{ext}} - \mathbf{E}_{\text{int}}(\mathbf{u})$
12: $\Delta\mathbf{u} \leftarrow \text{linearSolver}(\mathbf{K}(\mathbf{u}), \mathbf{R})$.
13: $\mathbf{u} \leftarrow \mathbf{u} + \Delta\mathbf{u}$
14: **if** termination criterion $< \text{tol}_{Newton}$ **then** ▷ See Equations (6.5), (6.3)
15: break
16: **end if**
17: **end for**
18: **end for**



(a) Load step 1 (12.5 MPa applied) (b) Load step 2 (25 MPa applied) (c) Load step 3 (37.5 MPa applied) (d) Load step 4 (50 MPa applied)

Figure 6.4: Displacement of Cook's membrane, computed in 4 load steps. The normalized norm of the residual (Equation (6.2)) is used as the termination criterion. In load step 4, the full load ρ is applied. The color bar shows the magnitude of the displacement in millimeter.

is spent on numerical integration, assembly and boundary condition enforcement on $\mathbf{K}(\mathbf{u})$. The share of the computation time used to set up the matrix grows with the discretization order.

Therefore, in order to reduce the computational effort for finding numerical solutions to high-order discretizations of Cook's membrane, we focus on the reduction of the set-up time for the matrices $\mathbf{K}(\mathbf{u})$. In particular, computation time for solving linear systems is negligible for the problems under consideration.

Table 6.1: Degrees of freedom for the discretizations of Cook's membrane with order p for the hierarchical basis functions. Basis functions are selected according to the trunk space, see Definition 4.12.

order of the trunk space $p =$	2	3	4	5
DOF (for 13 elements)	402	672	1,122	1,752

6.3 Modifications of the load step Newton method

As discussed in the previous subsection, there is a need for efficient modifications to Newton's method in order to address the significant increase in computational

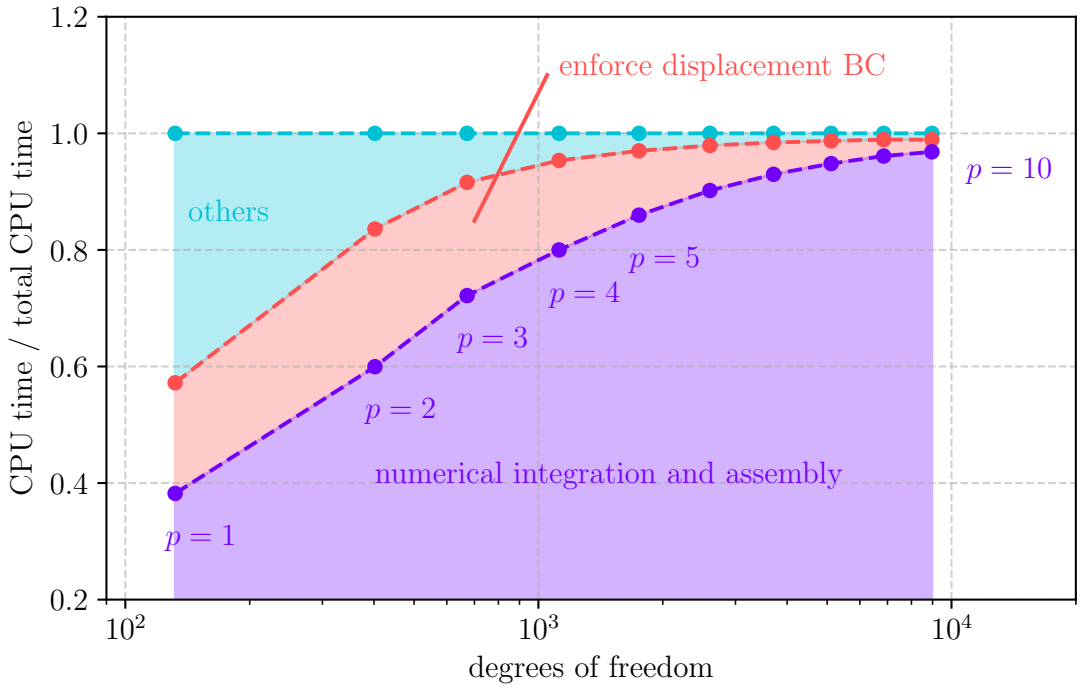


Figure 6.5: (Relative) CPU time for computation of the matrix $\mathbf{K}(\mathbf{u})$, the enforcement of boundary conditions and all other operations (including the solver for the linear systems) in Newton's method for Cook's membrane. Discretizations with maximum polynomial degree $p = 1, 2, \ldots$ according to the trunk space are used with 13 finite elements, see Figure 6.2. The applied traction is $\rho = 20$ and convergence is reached after 6 Newton iterations (except for $p = 1$, where 5 iterations are sufficient).
The set-up of $\mathbf{K}(\mathbf{u})$ in each iteration (Line 7 in Algorithm 6.1) consists of two steps. First, numerical integration and assembly take place. The resulting system of equations does not necessarily yield a solution that satisfies the displacement boundary conditions. Therefore, $\mathbf{K}(\mathbf{u}), \mathbf{E}_{ext}$ and $\mathbf{E}_{int}(\mathbf{u})$ are modified to ensure a numerical solution with the prescribed displacement values at the boundary as described in Appendix B. The total time for the computation of the matrices $\mathbf{K}(\mathbf{u})$ is the sum of the violet and red part of the graph.

effort required for high-order discretizations of elastic displacement problems.

The load step Newton method is a widely accepted approach for solving stationary displacement problems in elasticity. It is also applicable to a broad class of problems in solid mechanics, including elasto-plastic problems. Since the algorithm is closely related to Newton's method for root-finding, modifications of Newton's method (that are not specific to solid body displacement problems) like line search or quasi-Newton methods can be applied. Other ideas, like arc-length methods, are specific to stationary displacement problems. We again refer to the book of Wriggers [34], especially to Chapter 5, for an overview of established ideas.

Line search. As an alternative to load steps, the original Newton's method might be adapted by changing Line 11 of Algorithm 6.1 to

$$\mathbf{u} \leftarrow \mathbf{u} + \alpha \Delta \mathbf{u},$$

where the step width $0 < \alpha \leq 1$ is selected in every iteration. This idea is driven by the construction of Newton's method, see Section 3.1. As we replace the original nonlinear function by an affine-linear function that uses the first-order term of the multivariate Taylor series, we assume that the root of the affine-linear function is in a vicinity of a solution to our original problem. Line search algorithms provide a control mechanism for that assumption. A naive idea would be to select α such that

$$\|\mathbf{R}(\mathbf{u} + \alpha \Delta \mathbf{u})\| < \|\mathbf{R}(\mathbf{u})\|$$

in some norm $\|\cdot\|$ and for every Newton step. It can be shown that there are convex minimization problems for which Newton's method with naive line search diverges [24, p. 32]. Therefore, more sophisticated conditions have been investigated, see [24, p. 287] for an overview of line search methods on nonlinear equations. A widely accepted choice is given by the Wolfe conditions [24, pp. 32–33]. Line search was implemented for Cook's membrane according to [26, pp. 478–480] but did not improve convergence in our implementation.

Quasi-Newton methods. In Line 12 of Algorithm 6.2, a linear system with matrix $\mathbf{K}(\mathbf{u})$ is solved to obtain the Newton update $\Delta \mathbf{u}$. Methods that approximate or replace the matrix in this step are referred to as quasi-Newton methods. A first approach of replacing the global stiffness matrix with the initial matrix (meaning we move Line 9 in Algorithm 6.2 out of the inner loop into the outer loop), did not lead to convergence of the Newton runs for Cook's membrane. Updating the global stiffness matrix only every other iteration also resulted in divergent runs of Newton's method.

The quasi-Newton method BFGS (Broyden-Flechter-Goldfarb-Shanno) is based on a positive definite update of the approximate Jacobian or Hessian, see [10, p. 201]. BFGS has been applied to Cook's membrane in [2]. For first-order elements

($p = 1$), BFGS reduced computation time in comparison to the standard Newton-Krylov method (that is equivalent to Algorithm 6.2 if we use an iterative Krylov solver in Line 12) to about 80%, see [2, Table 2]. For second-order elements ($p = 2$), the computation time of BFGS is reduced to 95% of the benchmark time (Newton-Krylov).

These findings suggest that the performance of BFGS on high-order models ($p > 2$) may be suboptimal, but we know of no numerical evidence to confirm or deny this claim.

Arc-length method. Selecting the load step array Λ (see Equation (6.4)) for the load step Newton method often involves a process of trial and error. The load increments $\Lambda[i+1]-\Lambda[i]$ should be chosen small enough to enable the convergence of Newton's method. On the other hand, small load increments increase computational effort. The sub-problems in Algorithms 6.2 are root-finding problems

$$\underbrace{\mathbf{E}_{int}(\mathbf{u}) - \lambda \mathbf{E}_{ext}}_{=:\mathbf{G}(\mathbf{u})} = \mathbf{0},$$

with $\lambda \in (0, 1]$. The negative residual $\mathbf{R}(\mathbf{u}) = \lambda \mathbf{E}_{ext} - \mathbf{E}_{int}(\mathbf{u})$ is the right-hand side of the linear system for the Newton step and the Jacobian of $\mathbf{G}$ at $\mathbf{u}$ is $\mathbf{K}(\mathbf{u})$. Arc-length methods in FEM go back to Crisfield [9]. The parameter λ is set as an unknown, as opposed to the conventional approach of predefining the array Λ. This leads to $\mathbf{G}$ as a function of both $\mathbf{u}$ and λ. Let n be the number of DOF, so that $\mathbf{K}(\mathbf{u}) \in \mathbb{R}^{n \times n}$. To obtain a quadratic Jacobian, a scalar constraint $g : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ is added such that the root-finding problem is

$$\begin{pmatrix} \mathbf{G}(\lambda, \mathbf{u}) \\ g(\lambda, \mathbf{u}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

which leads to the linear system

$$\begin{pmatrix} \mathbf{K}(\mathbf{u}) & \mathbf{E}_{ext} \\ * & * \end{pmatrix} \begin{pmatrix} \Delta \mathbf{u} \\ \Delta \lambda \end{pmatrix} = \begin{pmatrix} \lambda \mathbf{E}_{ext} - \mathbf{E}_{int}(\mathbf{u}) \\ g(\lambda, \mathbf{u}) \end{pmatrix}$$

in the Newton step. The linear system can be solved with a Schur complement approach. In [34, Table 5.1], examples of constraint conditions g are given.

We know of no numerical evidence that shows the success of arc-length methods for high-order discretizations of Cook's membrane. The arc-length method has been applied to Cook's membrane with linear ($p = 1$) two-dimensional elements in [22]. But since the behavior of the nonlinear solver depends on the polynomial degree of the basis functions, see Figure 6.3, this result has limited significance for high-order problems.

In conclusion, the line-search method and naive approaches to quasi-Newton methods failed to converge for high-order discretizations of Cook's membrane. Consequently, there is a requirement for customized solvers to reduce the computational

effort. The arc-length method may be a good candidate. For the BFGS method, based on the results for low-order methods, it is not clear that the method would improve the solution process in terms of computation time.

6.4 Sparsity pattern of the global stiffness matrix

In this subsection, we characterize the global stiffness matrix in terms of sparsity and structure. The purpose of this analysis is to develop a customized nonlinear solver for high-order discretizations of Cook's membrane.

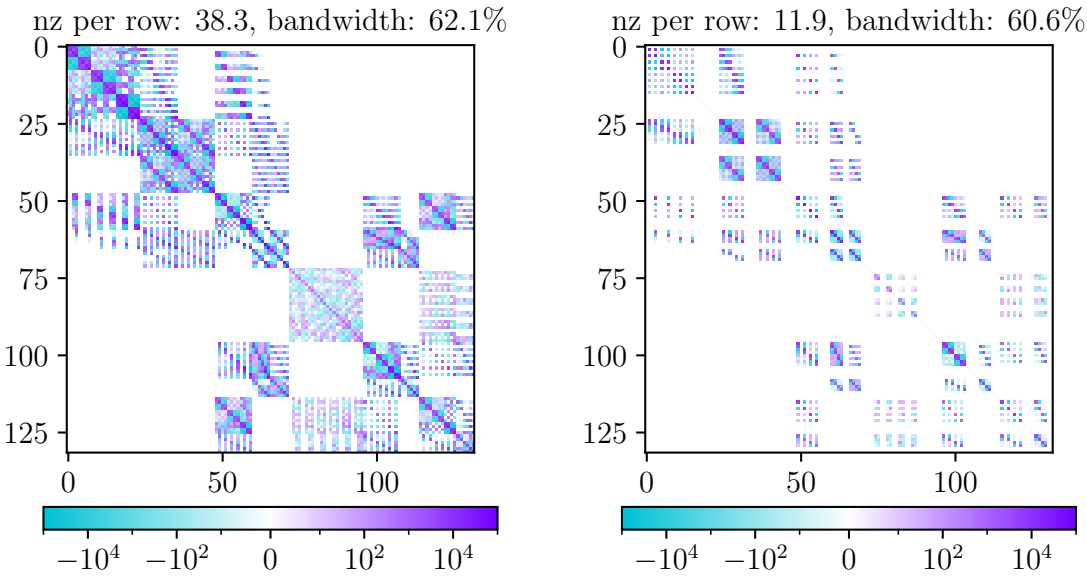
The average number of non-zero elements per row (nz per row) and the bandwidth are computed as follows for a quadratic matrix $\mathbf{K}(\mathbf{u}) \in \mathbb{R}^{n \times n}$:

$$\begin{aligned} \text{nz per row} &= \frac{1}{n} \# \left\{ \mathbf{K}(\mathbf{u})_{i,j} : |\mathbf{K}(\mathbf{u})_{i,j}| > 10^{-16}, 1 \leq i, j \leq n \right\}, \\ \text{bandwidth} &= \max_{1 \leq i,j \leq n} \left\{ |i-j| : |\mathbf{K}(\mathbf{u})_{i,j}| > 10^{-16} \right\}. \end{aligned}$$

If the bandwidth is given as a percentage, this refers to the matrix size n. Since the matrices are implemented in a sparse compressed row-major format (CSR), the number of non-zero entries of the stiffness matrix is connected to the memory requirement of the solution algorithm. A small bandwidth of the global stiffness matrix is beneficial for direct solvers such as Gaussian elimination since the matrix LU decomposition preserves the bandwidth.

Coarse mesh. Figure 6.6 shows the global stiffness matrix $\mathbf{K}(\mathbf{u})$ for a first-order discretization (only node modes are used, see Sections 4.3 and 4.4). We see that enforcing the displacement boundary condition reduces the average number of non-zeros per row from 38.3 to 11.9.

For a discretization with trunk space order $p = 2$ and higher, the sparsity patterns are shown in Figure 6.7. We see that the block structure is similar to the first-order discretization. The block structure of the global stiffness matrix on the coarse mesh (for $p = 1$, see Figure 6.6) is shown in Figure 6.8, with all non-zero blocks in gray. The block structure is connected to the mesh. The highlighted block row/column corresponds to degrees of freedom that are located in element E_2, see Figure 6.9. A non-zero block appears in the sparsity pattern if the respective DOF belong to neighbor elements. The matrix contains blocks of different sizes. This is because of the shared node basis functions that ensure continuity of the finite element solution and were introduced in Section 4.4. The degrees of freedom are assigned according to the element indices in Figure 6.9 (the complete labeling of the elements is included in Figure 6.2). For element E_1, node basis functions are assigned to all 8 nodes, resulting in 24 degrees of freedom, which is the size of the top left block in Figure 6.8. Element E_2 has four shared nodes with E_1. Thus, only $4 \cdot 3$ degrees of freedom are added for this element, which gives the block size of the dark violet block.



(a) Structure of the global stiffness matrix after numerical integration and assembly.

(b) Final structure after enforcement of the boundary condition.

Figure 6.6: Global stiffness matrix $\mathbf{K}(\mathbf{u})$ in Newton's method for a discretization with first-order finite elements ($p = 1$) before and after enforcing the displacement boundary condition. The matrix entries are shown on a symmetric logarithmic scale with linear threshold $T_{lin} = 10$: Matrix entries smaller than -10 are shown as $-\log(-\mathbf{K}(\mathbf{u})_{ij})$. The scale from -10 to 10 is linear. Entries larger than 10 are plotted logarithmically.

Increasing the order of the trunk space while keeping the mesh fixed leads to growing non-zero blocks, as the DOF connected to the higher-order basis functions are added. The size of the blocks relative to the total number of DOF stays roughly the same, compare to the block structure of the global stiffness matrices in Figure 6.7. Therefore, the relative bandwidth is about 60% for all matrices in Figures 6.6(a) and 6.7.

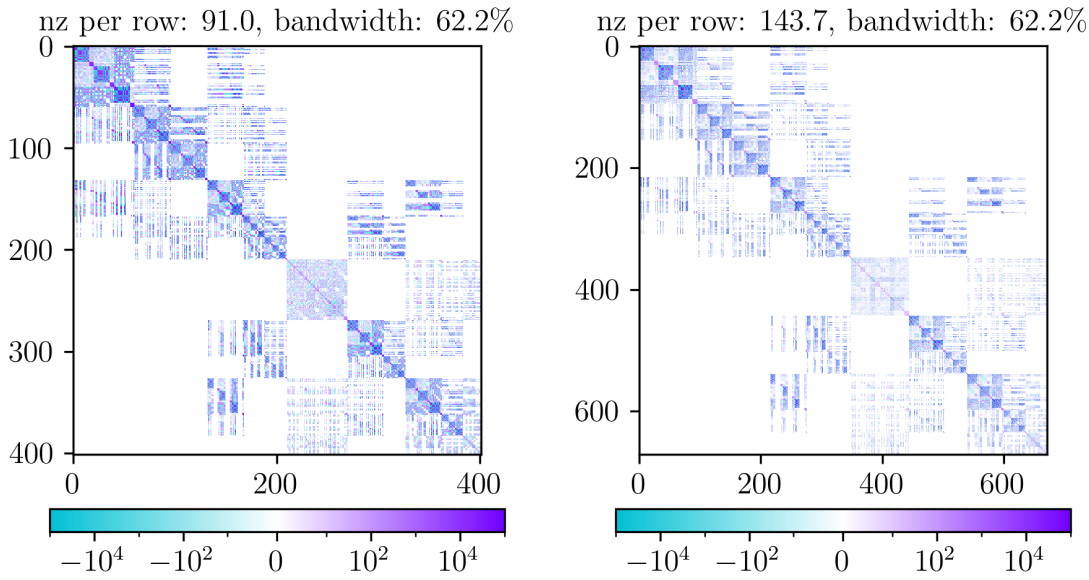
We saw in the previous section that the set-up and assembly of the global stiffness matrix is the bottleneck of Algorithm 6.1. The large number of non-zero entries of the matrices for the high-order discretizations introduces additional memory load (when compared to a low-order discretization on a fine mesh). More importantly, it makes the set-up of the matrix expensive.

Fine mesh. An alternative mesh with small element size is shown in Figure 6.10. A low-order discretization on this mesh results in global stiffness matrices with an average of 46.6 to 107.3 non-zero entries per row, see Figure 6.11.

Refining the mesh leads to global stiffness matrices with a smaller bandwidth (roughly 20% vs. 60% for the coarse mesh). This is because the number of elements is significantly higher (104 elements) and non-zero blocks appear in the sparsity pattern only for neighbor elements. Figure 6.12 shows the bandwidth of the global stiffness matrix for both the fine and the coarse mesh. For low polynomial degrees ($p = 1, 2, 3$), the bandwidth increases slower than a quadratic rate. In the range from $p = 3$ to $p = 6$, the bandwidth of both meshes behaves approximately quadratic. This behavior can be explained by the structure of the trunk space. For $p < 6$, only node, edge and face modes are included, and their total number grows quadratically with p if the mesh is kept fixed. Internal modes are introduced only for trunk spaces of order $p > 5$ (see Table 4.1), which leads to a change in growth behavior.

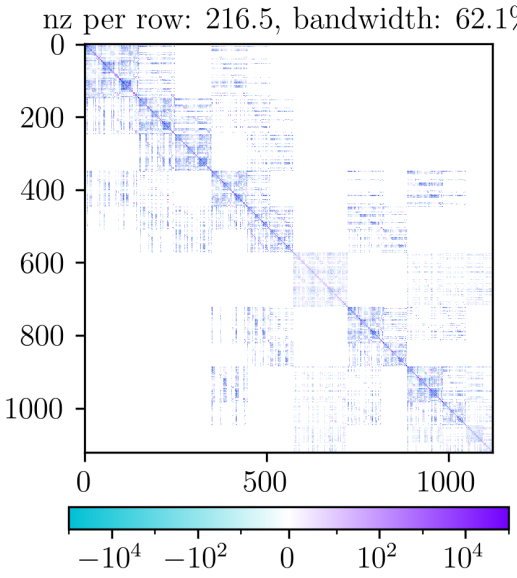
The high-order discretization in Figure 6.7(b) and the low-order discretization in Figure 6.11(a) have similar size ($n = 672$ vs. $n = 762$) and are therefore well-suited for direct comparison. The matrix in Figure 6.7(b) has an average of 143.7 non-zero entries per row (vs. 46.6 for the matrix from the first-order discretization). When solving a linear system with each matrix, e.g., using Gaussian elimination, we expect more fill-in for the matrix from the high-order discretization due to its larger bandwidth. While the bandwidth could be reduced by applying a reordering method such as the Cuthill-McKee algorithm [11, pp. 158 – 161], we have not pursued this approach for the following reason: For Cook's membrane, the computation time for the direct linear solver is negligible, and the fill-in does not cause a significant increase in memory.

In conclusion, while high-order discretizations prevent the undesirable locking effect, the global stiffness matrices resulting from high-order discretizations are less sparse than their low-order counterparts. Consequently, they require more computational resources (storage and computation time) for assembly and numerical integration.



(a) $p = 2$

(b) $p = 3$



(c) $p = 4$

Figure 6.7: Global stiffness matrix for three discretizations of Cook's membrane with maximum order $p = 2, 3, 4$ of the trunk space. The matrix is shown after numerical integration and assembly, before enforcement of the displacement boundary condition. Enforcing the boundary condition removes some of the non-zero entries. The matrix entries are shown on a symmetric logarithmic scale with linear threshold $T_{lin} = 10$.

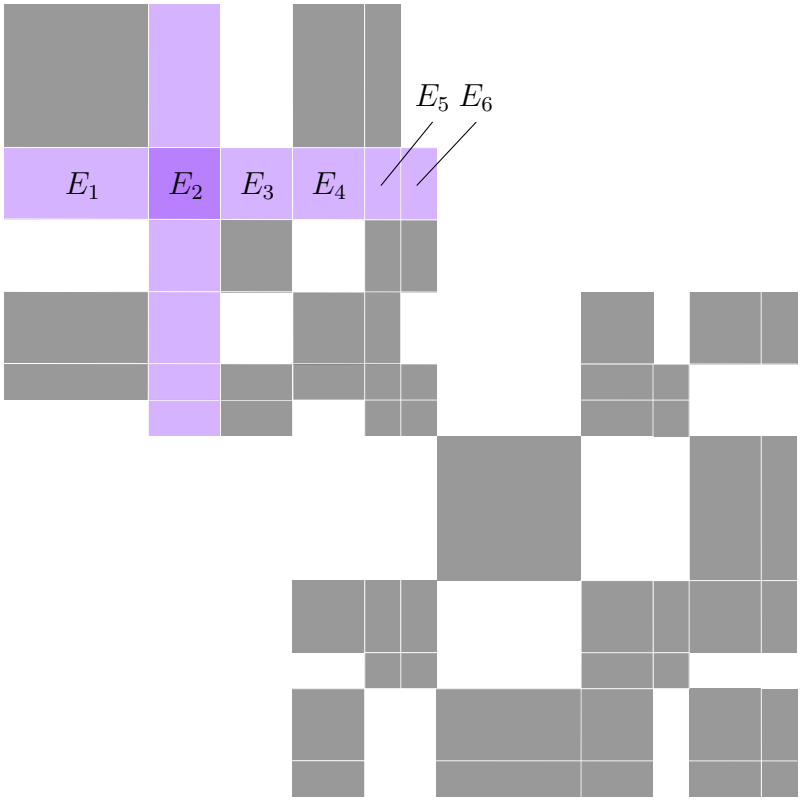


Figure 6.8: Block structure of the global stiffness matrix for the mesh with 13 elements and $p = 1$. The block row/column corresponding to element E_2 is highlighted. The degrees of freedom are labeled with the lowest element index of the neighbor elements. E.g., a degree of freedom of a shared node basis function that is non-zero on elements E_1 and E_2 is included in the E_1 block.

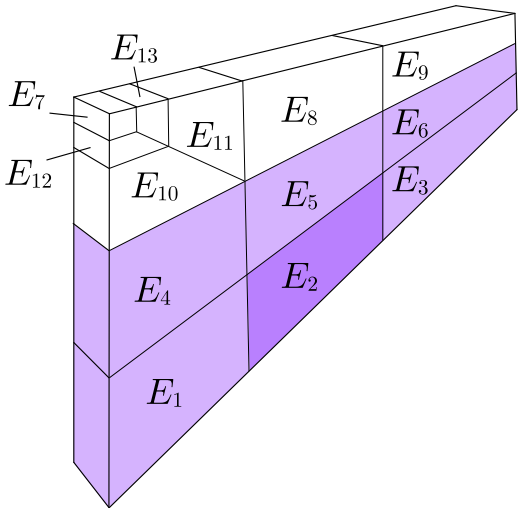


Figure 6.9: We highlight element E_2 and its five neighbor elements in the mesh with 13 elements.

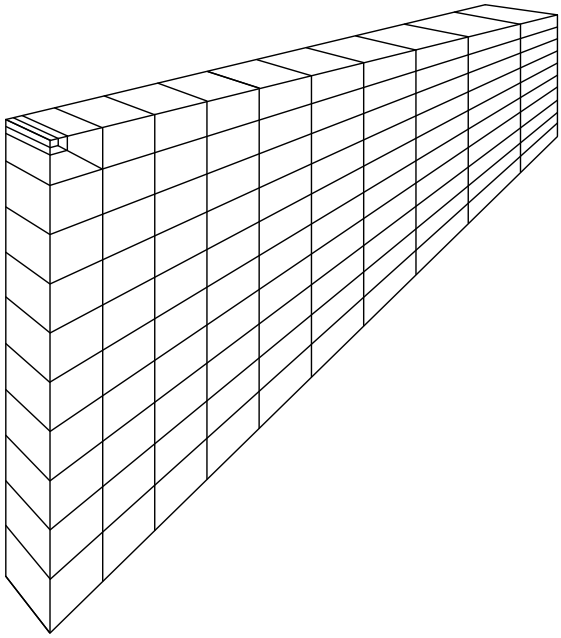
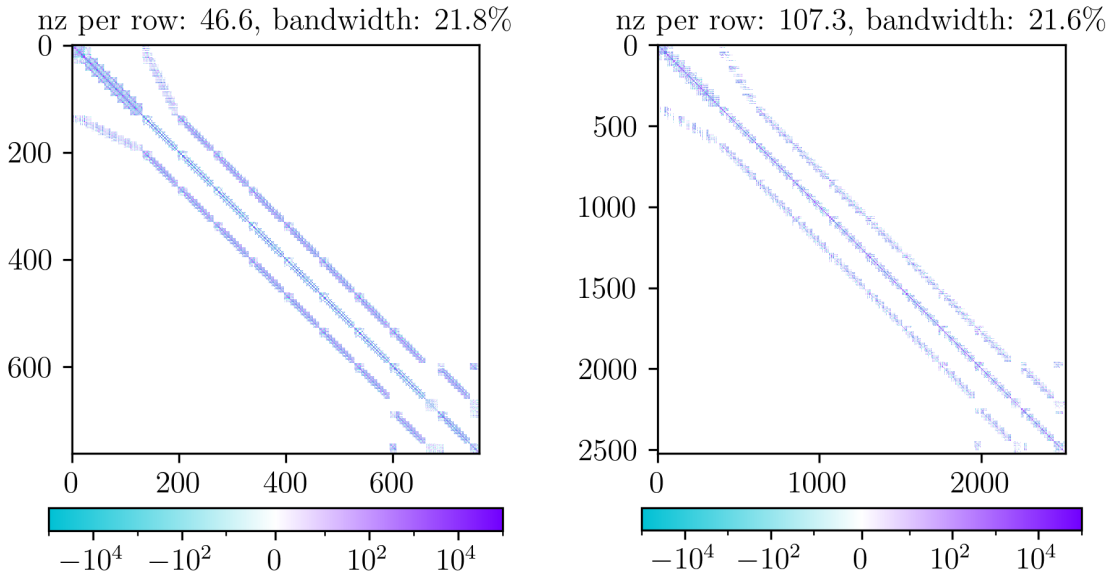


Figure 6.10: Refined mesh for Cook's membrane with 104 elements.



(a) $p = 1$ (b) $p = 2$

Figure 6.11: Sparsity patterns of the global stiffness matrix with low-order discretizations of Cook's membrane using the refined mesh with 104 elements before enforcing the displacement boundary condition. The matrix entries are shown on a symmetric logarithmic scale with linear threshold $T_{lin} = 10$.

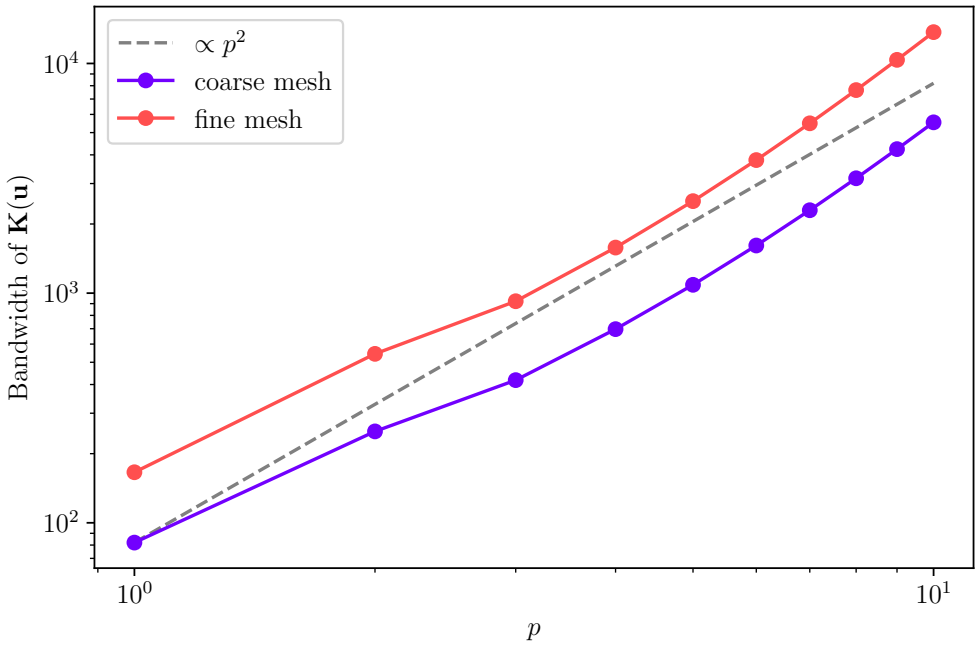


Figure 6.12: The bandwidth of the global stiffness matrix $\mathbf{K}(\mathbf{u})$ depends on the order of the trunk space p. The slope of the dashed line is 2.

Chapter 7

Surrogate Load Step Newton Method

In the previous chapter, we discussed that high-order discretizations of our model problem give numerical results with high accuracy when compared to a low-order fine mesh discretization with the same number of degrees of freedom.

On the other hand, the computational effort of the nonlinear solver is severely increased for high-order discretizations. This is because of the load steps that are necessary to obtain convergence of Newton's method and the associated set-up costs for the global stiffness matrices. Additionally, the global stiffness matrices are less sparse than in the low-order case. In this chapter, we will motivate a new approach to efficiently solve the nonlinear problem numerically, based on the hierarchical structure of high-order stiffness matrices. We already presented this approach for Cook's membrane in [13].

7.1 Hierarchical structure of the problem

The basis functions for the conforming high-order finite element space were introduced in Chapter 4. We use products of one-dimensional hierarchical polynomials to construct basis functions on three-dimensional hexahedral elements. The choice of hierarchical basis functions leads to a specific structure of the nonlinear problem in Core Concept 4.1, which is observable in the linear systems in Newton's method.

The basis functions are, according to Section 4.4, classified into:

1. Node basis functions.
2. Edge basis functions.
3. Face basis functions.
4. Internal basis functions.

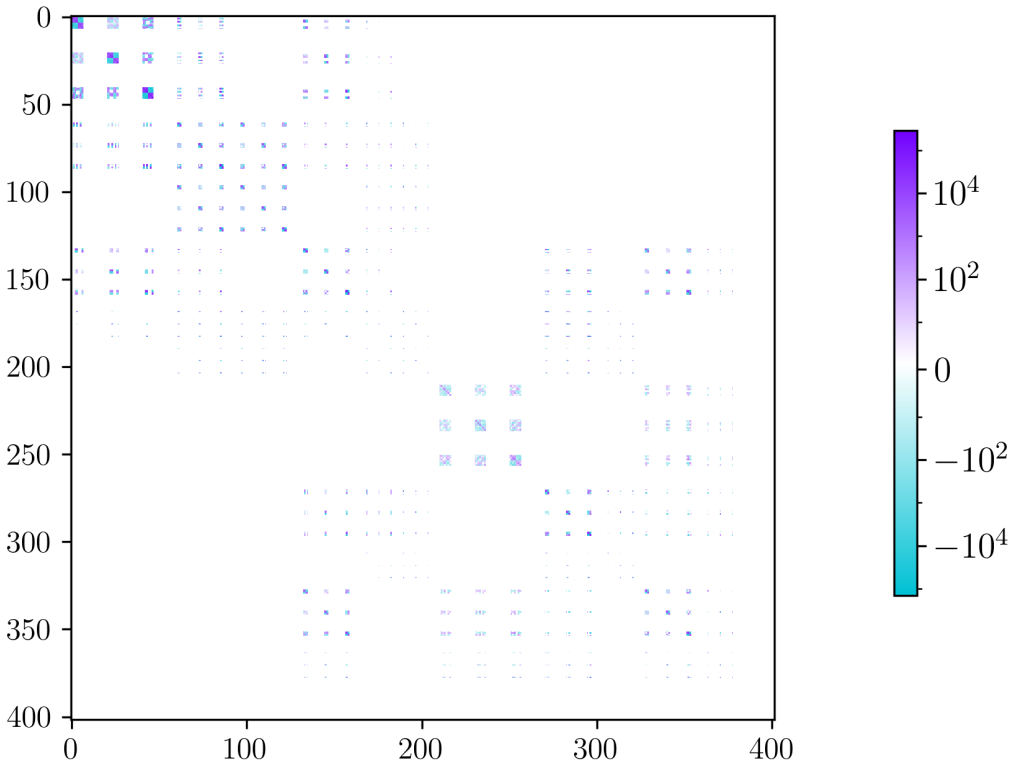


Figure 7.1: The node mode entries of the matrix $\mathbf{K}(\mathbf{u})$ for $p = 2$ after numerical integration and assembly, before application of the displacement boundary condition. The rows and columns of $\mathbf{K}(\mathbf{u})$ whose degree of freedom corresponds to an edge mode have been set to zero. The matrix entries are shown on a symmetric logarithmic scale with linear threshold $T_{lin} = 10$.

Each degree of freedom in the displacement vector $\mathbf{u}$ corresponds to one basis function. We can therefore use a classification into node, edge, face and internal degrees of freedom. Likewise, the matrix rows and columns are classified into the same categories, depending on the class of the corresponding degree of freedom. Understanding that the matrix rows and columns are related to one type of basis function reveals the hierarchical structure of the discretizations of different order. In Figure 7.1, a global stiffness matrix with maximum polynomial degree $p = 2$ is shown, where the rows and columns corresponding to the edge modes are set to zero. The remaining blocks, corresponding to the node modes, are the same as in Figure 6.6(a). This is due to the hierarchical structure of the basis functions as described in Section 4.3.

On Cook's membrane, the magnitude of the high-order degrees of freedom is

small. Figure 7.2 shows the displacement according to the degree of the basis functions. We computed the numerical solution for trunk space order $p = 5$ as in Figure 6.4. The model has 1752 degrees of freedom, see Table 6.1. We write the numerical solution as a linear combination of the basis functions as in Equation (4.3):

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{584} \mathbf{u}_i N_i(\mathbf{x}), \quad \mathbf{u}_i \in \mathbb{R}^3,$$

where $N_i : \Omega \to \mathbb{R}$ are the basis functions of the trunk space of order $p = 5$. Since the trunk space of order $p - 1$ is a subspace of the trunk space of order p, some summands are already included in trunk spaces of lower order. The basis functions are sorted in the following order:

$$\mathbf{u}(\mathbf{x}) = \underbrace{\underbrace{\underbrace{\underbrace{\sum_{i=1}^{44} \mathbf{u}_i N_i(\mathbf{x})}_{p=1} + \sum_{i=45}^{134} \mathbf{u}_i N_i(\mathbf{x})}_{p=2} + \sum_{i=135}^{224} \mathbf{u}_i N_i(\mathbf{x})}_{p=3} + \sum_{i=225}^{374} \mathbf{u}_i N_i(\mathbf{x})}_{p=4} + \sum_{i=375}^{584} \mathbf{u}_i N_i(\mathbf{x}).$$

We decompose the numerical solution according to the lowest order p of the trunk space in which the respective basis functions are included:

$$\mathbf{u}(\mathbf{x}) = \underbrace{\sum_{i=1}^{44} \mathbf{u}_i N_i(\mathbf{x})}_{=:\mathbf{u}(\mathbf{x})|_{p=1}} + \underbrace{\sum_{i=45}^{134} \mathbf{u}_i N_i(\mathbf{x})}_{=:\mathbf{u}(\mathbf{x})|_{p=2}} + \underbrace{\sum_{i=135}^{224} \mathbf{u}_i N_i(\mathbf{x})}_{=:\mathbf{u}(\mathbf{x})|_{p=3}} + \underbrace{\sum_{i=225}^{374} \mathbf{u}_i N_i(\mathbf{x})}_{=:\mathbf{u}(\mathbf{x})|_{p=4}} + \underbrace{\sum_{i=375}^{584} \mathbf{u}_i N_i(\mathbf{x})}_{=:\mathbf{u}(\mathbf{x})|_{p=5}}.$$

Figure 7.2(a) shows the linear combination of basis functions that are only included in the numerical solution for $p = 5$, but would not be included in a lower-order trunk space. The permitted basis functions depending on p are shown in Table 4.1. The basis functions in the linear combination $\mathbf{u}(\mathbf{x})|_{p=5}$ are the edge modes of total degree 7 and the face modes of total degree 5. Edge modes are linear in two directions, so these edge modes have degree 5 on one edge. The face modes are quadratic in two directions.

The next part of the linear combination is shown in Figure 7.2(b). The edge modes have maximum degree 4. We can see an edge mode on element 4 (according to the element numbers in Figure 6.2). In Figure 7.2(c), we see the edge modes of total degree 5 (maximum degree 3) which are clearly visible on all elements. Compare also to Figure 4.6 which shows an edge mode of total degree 5 on the reference element. Finally, the edge modes of total degree 4 (maximum degree 2) are shown in Figure 7.2(d). The quadratic polynomials can be observed on several elements, for example on the bottom edge of element 2. We note that the maximum magnitude is

largest in Figure 7.2(d). The magnitude is smallest for Figure 7.2(a), which shows the part of the solution that consists of the high-order modes. Compared to the full numerical solution with a maximum magnitude of the displacement (rounded to four digits)

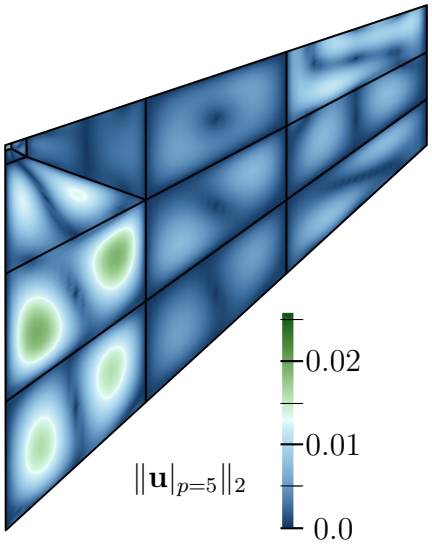
$$\max_{\mathbf{x}\in\Omega} \|\mathbf{u}(\mathbf{x})\|_2 = 27.4676,$$

the magnitude of the high-order corrections, especially of $\mathbf{u}|_{p=5}$, is small.

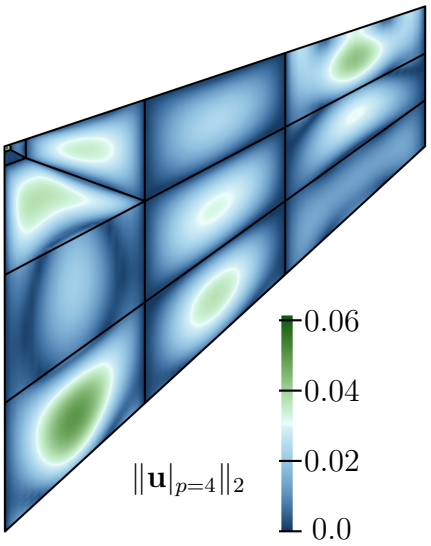
The hierarchical structure of the problem yields two possible approaches to reduce the computational effort.

1. A low-order matrix might be used as a preconditioner to solve the linear system in Newton's method.
2. For the convergence of Newton's method in each load step, it might be sufficient to iterate on low-order degrees of freedom in early load steps.

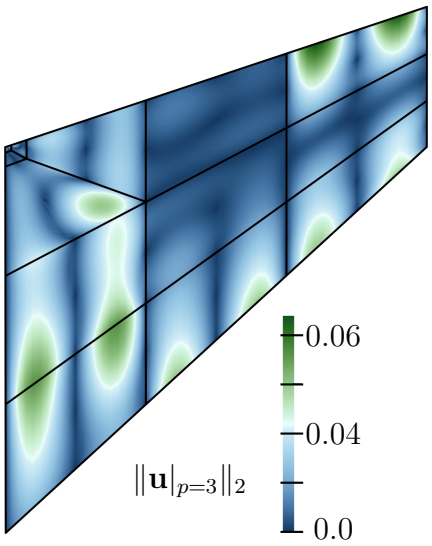
From these two approaches, only the second one has the potential to reduce computation time. This is because of the distribution of the computational work for Cook's membrane that was discussed in the previous chapter.



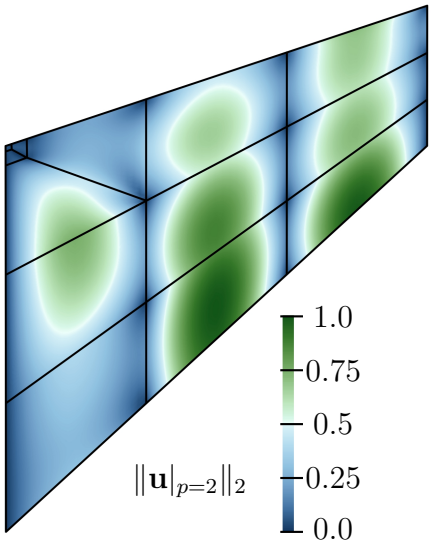
(a) Difference between the full model and the model with trunk space order $p = 4$.



(b) Difference between the models with $p = 4$ and $p = 3$.



(c) Difference between $p = 3$ and $p = 2$.



(d) Difference between $p = 2$ and $p = 1$.

Figure 7.2: From the numerical solution for $\rho = 50$ and trunk space order $p = 5$, we isolate the basis functions and degrees of freedom of different degree. The magnitude of $\mathbf{u}(\mathbf{x})|_p$ is shown in every point $\mathbf{x}$ of the computational domain Ω.

7.2 Generating start vectors on low-order models

We modify the load step Newton method according to the second approach from the previous section, see Algorithm 7.1. The new approach is called surrogate load step Newton method (SLSN). We use the trunk space as introduced in Definition 4.12. Compared to the load step Newton method, an additional array Π is required, defining the order $\Pi[i]$ of the trunk space in each load step i and thus the set of basis functions per load step. The SLSN approach is visualized in Figure 7.4.

Algorithm 7.1 Surrogate load step Newton method **(SLSN)**

Require: Maximum Newton steps $k_{\max}$, tol_{Newton}, order of the trunk space p
Require: Load step array Λ with length $i_{\max}$, $0 < \Lambda[i] < \Lambda[i{+}1]$ and $\Lambda[i_{\max}{-}1] = 1$,
Require: Trunk space order array Π with length $i_{\max}$ and $1 \leq \Pi[i] \leq \Pi[i+1]$

1: Set $\mathbf{u} = \mathbf{0}$.
2: Initialize an `adhocpp::initialBoundaryValueProblem IVP` with mesh $\mathcal{M}$ and trunk space basis functions of maximum order $\Pi[0]$.
3: Set traction on Γ_1 in direction $(0, 1, 0)^{\mathrm{T}}$ for the `IVP`.
4: Initialize zero displacement boundary condition on Γ_0 for the `IVP`.
5: Compute $\mathbf{E}_{\text{ext}}$ from the `IVP`.
6: **for** $i = 0, 1, \ldots, i_{\max}$ **do**
7: $\quad\lambda = \Lambda[i]$
8: $\quad$**for** $k = 0, 1, \ldots, k_{\max}$ **do**
9: $\quad\quad$Compute global stiffness matrix $\mathbf{K}(\mathbf{u})$ from the `IVP`.
10: $\quad\quad$Compute internal energy vector $\mathbf{E}_{\text{int}}(\mathbf{u})$ from the `IVP`.
11: $\quad\quad\mathbf{R} \leftarrow \lambda\mathbf{E}_{\text{ext}} - \mathbf{E}_{\text{int}}(\mathbf{u})$
12: $\quad\quad\Delta\mathbf{u} \leftarrow \text{linearSolver}(\mathbf{K}(\mathbf{u}), \mathbf{R})$.
13: $\quad\quad\mathbf{u} \leftarrow \mathbf{u} + \Delta\mathbf{u}$
14: $\quad\quad$**if** termination criterion $< \text{tol}_{Newton}$ **then** ▷ See Equations (6.5), (6.3)
15: $\quad\quad\quad$break
16: $\quad\quad$**end if**
17: $\quad$**end for**
18: $\quad$**if** $\Pi[i+1] \neq \Pi[i]$ **then**
19: $\quad\quad$Add basis functions of degree $\Pi[i+1]$ to the `IVP`.
20: $\quad\quad$Prolong $\mathbf{u}$ to the level $\Pi[i+1]$. ▷ See Figure 7.3
21: $\quad\quad$Compute $\mathbf{E}_{\text{ext}}$ from the `IVP`.
22: $\quad$**end if**
23: **end for**

We recapitulate the definition of the degrees of freedom, e.g., the entries of the vector $\mathbf{u} \in \mathbb{R}^{3n}$ in Algorithm 7.1. The vector represents an admissible displacement of Ω. The entries of $\mathbf{u}$ are used as coefficients in Equation (4.3) to obtain the current admissible deformation. Adding basis functions to the discretization (Line 19 in Algorithm 7.1) means that the linear combination in Equation (4.3) gets additional

summands, so the vector $\mathbf{u}$ needs to be longer to hold all degrees of freedom for the expanded model. This prolongation takes place in Line 20 of Algorithm 7.1.

The prolongation is implemented as follows. The components of the new vector $\mathbf{u}$ on level $\Pi[i+1]$ are initialized with the value of the degree of freedom of the respective basis function on the level $\Pi[i]$, if it was already included there. If the basis function is new, it is initialized with 0. See Figure 7.3 for a visualization.

The hierarchy of the trunk spaces is a requirement for the use of the surrogate load step Newton method. The trunk space basis is hierarchical by construction: From the set of possible basis functions constructed in Section 4.4, the ones with total degree not larger than p (for face and internal basis functions) or respective $p+2$ (for edge basis functions) span the trunk space of order p. Therefore, all basis functions for trunk spaces of smaller order are included. This makes the prolongation to a higher-order trunk space trivial, as it allows to add the basis functions of the respective level into the already existing set of basis functions. In particular, the components of the displacement vector can be transferred directly into the higher-order model as shown in Figure 7.3.

In SLSN, the initial load steps are performed on reduced-order models (surrogate models), which is visualized in Figure 7.4. As a trade-off between fast convergence and avoidance of locking effects polynomials of order $p = 2$ to 5 are used. For the surrogate models, using a relaxed tolerance in Line 14 of Algorithm 7.1 might be sufficient. In the next chapter, we will test SLSN with both strict and relaxed tolerance on two geometries. Our focus is on the impact on the number of Newton iterations and CPU time.

Connection to literature. According to the findings of Z. Yosibash and E. Priel on the active mechanical response of an artery wall, low-order models have the potential to reduce the computation time of elastic displacement simulations. In [36, Figure 4], the "p-prediction" algorithm is shown. First, all load steps are carried out on a low-order model to obtain a low-order numerical solution to the full-load problem. Then, the order of the polynomials is successively increased, with a run of Newton's method following each increase. For our model problem, we found that load steps were not necessary on the low-order model ($p = 1$). Increasing the load and order of the discretization simultaneously in SLSN has the additional advantage of providing more accurate numerical solutions to the load steps.

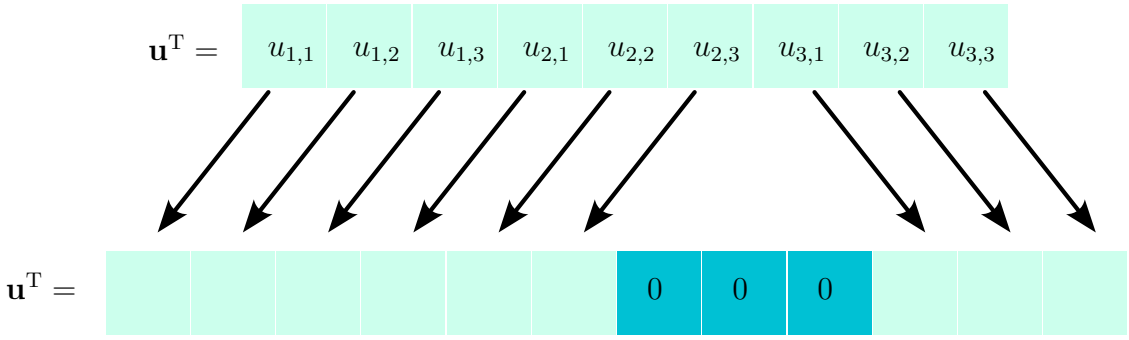


Figure 7.3: Illustration of the prolongation of the vector $\mathbf{u}$ for a small example with $n = 3$ basis functions on level $\Pi[i]$ and 4 basis functions on level $\Pi[i+1]$. On level $\Pi[i]$, the deformation is $\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{3} \sum_{j=1}^{3} u_{i,j} \mathbf{e}_j N_i(\mathbf{x})$, see Equation (4.3). The index j refers to the direction in space, i is the index of the basis function. The implementation in `AdhoC++` gives an ordering of the degrees of freedom $u_{i,j}$ into a global solution vector, shown above. The ordering is usually element-wise, starting with the low-order basis functions. On level $\Pi[i+1]$, a higher-order basis function is added to the basis, which results in three additional degrees of freedom. The additional degrees of freedom are initialized with 0.

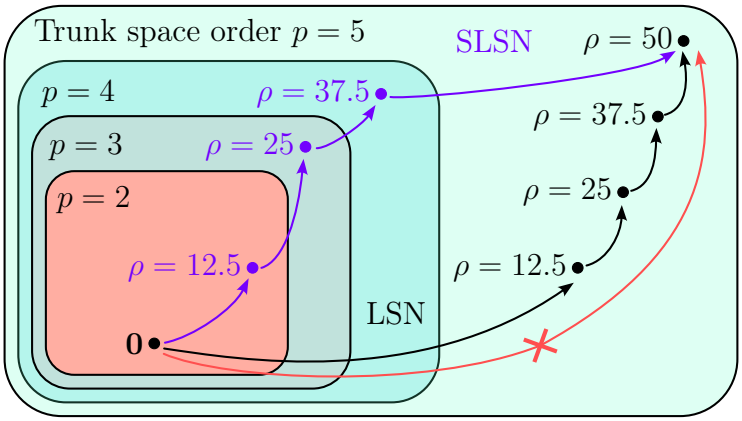


Figure 7.4: Schematic view of the LSN and SLSN methods. The numerical solutions to the discretizations of the Cook's membrane benchmark problem lie in trunk spaces with order $p = 2, \ldots, 5$. A traction boundary condition with different magnitudes ρ is used. The goal is a numerical solution to the discretization with $\rho = 50$ and $p = 5$, which leads to divergence of Newton's method when the zero vector $\mathbf{0}$ is used as a start vector. The numerical solutions in black illustrate the LSN method. Setting the magnitude of the traction boundary condition to 12.5 results in a convergent run of Newton's method. The numerical solution serves as a new start vector for the second run of Newton's method with $\rho = 25$. Over four load steps, the numerical solution for the full load is found. The SLSN method is shown in violet. The first run of Newton's method in the trunk space of order $p = 2$ converges to a numerical solution, which is prolonged and set as a starting vector for the next run.

Chapter 8

Numerical Tests

We test the surrogate load step Newton method (SLSN, Algorithm 7.1) on two displacement problems: Cook's membrane was introduced as a benchmark problem in Chapter 6. The purpose of the numerical tests on this geometry is to validate that SLSN converges to a numerical solution. We also test the influence of the Newton tolerance on intermediate load steps and compare the computation time against the traditional load step Newton method (LSN).

In the second part of the chapter, an elastic foam pore under displacement boundary conditions is investigated. Computation times for SLSN and LSN are compared for different target orders of the discretization. We expect high-order discretizations to profit more from the surrogate model approach than low-order discretizations.

8.1 Cook's membrane

We compare displacement simulations of high-order discretizations ($p = 5$) of Cook's membrane. The applied shear load is $\rho = 50$ for the geometry with 13 elements in Figure 6.2 and $\rho = 30$ for the refined geometry with 104 elements. The applied loads were chosen such that LSN requires two to four equidistant load steps for convergence. All tests for Cook's membrane run on an Intel Xeon processor (E5-2665). The linear systems are solved using the Pardiso solver (Intel Math Kernel Library Version 2022.0.2 for Linux).

Coarse mesh. Figure 8.1 depicts the convergence of LSN for the discretization with 13 elements (see the mesh in Figure 6.2). We see that the termination criterion is met after 5 Newton iterations on each load step. Relaxing the Newton tolerance on the intermediate load steps $1, 2$ and 3 leads to convergence of Newton's method in all load steps and a reduced computation time compared to the standard LSN. Table 8.1 shows the relative computation times for LSN with relaxed tolerance. The best result is achieved by LSN with a relaxed tolerance of $tol_{Newton} = 10^{-1}$ with 7

Table 8.1: Iterations and relative computation time of LSN for Cook's membrane (discretization with 13 elements, four load steps) with different Newton tolerances on intermediate load steps.

	LSN	**LSN + relaxed tolerance**		
tolerance intermediate load steps	10^{-8}	10^{-3}	10^{-2}	10^{-1}
total iterations intermediate load steps	15	12	11	7
tolerance final load step	10^{-8}	10^{-8}	10^{-8}	10^{-8}
iterations final load step	5	5	5	5
computation time (vs. LSN)	100%	86.65%	82.7%	66.13%

Newton iterations on the intermediate load steps (vs. 15 for standard LSN). This results in a computation time of about 66% compared to standard LSN.

In Figure 8.2, the convergence of the proposed SLSN method is shown. The first load steps are carried out on low-order FE models and the numerical solutions are used as starting vectors for the next load step. This results in convergence of Newton's method on the original problem with the full load (load step 4). We see that relaxing the tolerance on intermediate load steps reduces the number of Newton iterations when compared to the standard LSN method in Figure 8.1.

In Table 8.2, we compare computation times and number of iterations for SLSN with different choices for the relaxed tolerance. We see that the number of intermediate iterations increases with a stricter tolerance (10^{-3} and 10^{-8}). The total computation time is less than 55% of benchmark computation time (LSN) in all considered cases.

A combination of both strategies yields the best results. SLSN with a Newton tolerance of 10^{-1} needs about 40% of the computation time of the standard LSN method. The computational savings are due to two factors: The intermediate load steps are set up and solved with a relaxed tolerance and are of smaller size. While in the benchmark method (LSN with strict tolerance), four nonlinear problems with $1,752$ degrees of freedom are solved, for the surrogate load step Newton method, the number of degrees of freedom is substantially lower on intermediate load steps (see also Table 6.1). Only the nonlinear problem in the last load step is of full size and solved to the strict tolerance 10^{-8}.

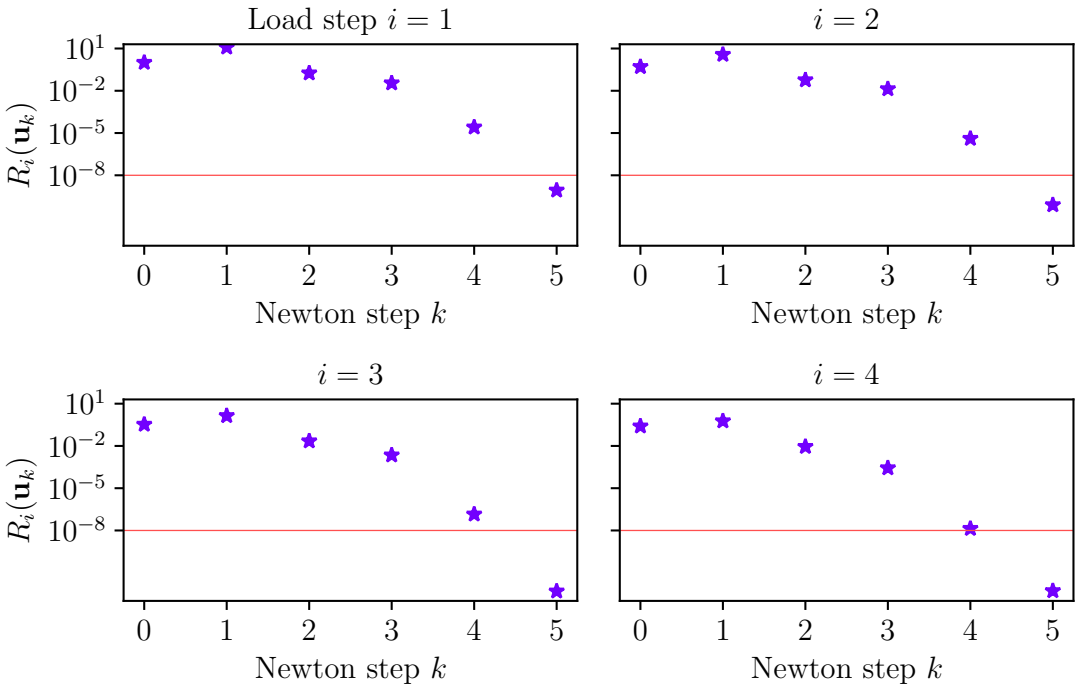


Figure 8.1: **LSN.** Convergence of the load step Newton method for Cook's membrane discretized with 13 finite elements and a high-order trunk space discretization of order 5. The normalized norm of the residual $R_i := \|\mathbf{R}\|_2 / \|\frac{i}{4}\mathbf{E}_{ext}\|_2$ (Equation (6.5)) with $tol_{Newton} = 10^{-8}$ is used as a termination criterion.

Table 8.2: Iterations and relative computation time of SLSN for Cook's membrane (discretization with 13 elements, four load steps) with different combinations of SLSN and relaxed tolerance on intermediate load steps.

	LSN	**SLSN**	**SLSN + relaxed tol.**		
tolerance intermediate load steps	10^{-8}	10^{-8}	10^{-3}	10^{-2}	10^{-1}
total iterations intermed. load steps	15	15	12	11	7
tolerance final load step	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
iterations final load step	5	5	5	5	5
computation time (vs. LSN)	100%	52.08%	47.58%	45.66%	39.71%

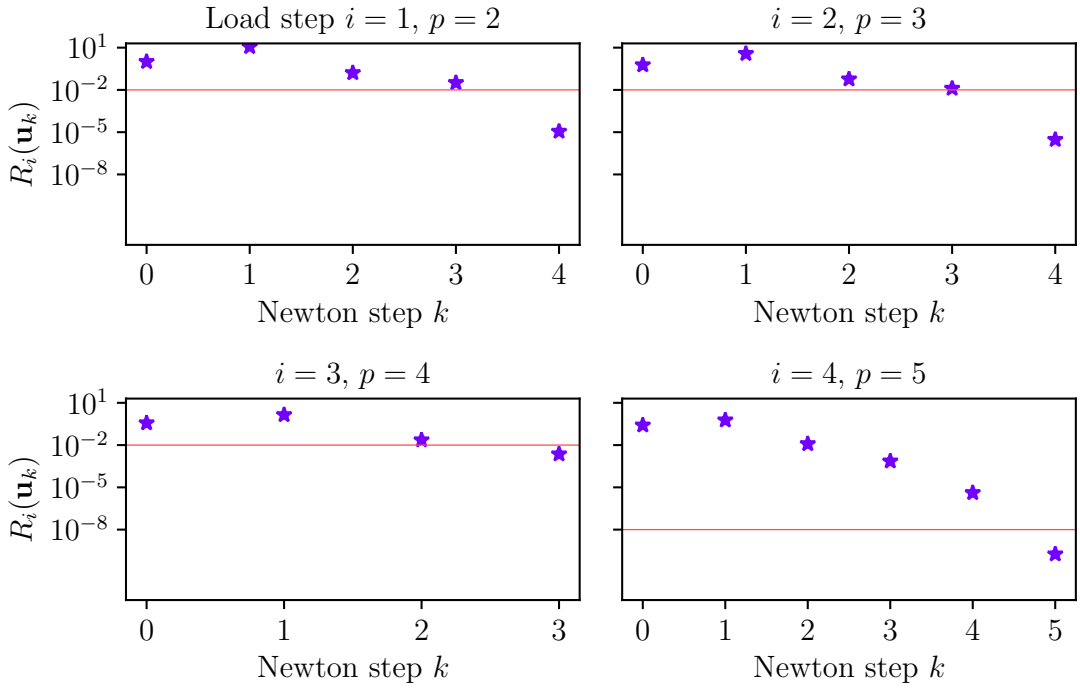


Figure 8.2: **SLSN.** Convergence of the surrogate load step Newton method for Cook's membrane, 13 elements, with a final trunk space order of $p = 5$. Surrogate models are used for all intermediate load steps as indicated in the titles of the subfigures. The tolerance for Newton's method is relaxed to 10^{-2} in intermediate load steps.

Fine mesh. Refining the geometry to 104 elements increases the number of degrees of freedom. This allows to test LSN and SLSN on a computationally more expensive problem. The fine mesh is depicted in Figure 6.10 and Table 8.3 shows the number of degrees of freedom.

Figure 8.3 shows the convergence of the traditional LSN method. In each load step, the termination criterion is met after 5 Newton steps. Relaxing the tolerance on the intermediate load steps decreases the number of intermediate Newton iterations, see Table 8.4. The savings in computation time are similar to the results on the coarse mesh (compare to Table 8.1).

The convergence of SLSN with relaxed tolerance can be seen in Figure 8.4. We observe that 10 Newton steps are needed for convergence on the last load step in comparison to 5 Newton steps for the traditional load step approach. Table 8.5 shows the computation times for all considered intermediate Newton tolerances.

The surrogate model increases the number of Newton iterations in all cases ($10-14$ vs. 5) and surprisingly even leads to divergence when used without any relaxation of the Newton tolerance. Here, the numerical solution from the previous load step computed with a strict tolerance seems to result in a poor starting vector for the last load step. This could be due to locking effects on the low-order models, i.e., unphysical small deformations that occur in conforming low-order finite elements, or the load step size.

In all cases with convergence on the last load step, we observe that SLSN decreases the computation time, but this decrease is not as significant as on the coarse mesh. This is related to an increased number of Newton iterations on the final load step. The best result on the fine mesh is achieved by SLSN with a relaxed Newton tolerance of 10^{-2}.

On both the coarse and the fine discretization of Cook's membrane, SLSN combined with a relaxed Newton tolerance yields the best results. On the coarse mesh, we need 40% of benchmark computation time and on the fine mesh we need 60%. The optimal choice for the intermediate Newton tolerance was 10^{-1} (coarse mesh) or 10^{-2} (fine mesh).

Table 8.3: Degrees of freedom for the fine discretization of Cook's membrane.

order of the trunk space	$p =$ **2**	**3**	**4**	**5**
degrees of freedom (104 elements)	2,523	4,284	7,359	11,748

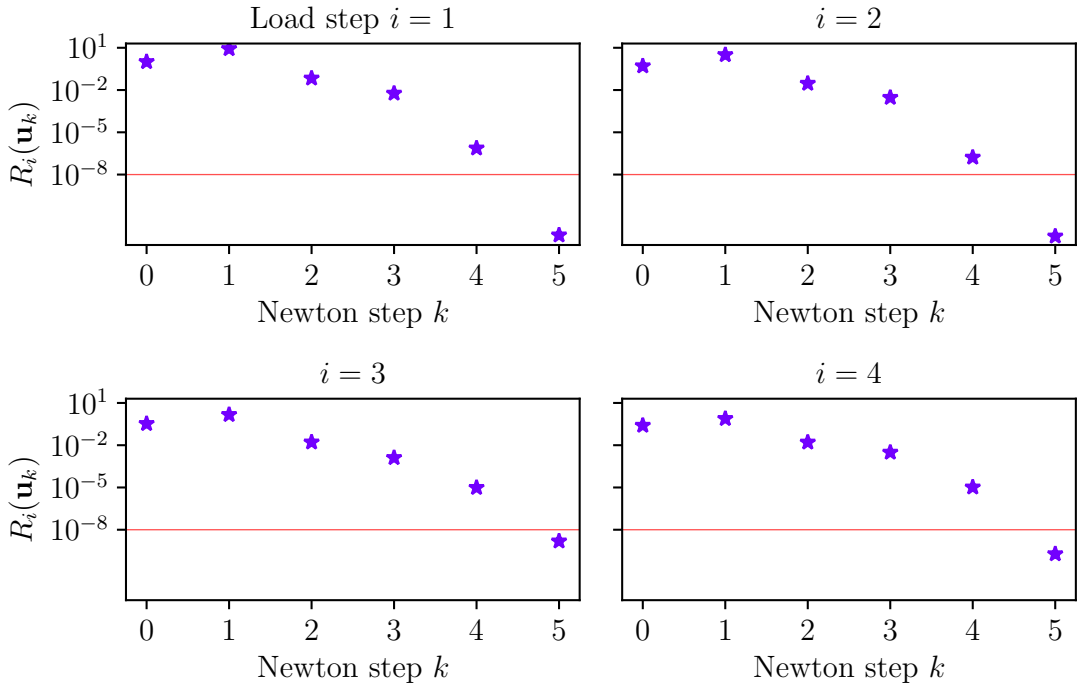


Figure 8.3: **LSN.** Traditional load step Newton method for Cook's membrane discretized with 104 finite elements and trunk space order $p = 5$. The magnitude of the traction boundary condition is $\rho = 30$. The load is increased in four load steps.

Table 8.4: Iterations and relative computation time of load step methods for Cook's membrane (discretization with 104 elements, four load steps).

	LSN	**LSN + relaxed tolerance**		
tolerance intermediate load steps	10^{-8}	10^{-3}	10^{-2}	10^{-1}
total iterations intermediate load steps	15	12	9	6
tolerance final load step	10^{-8}	10^{-8}	10^{-8}	10^{-8}
iterations final load step	5	5	5	6
computation time (vs. LSN)	100%	86.5%	75.33%	67.12%

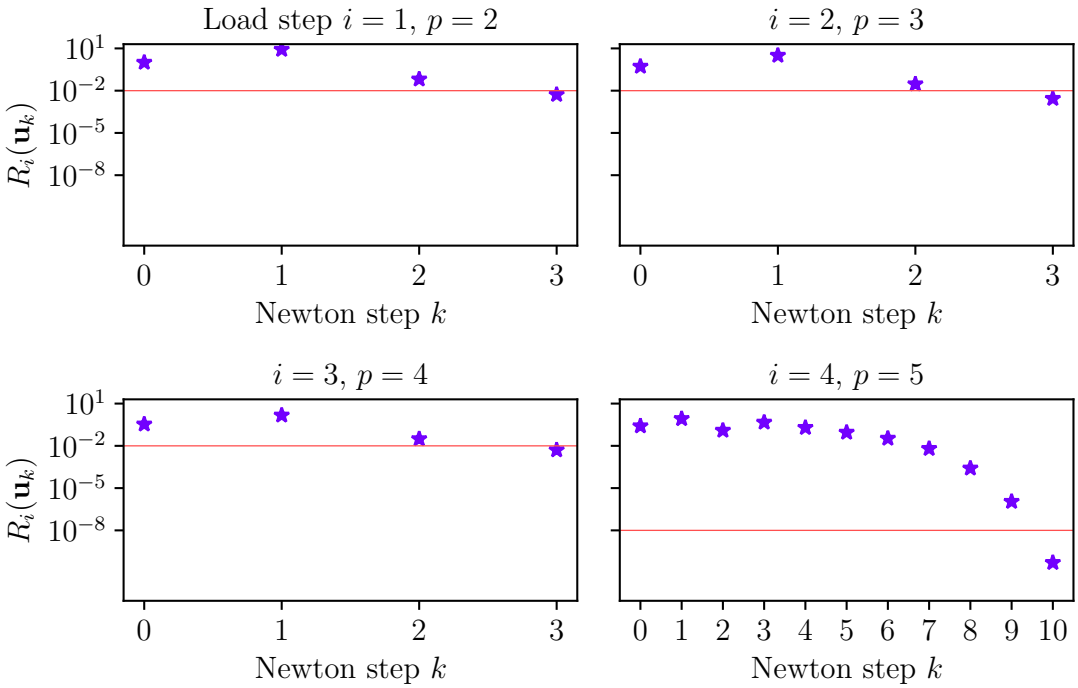


Figure 8.4: **SLSN.** The tolerance for Newton's method is relaxed to 10^{-2} in intermediate load steps.

Table 8.5: SLSN for Cook's membrane, discretization with 104 elements, four load steps.

	LSN	**SLSN**	**SLSN + relaxed tol**		
tolerance intermediate load steps	10^{-8}	10^{-8}	10^{-3}	10^{-2}	10^{-1}
total iterations intermediate load steps	15	16	12	9	6
tolerance final load step	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
iterations final load step	5	div.	14	10	13
computation time (vs. LSN)	100%		83.03%	62.93%	71.39%

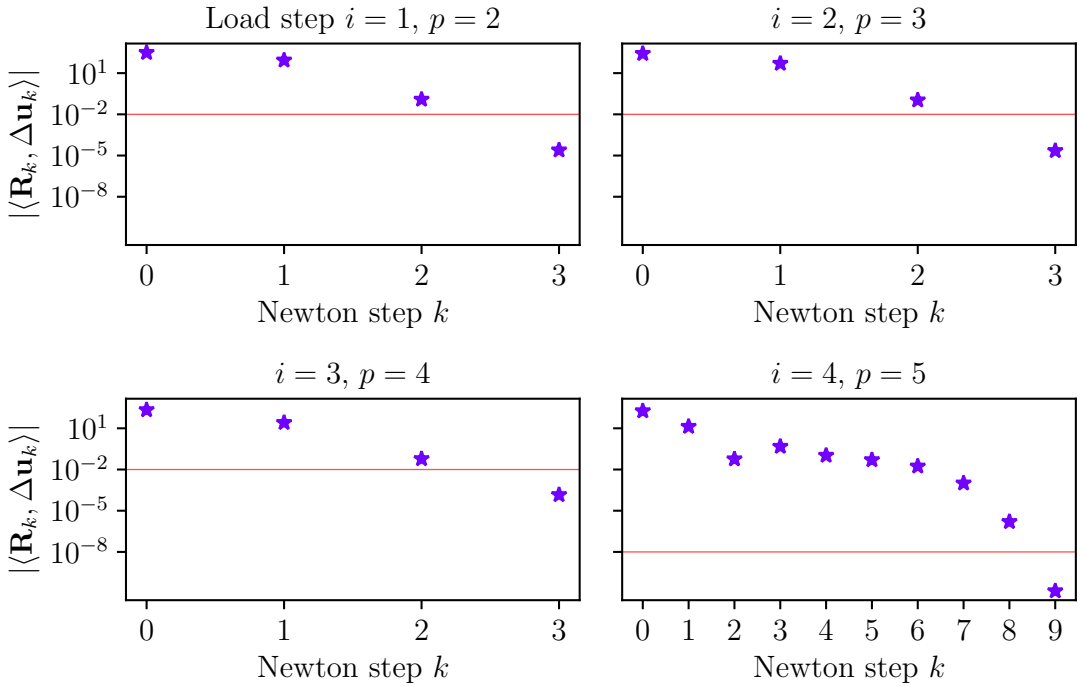


Figure 8.5: **SLSN with alternative termination criterion.** The scalar product of the residual and the solution increment is used as a termination criterion for Newton's method. The tolerance is relaxed to 10^{-2} in intermediate load steps.

Choice of the termination criterion. In Equations (6.2) and (6.3), two possible termination criteria for Newton's method were discussed. We used the normalized norm of the residual for the numerical tests on Cook's membrane. As the load is applied in increments, the external load in the denominator is scaled with the current load factor as shown in Equation (6.5). For the relaxed tolerance 10^{-2}, the termination criterion can be written as

$$\frac{\|\mathbf{R}\|_2}{\|\mathbf{E}_{ext}\|_2} < \lambda \cdot 10^{-2}.$$

Therefore, the termination criterion is stricter for smaller load factors. This contradicts the idea of applying relaxed tolerances in the initial load steps. We therefore compare with a numerical test that used the termination criterion in Equation (6.3), shown in Figure 8.5. We observe that the number of Newton iterations is comparable. For both termination criteria, three Newton iterations are required for each of the first three load steps. In the final load step, nine versus ten iterations are required.

8.2 Pore of an elastic foam

In this section, we test the surrogate load step Newton method on a complex domain. For our tests, we choose a single pore of a foam as shown in Figure 8.6. The test problem and discretization using the finite cell method (FCM) was proposed in [28, Section 3.2]. The elastic behavior of the foam pore under homogeneous displacement of the top plate is modeled using a hyperelastic material model with the strain energy function introduced in Core Concept 5.2.

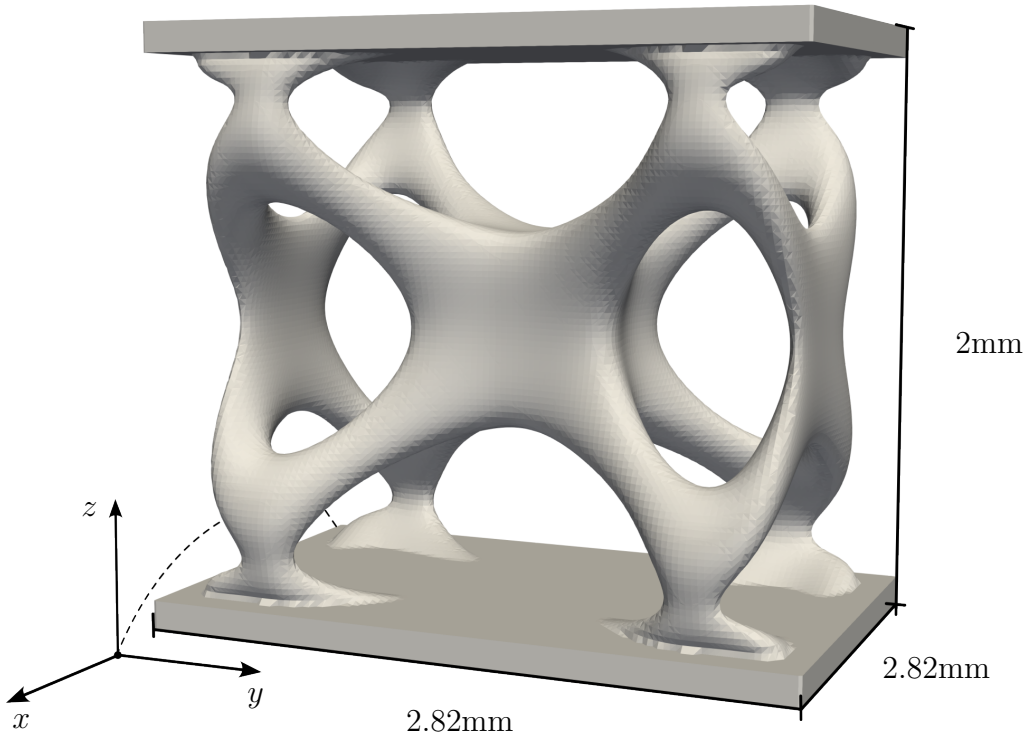
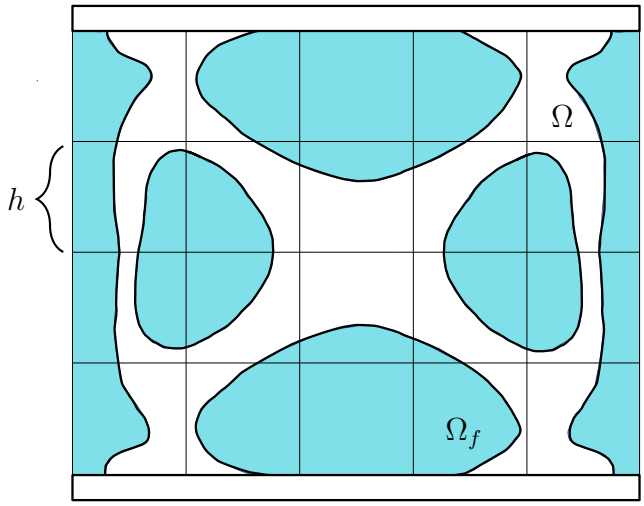


Figure 8.6: Geometry of a single pore of a foam. Two horizontal plates are connected by a symmetric structure. Displacement boundary conditions are defined on the top and bottom plate: The bottom plate is kept in place (zero displacement boundary condition), the top plate is displaced by 5mm in negative z-direction.

The FCM is usually applied to displacement problems where a mesh with an accurate representation of the domain is not available or difficult to obtain. The computational domain is extended to a simple domain, for example a cube, see Figure 8.7. High-order discretizations using the FCM are discussed in [12].

On the bottom plate $\Gamma_{\text{bottom}} = (0, 2.82) \times (0, 2.82) \times \{0\}$, a zero displacement boundary condition is applied. The displacement boundary condition $\mathbf{u}_0$ on the top plate $\Gamma_{\text{top}} = (0, 2.82) \times (0, 2.82) \times \{2\}$ (all measurements in mm) is

$$\mathbf{u}_0 = 5\text{mm} \begin{pmatrix} 0 & 0 & -1 \end{pmatrix}^{\mathrm{T}}.$$



$$\Omega_f \cup \overline{\Omega} = \Omega_e$$

Figure 8.7: The computational domain consists of the physical domain $\overline{\Omega}$ and the fictitious domain Ω_f. The computation of the integrals in the definition of $\mathbf{E}_{ext}, \mathbf{E}_{int}$ and the global stiffness matrix $\mathbf{K}$ has to be adapted. All integrals are evaluated using quadrature. If the quadrature node is inside the fictitious domain, the respective summand in the quadrature formula is multiplied with $\alpha = 10^{-5}$.

Table 8.6: Degrees of freedom for the foam pore depending on the order of the trunk space in the discretization with element size $h = 0.1$mm.

order of the trunk space	$p = \mathbf{1}$	**2**	**3**	**4**	**5**
degrees of freedom	12,939	46,011	79,083	139,887	228,423

We use two different element sizes, a coarse mesh with $h = 0.2$mm and a fine mesh with $h = 0.1$mm. The number of degrees of freedom for the fine mesh are shown in Table 8.6. Compared to the Cook's membrane benchmark problem, the number of DOF is significantly higher.

All tests in this section were run on the high performance computing cluster of TUHH. Two nodes, each equipped with two AMD Epyc 9354 CPUs, were used. Each CPU has 32 cores and a maximum frequency of 3.8GHz.

Applying LSN with 10 load steps to a finite cell trunk space discretization of order $p = 5$ with element size $h = 0.2$mm leads to convergence in every run of Newton's method. The resulting displacement is shown in Figure 8.8.

For Cook's membrane, we saw a significant increase of the set-up time for the global stiffness matrix with rising maximum polynomial degree of the discretization, which motivated the introduction of SLSN. The set-up time of $\mathbf{K}(\mathbf{u})$ for the foam

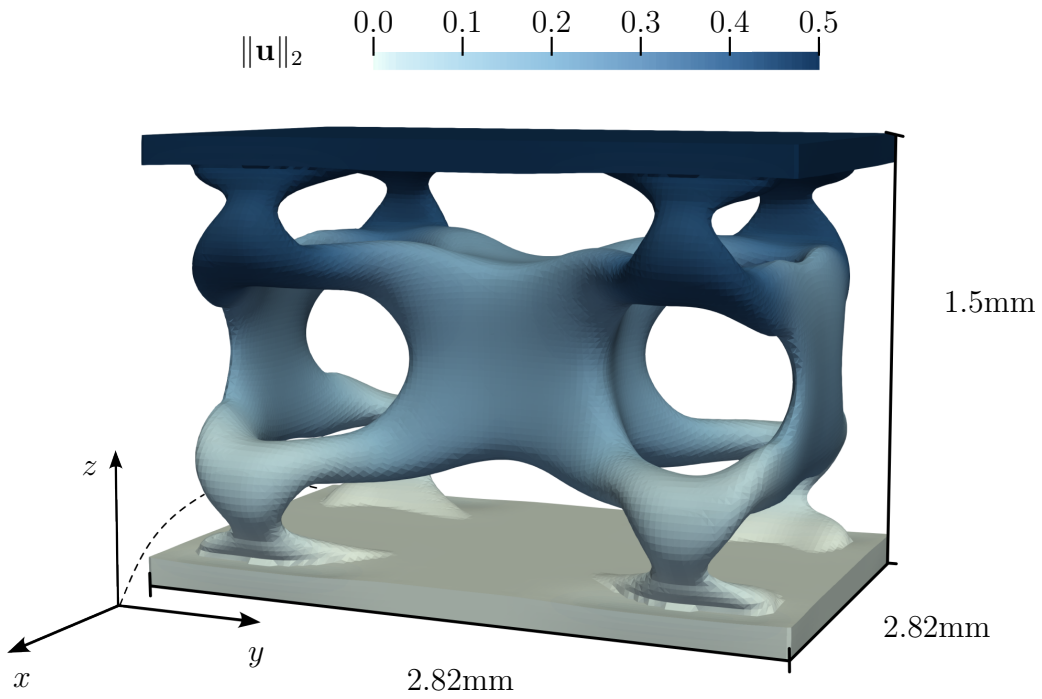


Figure 8.8: Applying a displacement boundary condition to the top plate causes the connecting structure to buckle outwards. A finite cell discretization using the trunk space with order $p = 5$ and element size $h = 0.2$ is used. All measurements are in mm. As the termination criterion for Newton's method, the scalar product of the Newton update and the residual is used, see Equation (6.3).

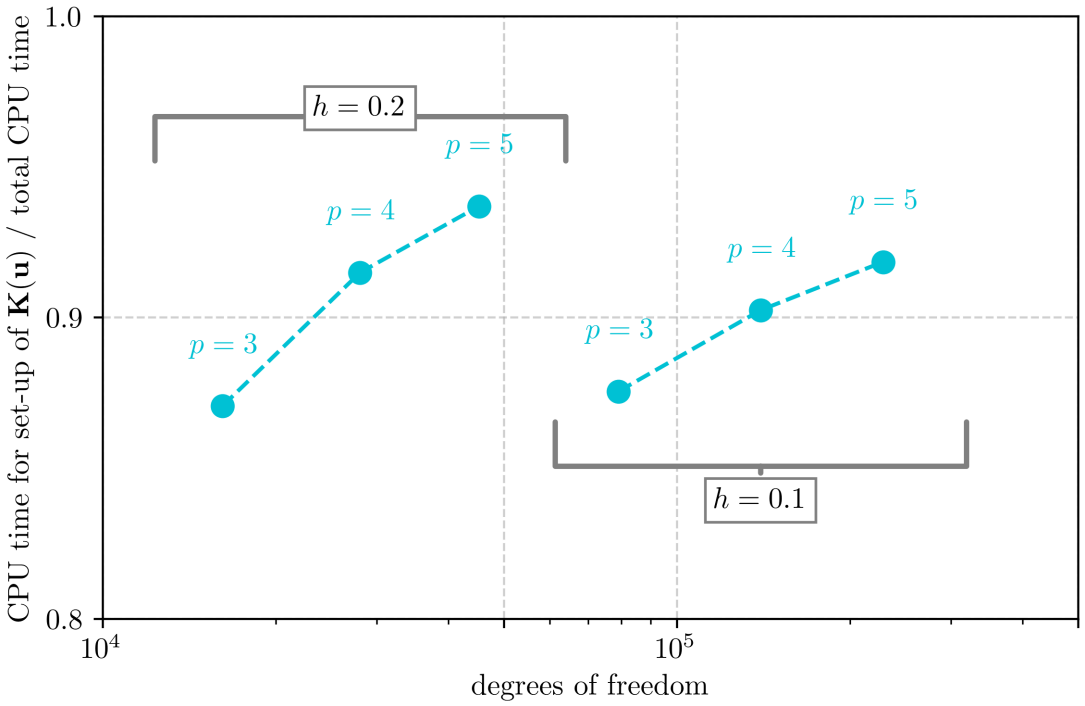


Figure 8.9: For the coarse (left) and fine (right) discretization of the foam pore, the CPU time for the set-up of the global stiffness matrix $\mathbf{K}(\mathbf{u})$ dominates the computation time. The higher the maximum polynomial degree of the discretization, the higher the share of the computation time spent on the set-up of $\mathbf{K}(\mathbf{u})$.

pore test problem is shown in Figure 8.9. We observe the same effect, both for a coarse discretization with element size $h = 0.2$mm and for a fine discretization with $h = 0.1$mm. For the coarse discretization and the high-order discretization with $p = 5$, the relative set-up time for the global stiffness matrix is the highest at about 94% of solution time.

In order to apply the surrogate load step Newton method (see Algorithm 7.1) to the foam pore problem, the traction boundary condition in Line 3 is omitted since the problem is purely displacement-driven. Line 4 has to be adapted to apply the zero displacement to Γ_{bottom}. Additionally, the non-zero displacement boundary condition on the top plate is initialized for the `adhocpp::initialBoundaryValueProblem`. For the non-zero displacement boundary condition, the scaling of the vector $\mathbf{E}_{ext}$ leads to the displacement being applied gradually over the load steps.

We apply SLSN to discretizations with $h = 0.1, 0.2$ mm and final polynomial degree $p = 3, 4, 5$, see Figure 8.10. In Table 8.7, the choice of the trunk space order is shown. Despite possible locking effects on the model with $p = 1$ and $p = 2$, the low-order models were still included to allow for a comparison of SLSN at small ($p = 3$) and larger goal orders ($p = 5$). The total CPU time is shown in Figure 8.10.

Table 8.7: Maximum polynomial degree for the 10 load steps, depending on the final polynomial degree of the discretization.

load step	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10 (final)**
	1	1	1	1	2	2	2	3	3	$p = \mathbf{3}$
order of the trunk space	1	1	1	2	2	2	3	3	4	$p = \mathbf{4}$
	1	1	2	2	3	3	4	4	5	$p = \mathbf{5}$

As expected, the CPU time increases with the number of degrees of freedom. The high-order coarse discretization ($h = 0.2, p = 5$) takes more CPU time than the low-order fine discretization ($h = 0.1, p = 3$) with more degrees of freedom. As shown in Figure 8.9, the set-up of the global stiffness matrix is the reason for this.

Using SLSN with strict tolerance reduces the computation time for the higher-order cases ($p = 4, 5$, both element sizes). Table 8.8 shows the normalized CPU times. We see that SLSL with strict tolerance increases CPU time to 104% in the low-order coarse case ($p = 3, h = 0.2$) and moderately decreases CPU time in the low-order fine ($p = 3, h = 0.1$) case.

The moderate reduction for $p = 3$ is likely due to the small size difference between the final model and the intermediate models. For $p = 4$, the intermediate model orders (Table 8.7) result in a significant reduction in CPU time. We iterated on models of order 1 and 2 for 6 load steps and switched to high-order models only for the last 4 load steps. For $p = 5$, we switch to $p = 3$ already on the fifth load step.

For SLSN with strict tolerance, the reduction in CPU time is most significant for the high-order case ($p = 5$). This aligns with the expectation that high-order discretizations profit most from SLSN, since they have a large size difference between initial and target trunk space order.

Relaxing the tolerance on intermediate load steps leads to a reduction in computation time for all test cases. For the low-order models ($p = 3$), the reduction is moderate (to $70 - 90\%$). For $p = 4$, the reduction of computation time is most significant, to $57\% - 70\%$ of the original CPU time. For $p = 5$ and $h = 0.2$, relaxing the tolerance increases the CPU time moderately compared to SLSN with strict tolerance. For $p = 5, h = 0.1$, the CPU time is almost the same as for SLSN with strict tolerance.

In conclusion, SLSN has the potential to decrease computation time especially for a high-order discretization. On the other hand, the method is sensitive to changes in the intermediate Newton tolerance.

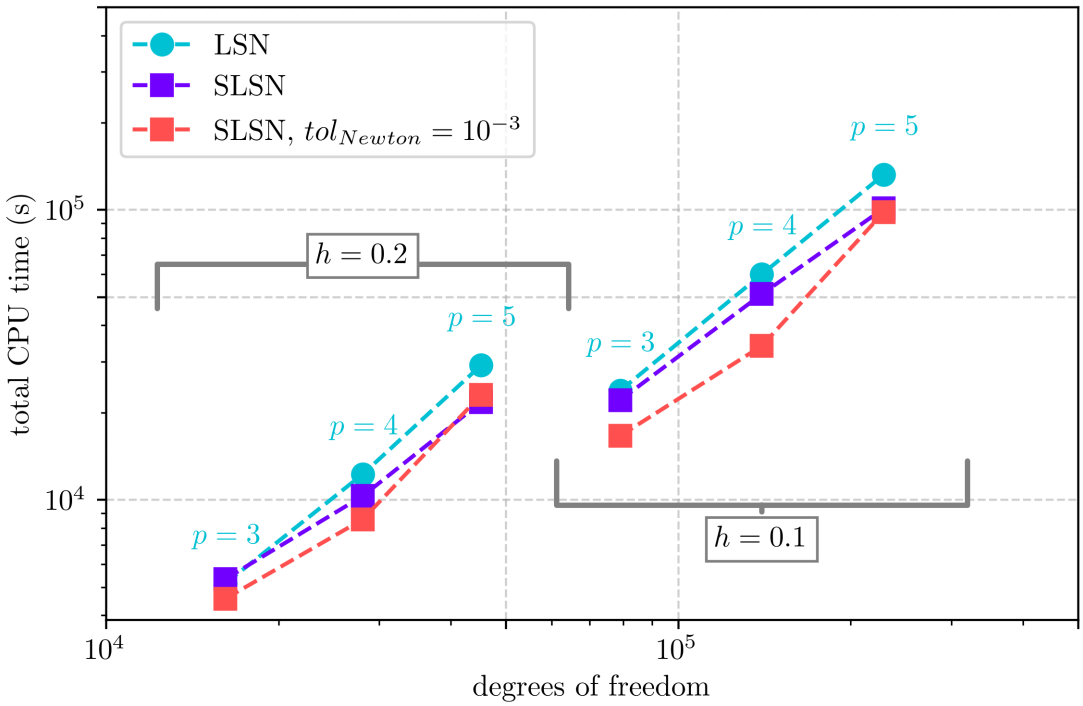


Figure 8.10: Total CPU time of displacement simulations for the foam pore. The load step Newton method (LSN) and surrogate load step Newton method (SLSN) are applied to solve different discretizations of the elastic foam pore. The number of load steps is 10. Simulations are run in parallel (using MPI and OpenMP) with 2 nodes, 4 threads each. The mesh width h is given in mm.

Table 8.8: Relative CPU time of SLSN compared to LSN on the foam pore geometry with element size h.

order of final trunk space		$p = 3$	$p = 4$	$p = 5$
SLSN	$h = 0.2$	1.04	0.84	0.75
SLSN, intermed. tol.=10^{-3}	$h = 0.2$	0.89	0.7	0.79
SLSN	$h = 0.1$	0.93	0.85	0.76
SLSN, intermed. tol.=10^{-3}	$h = 0.1$	0.7	0.57	0.74

Chapter 9

Summary and Outlook

Results. The numerical solution of displacement problems benefits from a high-order discretization. We support this claim in Section 6.1 with a literature review based on the Cook's membrane benchmark problem.

For nonlinear problems, the high-order discretization increases the number of Newton iterations and may cause Newton's method to diverge, as shown in Figure 6.3. Convergence is achieved by applying the load in increments using the load step Newton method (LSN).

Nonlinear high-order finite element problems require significant computational effort for numerical integration and global stiffness matrix assembly. Numerical tests on Cook's membrane (Section 6.2) and a foam pore geometry (Section 8.2) demonstrate this. This is due to the larger bandwidth and number of non-zero entries in the global stiffness matrix, as discussed in Section 6.4.

The new surrogate load step Newton method (SLSN) exploits the traditional incremental approach. If a hierarchical basis is used in the Galerkin ansatz, low-order models can be used in early load steps. As the external load increases, the high-order basis functions and their respective degrees of freedom are added, resulting in low computational effort in the initial load steps. SLSN reduces computation time to about 50% (for Cook's membrane, 75% for the foam pore geometry). Combining SLSN with a relaxed tolerance on early load steps has the potential to reduce computation time even further (40% for Cook's membrane and 57% for the foam pore, see Tables 8.2 and 8.8).

However, surrogate models can lead to an increased number of Newton iterations or even divergence. For Cook's membrane, we found that the convergence of Newton's method on the final load step depends on the choice of the Newton tolerance on intermediate load steps. Reduction in computation time also depends on the size difference between the final model and the surrogate model, as we demonstrated on the foam pore geometry.

Outlook. SLSN was studied using the finite element and finite cell methods on two geometries using a hyperelastic material model. To test the applicability and performance of the method, it must be applied to other geometries and material models. In particular, the method should be tested with almost incompressible elastic materials and elastoplastic materials.

In our numerical tests, the Newton tolerance, load step size, and order of the discretization are predetermined. These parameters can be optimized. To exploit the full potential of the surrogate model, automatic choices of model order, load and relaxed tolerance should be investigated. Error estimation techniques should be explored to determine when to switch to the next model order.

Recently, mixed-precision algorithms for matrix computations have been introduced, see for example [25]. This development has been driven by advancements in commercial mixed-precision hardware, used primarily for deep neural networks. Shifting some of the computation in SLSN to lower-precision hardware could potentially reduce the computational effort required, but the impact on Newton convergence remains to be investigated.

SLSN is motivated by the significant amount of the computation time spent on numerical integration and assembly in p-FEM. However, there are other approaches to address this bottleneck. In the `C++` finite element library MFEM [1], global assembly of the stiffness matrix is avoided in the high-order case. Instead, partial assembly and sum factorization are used, and this class of solvers is called „matrix-free".

Finite element and finite cell computations can be parallelized. With the increasing availability of high-performance hardware, it may be feasible to rely on the massive parallel computation of the global stiffness matrix. Matrix-free high-order computations on graphics processors are studied, for example, in [21].

Combining parallelization, matrix-free methods, and the proposed surrogate models could lead to high-performance solvers for nonlinear finite element and finite cell problems.

References

[1] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini. "MFEM: A modular finite element methods library". In: *Comput. Math. Appl.* 81 (2021), pp. 42–74. DOI: 10.1016/j.camwa.2020.06.009.

[2] N. A. Barnafi, L. F. Pavarino, and S. Scacchi. "Parallel inexact Newton-Krylov and quasi-Newton solvers for nonlinear elasticity". In: *Comput. Methods Appl. Mech. Engrg.* 400 (2022). Article number 115557. DOI: 10.1016/j.cma.2022.115557.

[3] H. R. Bayat, J. Krämer, S. Reese, C. Wieners, B. Wohlmuth, and L. Wunderlich. "Hybrid Discretizations in Solid Mechanics for Non–linear and Non-Smooth Problems". In: *Non–standard Discretisation Methods in Solid Mechanics*. Vol. 98. Lect. Notes Appl. Comput. Mech. Springer, Cham, 2022, pp. 1–35. DOI: 10.1007/978-3-030-92672-4_1.

[4] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. 2nd. John Wiley & Sons, Ltd., Chichester, 2014.

[5] J. Bonet, A. J. Gil, and R. D. Wood. *Nonlinear Solid Mechanics for Finite Element Analysis: Statics*. Cambridge University Press, Cambridge, 2016.

[6] R. de Borst, M. A. Crisfield, J. J. C. Remmers, and C. V. Verhoosel. *Non–linear Finite Element Analysis of Solids and Structures*. John Wiley & Sons, Ltd., Chichester, 2012. DOI: 10.1002/9781118375938.

[7] D. Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. 5th. Springer-Lehrbuch Masterclass. Berlin, Heidelberg: Springer Spektrum, 2013.

[8] P. G. Ciarlet. *Mathematical Elasticity. Volume I. Three-dimensional Elasticity*. Vol. 84. Classics in Applied Mathematics. Reprint of the 1988 edition. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2022.

[9] M. A. Crisfield. "A Fast Incremental/Iterative Solution Procedure That Handles "Snap-Through"". In: *Computers & Structures* 13.1 (June 1, 1981), pp. 55–62. DOI: 10.1016/0045-7949(81)90108-5.

[10] J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Vol. 16. Classics in Applied Mathematics. Corrected reprint of the 1983 original. Society for Industrial and Applied Mathematics, Philadelphia, 1996. DOI: 10.1137/1.9781611971200.

[11] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. 2nd. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2017. DOI: 10.1093/acprof:oso/9780198508380.001.0001.

[12] A. Düster, E. Rank, and B. Szabó. "The p-Version of the Finite Element and Finite Cell Methods". In: *Encyclopedia of Computational Mechanics Second Edition*. John Wiley & Sons, Ltd., Chichester, 2017, pp. 1–35. DOI: 10.1002/9781119176817.ecm2003g.

[13] L. Fesefeldt, S. Le Borne, A. Düster, and L. Radtke. "Using surrogate models to accelerate load step methods for nonlinear finite element problems in hyperelasticity". In: *Proceedings in Applied Mathematics and Mechanics* 24.3 (2024). DOI: 10.1002/pamm.202400081.

[14] M. J. Gander and F. Kwok. *Numerical Analysis of Partial Differential Equations Using Maple and MATLAB*. Vol. 12. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, Philadelphia, 2018. DOI: 10.1137/1.9781611975314.ch1.

[15] N. J. Higham. *Handbook of Writing for the Mathematical Sciences*. 3rd. Society for Industrial and Applied Mathematics, Philadelphia, 2020. DOI: 10.1137/1.9781611976106.

[16] R. Hiptmair. *Numerical Methods for Partial Differential Equations*. Seminar für Angewandte Mathematik, ETH Zürich. URL: https://www.sam.math.ethz.ch/~grsam/NUMPDEFL/NUMPDE.pdf. Accessed: 9. October 2025.

[17] L. Hug, M. Potten, G. Stockinger, K. Thuro, and S. Kollmannsberger. "A three-field phase-field model for mixed-mode fracture in rock based on experimental determination of the mode II fracture toughness". In: *Eng. Comput.* 38.6 (July 2022), pp. 5563–5581. DOI: 10.1007/s00366-022-01684-9.

[18] J. Jaśkowiec and N. Sukumar. "Penalty-free discontinuous Galerkin method". In: *Internat. J. Numer. Methods Engrg.* 125.12 (2024). Article number e7472. DOI: 10.1002/nme.7472. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.7472.

[19] J. N. Jomo. "Towards scalable finite cell computations on massively parallel systems". en. PhD thesis. Technische Universität München, 2021, p. 181.

[20] G. Királyfalvi and B. A. Szabó. "Quasi-regional mapping for the p-version of the finite element method". In: *Finite Elem. Anal. Des.* 27.1 (1997), pp. 85–97. DOI: 10.1016/S0168-874X(97)00006-1.

[21] M. Kronbichler and K. Ljungkvist. "Multigrid for Matrix-Free High-Order Finite Element Computations on Graphics Processors". In: *ACM Trans. Parallel Comput.* 6.1 (Mar. 2019). Article number 2. DOI: 10.1145/3322813.

[22] F. S. Liguori, A. Madeo, S. Marfia, G. Garcea, and E. Sacco. "A stabilization-free hybrid virtual element formulation for the accurate analysis of 2D elasto-plastic problems". In: *Comput. Methods Appl. Mech. Engrg.* 431 (2024). Article number 117281. DOI: 10.1016/j.cma.2024.117281.

[23] M. Neunteufel, A. S. Pechstein, and J. Schöberl. "Three-field mixed finite element methods for nonlinear elasticity". In: *Comput. Methods Appl. Mech. Engrg.* 382 (2021). Article number 113857. DOI: 10.1016/j.cma.2021.113857.

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd. Springer Series in Operation Research and Financial Engineering. New York: Springer, New York, 2006. 664 pp. DOI: 10.1007/978-0-387-40065-5.

[25] E. Oktay and E. Carson. "Multistage mixed precision iterative refinement". In: *Numer. Linear Algebra Appl.* 29.4 (2022). Article number e2434. DOI: 10.1002/nla.2434.

[26] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. 3rd. New York: Cambridge University Press, New York, 2007.

[27] S. Reese, H. Bayat, and S. Wulfinghoff. "On an equivalence between a discontinuous Galerkin method and reduced integration with hourglass stabilization for finite elasticity". In: *Comput. Methods Appl. Mech. Engrg.* 325 (2017), pp. 175–197. DOI: 10.1016/j.cma.2017.07.005.

[28] R. Sartorti and A. Düster. "Remeshing in the finite cell method for different types of geometry descriptions". In: *Proceedings in Applied Mathematics and Mechanics* 24.4 (2024). DOI: 10.1002/pamm.202400046.

[29] J. Schröder, T. Wick, S. Reese, P. Wriggers, R. Müller, S. Kollmannsberger, M. Kästner, A. Schwarz, M. Igelbüscher, N. Viebahn, H. R. Bayat, S. Wulfinghoff, K. Mang, E. Rank, T. Bog, D. D'Angella, M. Elhaddad, P. Hennig, A. Düster, W. Garhuom, S. Hubrich, M. Walloth, W. Wollner, C. Kuhn, and T. Heister. "A Selection of Benchmark Problems in Solid Mechanics and Applied Mathematics". In: *Arch. Comput. Methods Eng.* 28.2 (Sept. 2020), pp. 713–751. DOI: 10.1007/s11831-020-09477-3.

[30] B. Szabó and I. Babuška. *Introduction to Finite Element Analysis*. John Wiley & Sons, Ltd., Chichester, 2011. DOI: 10.1002/9781119993834.

[31] M. Wagner. *Lineare und nichtlineare FEM*. Springer Fachmedien, Wiesbaden, 2017. DOI: 10.1007/978-3-658-17866-6.

[32] Z.-Y. Wang, Y.-F. Jin, Z.-Y. Yin, and Y.-Z. Wang. "Overcoming volumetric locking in stable node-based smoothed particle finite element method with cubic bubble function and selective integration". In: *Internat. J. Numer. Methods Engrg.* 123.24 (2022), pp. 6148–6169. DOI: 10.1002/nme.7107.

[33] P. Wriggers. "Continuum Mechanics, Nonlinear Finite Element Techniques and Computational Stability". In: *Progress in Computational Analysis of Inelastic Structures*. Springer, Vienna, 1993, pp. 245–287. DOI: 10.1007/978-3-7091-2626-4_5.

[34] P. Wriggers. *Nonlinear Finite Element Methods*. Springer-Verlag, Berlin, 2008.

[35] S. Wulfinghoff, H. R. Bayat, A. Alipour, and S. Reese. "A low-order locking-free hybrid discontinuous Galerkin element formulation for large deformations". In: *Comput. Methods Appl. Mech. Engrg.* 323 (2017), pp. 353–372. DOI: 10.1016/j.cma.2017.05.018.

[36] Z. Yosibash and E. Priel. "Artery Active Mechanical Response: High Order Finite Element Implementation and Investigation". In: *Comput. Methods Appl. Mech. Engrg.* 237–240 (Sept. 1, 2012), pp. 51–66. DOI: 10.1016/j.cma.2012.05.001.

[37] N. Zander, T. Bog, M. Elhaddad, R. Espinoza, H. Hu, A. Joly, C. Wu, P. Zerbe, A. Düster, S. Kollmannsberger, J. Parvizian, M. Ruess, D. Schillinger, and E. Rank. "FCMLab: A finite cell research toolbox for MATLAB". In: *Advances in Engineering Software* 74 (2014), pp. 49–63. DOI: 10.1016/j.advengsoft.2014.04.004.

Appendix A

Collection of Formulas

Integration by parts

Let $\Omega \in \mathbb{R}^n$ be compact and the boundary $\partial\Omega$ piecewise smooth. Assume there exists an outer normal vector field $\mathbf{n}$ with $\|\mathbf{n}\| = 1$. Let $\mathbf{v}$ be a continuously differentiable vector field defined on an open neighbourhood of Ω and φ a continuously differentiable scalar field on Ω. Then the relation

$$\int_\Omega \varphi \operatorname{div} \mathbf{v} \, \mathrm{d}V = \oint_{\partial\Omega} \varphi \mathbf{v} \cdot \mathbf{n} \, \mathrm{d}S - \int_\Omega \mathbf{v} \cdot \nabla\varphi \, \mathrm{d}V \tag{A.1}$$

holds.

Transformation theorem

Let $\Omega \subseteq \mathbb{R}^d$ be open and $\boldsymbol{\varphi} : \Omega \to \boldsymbol{\varphi}(\Omega) \subseteq \mathbb{R}^d$ bijective and continuously differentiable. Then the function $\mathbf{f}$ is integrable on $\boldsymbol{\varphi}(\Omega)$ if and only if $x \mapsto \mathbf{f}(\boldsymbol{\varphi}(x))|\det(J_{\boldsymbol{\varphi}}(x))|$ is integrable on Ω. In this case,

$$\int_{\boldsymbol{\varphi}(\Omega)} \mathbf{f}(\mathbf{y}) \, \mathrm{d}\mathbf{y} = \int_\Omega \mathbf{f}(\boldsymbol{\varphi}(\mathbf{x})) \; |\det(J_{\boldsymbol{\varphi}}(\mathbf{x}))| \, \mathrm{d}\mathbf{x}. \tag{A.2}$$

Multidimensional chain rule

Let $\mathbf{U} = \mathbf{f}(\mathbf{X})$ with $\mathbf{U} \in \mathbb{R}^{m\times n}$ and consider a function $\mathbf{g}(\mathbf{U})$. The derivative of $\mathbf{g}$ with respect to the components of $\mathbf{X}$ is

$$\frac{\partial \mathbf{g}(\mathbf{U})}{\partial x_{ij}} = \sum_{k=1}^{m} \sum_{l=1}^{n} \frac{\partial \mathbf{g}(\mathbf{U})}{\partial u_{kl}} \frac{\partial u_{kl}}{\partial x_{ij}}. \tag{A.3}$$

Appendix B

Enforcing Displacement Boundary Conditions

In Section 6.2, we mention that due to the structure of the `AdhocC++` code, the displacement boundary conditions are applied after the global stiffness matrix and the external and internal load vectors are computed. This is because the admissible basis functions are generated from the mesh and trunk space order first. Memory is then allocated for the respective degrees of freedom and the global stiffness matrix. When computing the global stiffness matrix, it is assumed that all the degrees of freedom in memory are "true" degrees of freedom. After computation and assembly, the values of some degrees of freedom have to be enforced on the linear system to ensure that the numerical solution fulfills the displacement boundary conditions.

We consider the initial linear system, which is computed using the basis functions that are generated by `AdhoC++`, including basis functions those that are non-zero on the part of the boundary where displacement boundary conditions are applied. Let $\mathbf{K}(\mathbf{u}) \in \mathbb{R}^{n\times n}$ be the global stiffness matrix, $\mathbf{E}_{ext} \in \mathbb{R}^n$ the external load vector and $\mathbf{E}_{int} \in \mathbb{R}^n$ the internal load vector. The Newton update is a solution of the linear system $\mathbf{K}(\mathbf{u})\Delta\mathbf{u} = \mathbf{E}_{ext} - \mathbf{E}_{int} =: \mathbf{R}$, or component-wise

$$\mathbf{K}\Delta\mathbf{u} = \begin{pmatrix} k_{1,1} & \cdots & k_{1,i} & \cdots & k_{1,j} & \cdots & k_{1,n} \\ \vdots & \ddots & & & & & \vdots \\ k_{i,1} & & k_{i,i} & & & & k_{i,n} \\ \vdots & & & \ddots & & & \vdots \\ k_{j,1} & & & & k_{j,j} & & k_{j,n} \\ \vdots & & & & & \ddots & \vdots \\ k_{n,1} & \cdots & k_{n,i} & \cdots & k_{n,j} & \cdots & k_{n,n} \end{pmatrix} \begin{pmatrix} \Delta u_1 \\ \vdots \\ \Delta u_i \\ \vdots \\ \Delta u_j \\ \vdots \\ \Delta u_n \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ r_i \\ \vdots \\ r_j \\ \vdots \\ r_n \end{pmatrix}.$$

We first enforce a homogenous displacement boundary condition on Δu_j by replacing

the equation in row j with the trivial equation $\Delta u_j = 0$, resulting in the linear system

$$\begin{pmatrix} k_{1,1} & \cdots & k_{1,i} & \cdots & k_{1,j} & \cdots & k_{1,n} \\ \vdots & \ddots & & & & & \vdots \\ k_{i,1} & & k_{i,i} & & & & k_{i,n} \\ \vdots & & & \ddots & & & \vdots \\ 0 & \cdots & & 0 & 1 & 0 & 0 \\ \vdots & & & & & \ddots & \vdots \\ k_{n,1} & \cdots & k_{n,i} & \cdots & k_{n,j} & \cdots & k_{n,n} \end{pmatrix} \begin{pmatrix} \Delta u_1 \\ \vdots \\ \Delta u_i \\ \vdots \\ \Delta u_j \\ \vdots \\ \Delta u_n \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ r_i \\ \vdots \\ 0 \\ \vdots \\ r_n \end{pmatrix}.$$

The entries in column j outside of the diagonal can be omitted, since they are multiplied with 0. For symmetric $\mathbf{K}$, this ensures that the symmetry is kept in the modified matrix:

$$\begin{pmatrix} k_{1,1} & \cdots & k_{1,i} & \cdots & 0 & \cdots & k_{1,n} \\ \vdots & \ddots & & & \vdots & & \vdots \\ k_{i,1} & & k_{i,i} & & & & k_{i,n} \\ \vdots & & & \ddots & 0 & & \vdots \\ 0 & \cdots & & 0 & 1 & 0 & 0 \\ \vdots & & & & 0 & \ddots & \vdots \\ k_{n,1} & \cdots & k_{n,i} & \cdots & 0 & \cdots & k_{n,n} \end{pmatrix} \begin{pmatrix} \Delta u_1 \\ \vdots \\ \Delta u_i \\ \vdots \\ \Delta u_j \\ \vdots \\ \Delta u_n \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ r_i \\ \vdots \\ 0 \\ \vdots \\ r_n \end{pmatrix}.$$

For a non-homogenous displacement boundary condition, e.g. $\Delta u_i = 2$, row i is replaced:

$$\begin{pmatrix} k_{1,1} & \cdots & k_{1,i} & \cdots & 0 & \cdots & k_{1,n} \\ \vdots & \ddots & & & \vdots & & \vdots \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & & & \ddots & 0 & & \vdots \\ 0 & \cdots & & 0 & 1 & 0 & 0 \\ \vdots & & & & 0 & \ddots & \vdots \\ k_{n,1} & \cdots & k_{n,i} & \cdots & 0 & \cdots & k_{n,n} \end{pmatrix} \begin{pmatrix} \Delta u_1 \\ \vdots \\ \Delta u_i \\ \vdots \\ \Delta u_j \\ \vdots \\ \Delta u_n \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ 2 \\ \vdots \\ 0 \\ \vdots \\ r_n \end{pmatrix}.$$

To keep the symmetry of $\mathbf{K}$, the entries in column i are moved to the right-hand side of the equation:

$$\underbrace{\begin{pmatrix} k_{1,1} & \cdots & 0 & \cdots & 0 & \cdots & k_{1,n} \\ \vdots & \ddots & & & \vdots & & \vdots \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & & & \ddots & 0 & & \vdots \\ 0 & \cdots & & 0 & 1 & 0 & 0 \\ \vdots & & & & 0 & \ddots & \vdots \\ k_{n,1} & \cdots & 0 & \cdots & 0 & \cdots & k_{n,n} \end{pmatrix}}_{=:\widetilde{\mathbf{K}}} \underbrace{\begin{pmatrix} \Delta u_1 \\ \vdots \\ \Delta u_i \\ \vdots \\ \Delta u_j \\ \vdots \\ \Delta u_n \end{pmatrix}}_{=:\widetilde{\Delta \mathbf{u}}} = \underbrace{\begin{pmatrix} r_1 - 2k_{12} \\ \vdots \\ 2 \\ \vdots \\ 0 \\ \vdots \\ r_n - 2k_{n,i} \end{pmatrix}}_{=:\widetilde{\mathbf{R}}}$$

The rank of the resulting matrix $\widetilde{\mathbf{K}}$ is larger or equal to the rank of the initial matrix $\mathbf{K}$, since the new rows are linearly independent. The matrix $\widetilde{\mathbf{K}}$ is not guaranteed to be regular; The regularity of the system depends on the geometry and boundary conditions of the continuous problem. Assuming that the resulting linear system is regular, $\widetilde{\Delta\mathbf{u}}$ is also a solution to the initial linear system if all replaced rows of the initial matrix were linear combinations of the remaining rows.