JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Felix Scholz

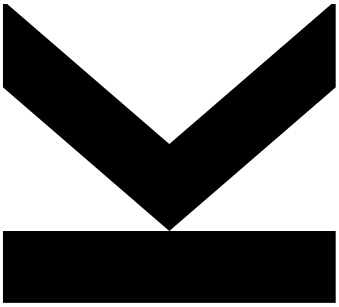
Submitted at
Institute of Applied Geometry

Supervisor and
First Examiner
Bert Jüttler

Second Examiner
Giancarlo Sangalli

June 2019

Efficient Matrix Assembly for Isogeometric Analysis



Doctoral Thesis
to obtain the academic degree of
Doktor der Naturwissenschaften
in the Doctoral Program
Naturwissenschaften

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Abstract

Isogeometric discretizations for the numerical solution of partial differential equations possess significant advantages compared to classical finite element methods. These include the higher smoothness of the obtained numerical solution (when using higher polynomial degree), the compatibility of the representation with models coming from computer-aided design (CAD) systems, as well as the reduction of the number of degrees of freedom required to reach a prescribed accuracy level. However, there still exist demanding challenges that need to be overcome in order to achieve the full integration of CAD and numerical simulation. Two of them, both related to the matrix assembly step for Galerkin discretizations, are the subject of this thesis.

System matrix assembly for isogeometric (i.e., spline-based) discretizations of partial differential equations is more challenging than for classical finite elements, due to the increased polynomial degrees and the larger (and hence more overlapping) supports of the basis functions. The global tensor-product structure of the discrete spaces employed in isogeometric analysis can be exploited to accelerate the computations, using sum factorization, precomputed look-up tables, and tensor decomposition.

In the first part of the thesis, we generalize the tensor decomposition approach by considering partial tensor decompositions. We show that the resulting new method preserves the global discretization error and that its computational complexity compares favorably to the existing approaches. Moreover, the numerical realization simplifies considerably since it relies on standard techniques from numerical linear algebra.

Next to the application of this new method for matrix assembly to elliptic partial differential equations in three-dimensional domains, another natural application are partial differential equations in four-dimensional space-time domains. To demonstrate this, we present a space-time isogeometric analysis scheme for the discretization of parabolic evolution equations with diffusion coefficients depending on the time and space variables. The problem is considered in a space-time

cylinder in $\mathbb{R}^4$ and is discretized using high-order spline spaces, making the matrix assembly a bottle neck in terms of computation time. We overcome this problem by decoupling the operator into space and time components using our partial tensor decomposition.

In the second part of the thesis, we consider the treatment of trimmed domains in isogeometric analysis, i.e., domains where only a part of the domain is active while the rest is discarded. The use of trimming greatly enhances the flexibility of the NURBS representations that are commonly employed in CAD. More precisely, we develop a specialized quadrature rule for trimmed domains.

We first consider trimmed two-dimensional domains. In our setting, the trimming curve is given implicitly by a real-valued function on the whole domain. We follow an error correction approach: In a first step, we obtain an adaptive subdivision of the domain in such a way that each cell falls in a predefined base case. We then extend the classical approach of linear approximation of the trimming curve by adding an error correction term based on a Taylor expansion of the blending between the linearized implicit trimming curve and the original one. This approach leads to an accurate method which improves the convergence of the quadrature error by one order compared to piecewise linear approximation of the trimming curve. It is at the same time efficient and easy to implement, since only the computation of one additional one-dimensional integral on each trimmed cell is required. The convergence is analyzed theoretically and numerical experiments confirm that the accuracy is improved without compromising the computational complexity.

Finally, we generalize our quadrature rule to the three-dimensional case, thus finding a quadrature rule for trimmed volumes. In this case, the linear approximation is performed using a constrained least squares approximation that preserves the topology of the trimming surface. As in the two-dimensional case, we introduce an error correction term that makes it possible to achieve a cubic convergence rate, which is one order higher than the rate obtained by using a linear approximation only. We also present numerical experiments that demonstrate the method's potential for applications in isogeometric analysis.

Zusammenfassung

Isogeometrische Diskretisierungen für die numerische Lösung von partiellen Differentialgleichungen haben diverse Vorteile gegenüber klassischen Finite-Elemente-Methoden. Unter ihnen sind eine höhere Glattheit der diskreten Lösung, die Kompatibilität der Geometriedarstellungen mit Modellen aus Computer Aided Design (CAD)-Systemen und eine kleinere Anzahl von Freiheitsgraden, die für dieselbe Approximationsgenauigkeit benötigt werden. Es gibt allerdings noch einige Herausforderungen, die bewältigt werden müssen, um CAD und numerische Simulation vollständig miteinander zu vereinen. Zwei dieser Herausforderungen, die beide mit der Matrixassemblierung zusammenhängen, sind Thema dieser Dissertation.

Aufgrund der höheren Grade der eingesetzten Polynome und der größeren Träger der Basisfunktionen ist die Matrixassemblierung für isogeometrische (d.h. auf Splines basierende) Diskretisierungen herausfordernder als die für klassische Finite-Elemente-Methoden. Die globale Tensorproduktstruktur der in der isogeometrischen Analysis eingesetzten diskreten Räume kann genutzt werden, um die Berechnungen zu beschleunigen. Bisher wurde dies mithilfe von Summenfaktorisierung, zuvor berechneter Nachschlagetabellen und Tensorzerlegung getan. Im ersten Abschnitt dieser Dissertation verallgemeinern wir den Ansatz der Tensorzerlegung, indem wir eine partielle Tensorzerlegung betrachten. Wir zeigen, das die daraus resultierende neue Methode den globalen Diskretisierungsfehler erhält und dass die Rechenkomplexität im Vergleich zu den existierenden Ansätzen geringer ist. Zudem ist die Implementierung der neuen Methode erheblich einfacher, da sie nur auf Standardtechniken aus der numerischen linearen Algebra basiert.

Eine weitere natürliche Anwendung dieser neuen Methode ist – neben der Anwendung auf elliptische partielle Differentialgleichungen in dreidimensionalen Gebieten – die Anwendung auf partielle Differentialgleichungen in vierdimensionalen Raumzeitgebieten. Um dies zu demonstrieren, stellen wir eine isogeometrische Raumzeit-Methode für die Diskretisierung von parabolischen Evolutionsgleichungen, deren Diffusionskoeffizienten von den Raum- und Zeitvariablen abhängen,

vor. Die Gleichung wird in einem Raumzeitzylinder im $\mathbb{R}^4$ betrachtet und in Splineräumen von hohem Grad diskretisiert. Hierdurch wird die Matrixassemblierung zu einem Engpass für die Rechenzeit. Wir lösen dieses Problem, indem wir den Operator mithilfe unserer Methode der partiellen Tensorzerlegung in Raum- und Zeitkomponenten zerlegen.

Im zweiten Teil der Dissertation betrachten wir die Verarbeitung von getrimmten Gebieten in der isogeometrischen Analysis, d.h. von Gebieten, von denen nur ein bestimmter Teil aktiv ist. Die Verwendung von Trimming erhöht die Flexibilität der im CAD üblicherweise eingesetzten NURBS-Darstellungen erheblich. Wir entwickeln nun eine spezialisierte Quadraturregel für getrimmte Gebiete.

Wir beginnen die Darstellung unserer Methode mit der Betrachtung von getrimmten zweidimensionalen Gebieten. Dabei nehmen wir an, dass die Trimmkurve implizit durch eine reellwertige Funktion auf dem gesamten Gebiet gegeben ist. Wir verfolgen einen Fehlerkorrekturansatz: Im ersten Schritt der Methode zerteilen wir das Gebiet in Quadraturzellen, die alle in einem vorher definiertem Basisfall liegen. Wir erweitern dann den klassischen Ansatz einer stückweisen linearen Approximation der Trimmkurve, indem wir einen Fehlerkorrekturterm addieren. Dieser basiert auf einer Taylorentwicklung der Interpolation zwischen der ursprünglichen Trimmkurve und ihrer linearen Approximation. Mithilfe dieses Ansatzes erzeugen wir eine akkurate Methode, deren Konvergenzordnung um eins höher ist als die Konvergenzordnung, wenn nur die lineare Approximation verwendet wird. Sie ist zudem effizient und einfach zu implementieren, da nur ein zusätzliches eindimensionales Integral auf jeder getrimmten Quadraturzelle berechnet werden muss. Wir untersuchen die Konvergenzordnung analytisch und zeigen numerische Experimente, die bestätigen, dass die Genauigkeit der Quadratur verbessert wird, ohne die Rechenkomplexität zu beeinträchtigen.

Zuletzt verallgemeinern wir die Quadraturregel auf getrimmte dreidimensionale Gebiete. In diesem Fall finden wir die lineare Approximation der Trimmfläche durch eine Least-Squares-Approximation mit Nebenbedingungen, die die Topologie der Trimmfläche erhält. Wie auch im 2D-Fall addieren wir einen Fehlerkorrekturterm, der eine kubische Konvergenzordnung bedingt. Sie ist somit um eins höher ist als bei einer reinen linearen Approximation. Unsere numerischen Experimente belegen die Eignung der Methode für Anwendungen in der isogeometrischen Analysis.

Acknowledgments

First of all, I would like to thank Prof. Bert Jüttler for giving me the opportunity to work with him, for his guidance and supervision, for the many interesting and informative discussions, and for the thorough revisions while writing our papers. I also owe my thanks to Prof. Giancarlo Sangalli for reviewing this thesis.

I thank Dr. Angelos Mantzaflaris for the patient and instructive explanations about implementation techniques in C++, for introducing me to new topics, and for the fruitful collaboration. I also collaborated with Dr. Ioannis Toulopoulos, who I want to thank for his great ideas for our joint paper.

My research was funded by the Austrian Science Fund (FWF) through project NFN S11708, the support is gratefully acknowledged.

My heartfelt thanks go to my colleagues at the Institute of Applied Geometry and at the Radon Institute for Computational and Applied Mathematics (RICAM), who have made my stay in Linz a great experience.

Finally, I am very grateful to my family and to my friends for the constant support that has enabled me to reach this milestone.

Contents

1. Introduction

Isogeometric analysis (IgA) [14, 19] is a method for the numerical approximation of solutions to partial differential equations (PDE) that aims to unify the fields of computer-aided design (CAD) and numerical simulation. To this end the NURBS representations [74] that are commonly used in CAD are used directly for representing the geometry in the simulation step without performing an additional approximation of the geometry. According to the isoparametric paradigm, the same discrete space is employed for the discretization of the solution to the PDE as for the representation of the geometry.

Isogeometric discretizations possess significant advantages over classical finite element discretizations. Besides the compatibility of the representation with models generated by CAD systems, these include the higher smoothness of the obtained numerical solution (when using a higher polynomial degree) as well as the reduction of the number of degrees of freedom required to reach a prescribed accuracy level.

A large amount of research on IgA has been published in the last years. However, there still exist challenges that need to be overcome in order to apply isogeometric methods efficiently. Two of them, both related to the matrix assembly step in isogeometric Galerkin methods, are the subject of this thesis. It consists of two parts: In Part A, we develop an efficient method for the computation of the system matrix in isogeometric Galerkin methods on tensor-product patches. The method is based on tensor decomposition and is applied to three-dimensional elliptic PDEs and four-dimensional time-dependent parabolic PDEs. In Part B, we consider the application of IgA to trimmed surfaces and volumes, which often occur in CAD models, and develop an efficient method for the numerical quadrature on this type of domains. The content of this thesis has been published in [66], [82], [83] and [84].

Part A.

Assembly on tensor-product domains by partial tensor decomposition

2. Contribution and outline

The advantages of isogeometric analysis come at the price of increased computation costs per degree of freedom, and this effect becomes even more pronounced as the dimension increases [18]. In fact, the complexity depends exponentially on the dimension, this is sometimes called the *curse of dimensionality*. The computational efficiency challenges manifest themselves both in the matrix assembly, which is in the focus of this work, and in the solution of the resulting linear systems, see e.g. [28, 91] for recent advances in that direction.

There are several lines of work which aim at improving the computational efficiency of isogeometric matrix computations. First, quadrature rules with a reduced (or even minimal) number of nodes have been studied [5, 8, 9, 41]. These rules are defined for univariate spline spaces and require less quadrature points compared to standard Gaussian quadrature, taking into account the higher regularity of the space. Specialized reduced quadrature rules have also emerged [1, 35, 36]. The generalization to multivariate integrals is achieved by a tensor-product approach.

Second, the collocation approach to discretization of PDEs has been extensively explored in isogeometric analysis [4, 80]. Intuitively, collocation may be thought as one-point quadrature, since approximately one evaluation per element is required for highly-smooth B-spline discretizations. In [32] the efficiency of the collocation approach is increased by identifying collocation points with optimal approximation properties. However, the method acts on the strong form of the PDE at hand, thereby sacrificing some of the benefits of the Galerkin method.

Third, the built-in tensor-product structure of isogeometric discretizations has been exploited to improve efficiency. The evaluation of mass and stiffness matrices was addressed in [61, 62], based on small look-up tables for univariate B-spline integral. These are used in conjunction with an interpolation approach that transforms the integrands into spline functions. Building on these ideas, singular

value decomposition (SVD) is used in [63] for the decomposition of bivariate integrals into univariate integrals. This has been generalized to higher dimensions in [64], using a similar decomposition approach for tensors.

A different idea to exploit the tensor-product structure has been explored in [17]: the authors use quadrature rules which are exact for all integrals that involve a fixed basis function. This approach is combined with sum factorization [2]. The tensor-product structure has also been employed for constructing fast preconditioners for the linear systems that arise from isogeometric discretization [79].

Tensor methods are quite effective when dealing with high-dimensional operators [6, 50, 60] and have found applications in several fields where structured data are present [34, 49]. Recently, tensor decomposition has been proposed as a general approach to reduce the complexity of simulations in isogeometric analysis [64]. While in 2D this approach requires solely standard linear algebra tools, its extension to higher dimensions requires advanced decomposition algorithms for tensors, such as higher-order singular value decomposition (HOSVD) and alternating least squares (ALS). Those methods are not supported by all standard scientific computing libraries. Moreover, they require non-linear optimization and their properties are not yet fully understood. In contrast to this, singular value decomposition of matrices – which is the main tool required for the 2D case – is well established and highly optimized implementations are provided by linear algebra packages.

In Part A of this thesis, we present an new approach that improves on the tensor decomposition method from [64]. We employ SVD in order to decouple isogeometric discretizations partially, while maintaining a quasi-optimal complexity for the task of matrix formation. This demonstrates that for efficient 3D isogeometric matrix computations, sophisticated tools such as HOSVD/ALS can be replaced by standard SVD. More precisely, we split the domain variables into two groups and perform SVD with respect to this partition. Consequently, our initial 3D matrix computation problem is replaced by a set of univariate and bivariate integration problems, which are solved independently, and provide a Kronecker product representation of the original matrix. We show that the overall asymptotic complexity remains quasi-optimal, and the obtained rank values for this decomposition are in the worst case equal to the case of a full 3D decomposition and can be much lower in practice. The description of the new partial tensor decomposition method and its application

to matrix assembly in 3D isogeometric Galerkin method is the subject of Chapter 4, this work has also been published in [83].

In the last chapter of Part A, we present a second natural application of our partial tensor decomposition method: parabolic initial-boundary value problems, which are frequently used to describe time evolution phenomena in physics and medicine. The standard approach for these problems is to discretize separately in space and in time.

As an alternative, space-time methods have been considered that approximate the solution over a global space-time domain. This means that the time variable is regarded as an extra spatial variable and therefore the computational domain has an additional dimension compared to the spatial domain. For high dimensions, the tensor-train format has been used for a global space-time finite element approximation [22, 23].

In the context of isogeometric analysis, a space-time method has been proposed in [58] for the pure heat equation with constant coefficients. Here, the space-time cylinder is parameterized as a NURBS volume and high-order spline spaces are used for the discretization. The variational formulation is based on [55, 56].

The presence of an additional variable increases the computational complexity even further, especially for the matrix assembly step. For the assembly of the Galerkin matrices using standard methods it is necessary to perform numerical quadrature in $\mathbb{R}^{d+1}$, which becomes prohibitively slow, even for a moderate number of degrees of freedom. To treat this problem efficiently, we apply our partial tensor decomposition to the isogeometric space-time method for the heat equation from [58] as well as to a more general parabolic equation with a diffusion coefficient that depends on both, space and time. We arrive at a decomposition of the system matrix into Kronecker products of space and time components, making it possible to perform the numerical quadrature independently.

This application has been published in [66], where also the theoretical analysis from [58] for the case of constant diffusion is generalized to the parabolic equation with varying diffusion coefficient and error estimates for the isogeometric discretization are presented.

3. Isogeometric discretization and model problem

In this chapter, we present a class of elliptic partial differential equations and their discretization in isogeometric analysis. They will serve as a model problem for our methods for the efficient assembly of the system matrix in isogeometric analysis, in particular for the partial tensor decomposition method that is the subject of Chapter 4.

3.1. Isogeometric formulation of elliptic equations

Let L be a differential operator

$$Lu = -\nabla \cdot (A(x)\nabla u) + \boldsymbol{b}(x) \cdot \nabla u + c(x)u. \tag{3.1}$$

We assume L to be elliptic, the coefficients to be in L^∞ and A to be symmetric.

Our goal is to approximate the weak solution of the boundary value problem (BVP)

$$\begin{cases} Lu &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \Gamma_D, \end{cases} \tag{3.2}$$

on a domain $\Omega \subset \mathbb{R}^3$ which we assume to be parameterized by a regular diffeomorphism

$$G : \hat{\Omega} \longrightarrow \Omega$$

on the parameter domain $\hat{\Omega} = [0,1]^3$.

Since we only want to work on the parameter domain and not on the physical domain, the next step is to transform the differential operator L to functions

$\hat{u} = u \circ G$ on the parameter domain. A short computation confirms that

$$\hat{L}\left(\hat{u} \circ G^{-1}\right) = \frac{-1}{|\det J_G|}\hat{\nabla} \cdot \left(|\det J_G| J_G^{-1} A J_G^{-T} \hat{\nabla}\hat{u}\right) + (J_G^{-1}\boldsymbol{b}) \cdot \hat{\nabla}\hat{u} + c\hat{u}.$$

Thus, we transform the original BVP to an equivalent BVP

$$\begin{cases} \hat{L}\hat{u} &= \hat{f} \quad \text{in } \hat{\Omega}, \\ \hat{u} &= 0 \quad \text{on } \hat{\Gamma}_D, \end{cases} \tag{3.3}$$

on the parameter domain $\hat{\Omega}$ by setting $\hat{f} = |\det J_G| f \circ G$ and

$$\hat{L}\hat{u} = |\det J_G| L\left(\hat{u} \circ G^{-1}\right) = -\hat{\nabla} \cdot (K\hat{\nabla}\hat{u}) + \boldsymbol{\ell} \cdot \hat{\nabla}\hat{u} + m\hat{u},$$

where

$$K = |\det(J_G)|(J_G)^{-1} A (J_G)^{-T}, \tag{3.4}$$

$$\boldsymbol{\ell} = |\det(J_G)|(J_G)^{-1}\boldsymbol{b}, \tag{3.5}$$

$$m = |\det(J_G)|c. \tag{3.6}$$

The associated bilinear form of $\hat{L}$ on $V \times V$, where $V = H_0^1(\hat{\Omega})$, is defined as

$$\hat{b}(\hat{u}, \hat{v}) = \int_{\hat{\Omega}} \hat{\nabla}\hat{u} \cdot (K\hat{\nabla}\hat{v}) + (\boldsymbol{\ell} \cdot \hat{\nabla}\hat{u})\hat{v} + m\hat{u}\hat{v}\,\mathrm{d}\xi \tag{3.7}$$

and the weak formulation of the transformed problem on the parameter domain is

$$\hat{b}(\hat{u}, \hat{v}) = \hat{\lambda}(\hat{v}) \text{ for all } \hat{v} \in V, \tag{3.8}$$

where the right hand side is given by the linear functional

$$\hat{\lambda}(\hat{v}) = \int_{\hat{\Omega}} \hat{f}\hat{v}\,\mathrm{d}\xi. \tag{3.9}$$

Finally we note that if $\hat{u} \in V$ is a weak solution to the BVP on the parameter domain, then

$$u = \hat{u} \circ G^{-1} \in H_0^1(\Omega)$$

is a weak solution to the BVP on the physical domain.

3.2. Isogeometric discretization

Under certain conditions on the coefficients, the weak problem (3.8) has a unique solution. Assuming that a unique solution exists, we approximate it using an isogeometric Galerkin method, which means that we perform a discretization based on tensor–product spline functions.

More precisely, we choose three univariate spline spaces $S^{p_\ell}_{\boldsymbol{\tau}_\ell}$ of degree p_ℓ with open knot vectors $\boldsymbol{\tau}_\ell$, $\ell = 1, 2, 3$, and consider their tensor-product

$$\mathbb{S} = S^{p_1}_{\boldsymbol{\tau}_1} \otimes S^{p_2}_{\boldsymbol{\tau}_2} \otimes S^{p_3}_{\boldsymbol{\tau}_3}.$$

To simplify notation, we shall omit the indices for the polynomial degrees. We denote by

$$\{\hat{\beta}^1_i\}_{i\in\{1,\dots,n_1\}}, \{\hat{\beta}^2_j\}_{j\in\{1,\dots,n_2\}}, \text{ and } \{\hat{\beta}^3_k\}_{k\in\{1,\dots,n_3\}}$$

the basis functions of the univariate spline spaces. We assume that homogeneous Dirichlet boundary conditions have been implemented in each of the spaces. The basis functions of the tensor–product spline space $\mathbb{S}$ are given as the tensor product of the univariate bases. We denote them by

$$\hat{\beta}_{ijk}(\xi) = \hat{\beta}^1_i(\xi_1)\hat{\beta}^2_j(\xi_2)\hat{\beta}^3_k(\xi_3).$$

In addition we will use the basis functions

$$\hat{\beta}_{ij}(\xi_{12}) = \hat{\beta}_{ij}(\xi_1, \xi_2) = \hat{\beta}^1_i(\xi_1)\hat{\beta}^2_j(\xi_2)$$

of the bivariate spline space $S^{p_1}_{\tau_1} \otimes S^{p_2}_{\tau_2}$, were we use the notation $\xi_{12} = (\xi_1, \xi_2)$. Whenever it is clear from context we will omit the upper indices on the univariate basis functions.

In order to assemble the system matrix for the Galerkin method we need to evaluate the bilinear form (3.7) for all pairs of basis functions of $\mathbb{S}$. In particular

we have to compute the stiffness matrix S with elements

$$\begin{aligned} S_{(ijk),(i'j'k')} &= \int_{(0,1)^3} \hat{\nabla}\hat{\beta}_{ijk}(\xi)\left(K(\xi)\hat{\nabla}\hat{\beta}_{i'j'k'}(\xi)\right)\mathrm{d}\xi \\ &= \sum_{p,q=1}^{3}\int_{(0,1)^3}\frac{\partial}{\partial\xi_p}\hat{\beta}_{ijk}K_{pq}\frac{\partial}{\partial\xi_q}\hat{\beta}_{i'j'k'}\,\mathrm{d}\xi, \end{aligned} \tag{3.10}$$

the advection matrix C with elements

$$C_{(ijk),(i'j'k')} = \int_{(0,1)^3}(\boldsymbol{\ell}(\xi)\cdot\hat{\nabla}\hat{\beta}_{ijk}(\xi))\hat{\beta}_{i'j'k'}(\xi)\,\mathrm{d}\xi = \sum_{p=1}^{3}\int_{(0,1)^3}\boldsymbol{\ell}_p\frac{\partial}{\partial\xi_p}\hat{\beta}_{ijk}\hat{\beta}_{i'j'k'}\,\mathrm{d}\xi \tag{3.11}$$

and the mass matrix M with elements

$$M_{(ijk),(i'j'k')} = \int_{(0,1)^3} m(\xi)\hat{\beta}_{ijk}(\xi)\hat{\beta}_{i'j'k'}(\xi)\,\mathrm{d}\xi. \tag{3.12}$$

Here, we use (ijk) to denote a lexicographical ordering of the components.

The approximation u_h of the exact solution to the weak problem (3.8) is then

$$u_h = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} u_{ijk}\hat{\beta}_{ijk}\,,$$

where the coefficients u_{ijk} are obtained by solving the linear problem

$$Bu = F.$$

The matrix representation B of the bilinear form $\hat{b}(\cdot,\cdot)$ with respect to the B-spline basis is the sum of S, C and M. The right hand side vector F is computed accordingly and its entries are inner products of the form $(\hat{f},\hat{\beta}_{ijk})_{L^2}$.

The spline discretization $u_h \in \mathbb{S}$ of the transformed problem (3.3) is an *isogeometric discretization* of the original boundary value problem (3.2) if the regular diffeomorphism G, which is called the geometry mapping, is an element of the space $\mathbb{S}^3$. In this situation one uses the same space for discretizing the problem and for representing the geometry.

More generally, one considers geometry mappings of the form G/w, where $w \in \mathbb{S}$ is a non-negative denominator. In this situation, the transformation of the original problem (3.2) to an equivalent BVP on the parameter domain needs to use the substitution $u = (\hat{u}/w) \circ G^{-1}$. We omit any details, since this leads to slightly involved formulas for the quantities K, $\boldsymbol{\ell}$ and m that define the equivalent BVP (3.3). Again, one obtains an isogeometric discretization by considering spline functions $u_h \in \mathbb{S}$.

4. Efficient matrix assembly using partial tensor decomposition

In this chapter, we present our new partial tensor decomposition method. First we recall in Section 4.1 the theory of singular value decomposition of functions, in particular in finite dimensional function spaces. We then apply this to decouple the isogeometric discretization in Section 4.2. The following section examines the assembly of the system matrices using the decoupled discretization. Based on this we analyze the consistency errors and their contribution to the total error in Section 4.4. We complete the analysis by discussing the complexity of our method and by comparing it to the full decomposition method from [64] in Section 4.5. Finally, we present the results of our numerical experiments.

4.1. Singular value decomposition in finite dimensional function spaces

Let $\Omega_1 \subset \mathbb{R}^{d_1}$ and $\Omega_2 \subset \mathbb{R}^{d_2}$ be domains of dimensions $d_1, d_2 \in \mathbb{N}$. We will use the singular value decomposition (SVD) of matrices to compute low-rank approximations of functions

$$f : \Omega_1 \times \Omega_2 \longrightarrow \mathbb{R}$$

in a tensor–product space $\Phi \otimes \Psi$ generated by two finite–dimensional spaces $\Phi \subset C(\Omega_1)$ and $\Psi \subset C(\Omega_2)$. Analogously to the rank of a matrix, the rank of a function $g \in \Phi \otimes \Psi$, denoted by $\operatorname{rank}(g)$, is defined to be the smallest integer such

that

$$g(\xi) = \sum_{r=1}^{\operatorname{rank}(g)} g_r^1(\xi_1) g_r^2(\xi_2)$$

with $\{g_r^1\} \subset \Phi$ and $\{g_r^2\} \subset \Psi$.

First we recall the SVD of a matrix and some of its properties, see e.g. [34, Section 2.5.3]: For any matrix $C \in \mathbb{R}^{\mu\times\nu}$, there exist orthogonal matrices $U \in \mathbb{R}^{\mu\times\mu}$, $V \in \mathbb{R}^{\nu\times\nu}$ and a rectangular diagonal matrix $\Sigma \in \mathbb{R}^{\mu\times\nu}$, such that

$$C = U\Sigma V^T = \sum_{r=1}^{\min(\mu,\nu)} \sigma_r \mathbf{u}_r \mathbf{v}_r^T,$$

where the diagonal entries of Σ, which are called the singular values, satisfy $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_{\min(\mu,\nu)} \geq 0$ and $\mathbf{u}_r$ and $\mathbf{v}_r$ are the columns of the matrices U and V. By defining the truncated diagonal matrix

$$(\Sigma_R)_{ij} = \begin{cases} \sigma_i & \text{for } i = j \leq R \\ 0 & \text{otherwise} \end{cases}$$

for any positive integer R, we obtain the *rank-R approximation*

$$C_R = U\Sigma_R V^T = \sum_{r=1}^{R} \sigma_r \mathbf{u}_r \mathbf{v}_r^T \tag{4.1}$$

of the given matrix C.

In order to apply SVD to the function approximation problem, we choose two bases

$$\boldsymbol{\phi} = \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_\mu \end{pmatrix} \text{ and } \boldsymbol{\psi} = \begin{pmatrix} \psi_1 \\ \vdots \\ \psi_\nu \end{pmatrix}$$

for the two finite-dimensional spaces $\Phi \subset C(\Omega_1)$ and $\Psi \subset C(\Omega_2)$ of dimensions μ and ν, respectively. The Euclidean inner product of the coefficients then defines the inner product

$$\langle h, h' \rangle_\Phi = \mathbf{h}^T \mathbf{h}' = \sum_{i=1}^{\mu} h_i h_i'$$

on Φ (and analogously for Ψ), where the functions are represented as

$$h = \langle \mathbf{h}, \boldsymbol{\phi} \rangle_{\mathbb{R}^\mu} = \sum_{i=1}^{\mu} h_i \phi_i \quad \text{and} \quad h' = \langle \mathbf{h}', \boldsymbol{\phi} \rangle_{\mathbb{R}^\mu} = \sum_{i=1}^{\mu} h'_i \phi_i$$

with coefficient vectors $\mathbf{h}$ and $\mathbf{h}'$. These inner products on Φ and Ψ induce an inner product on

$$\Phi \otimes \Psi \subset C(\Omega_1 \times \Omega_2),$$

which is equal to the inner product defined by the tensor–product basis $\boldsymbol{\phi} \otimes \boldsymbol{\psi}$. We find it convenient to express it via the Frobenius inner product $\langle ., . \rangle_F$ of matrices,

$$\langle g, g' \rangle_{\Phi \otimes \Psi} = \langle G, G' \rangle_F = \sum_{i=1}^{\mu} \sum_{j=1}^{\nu} G_{ij} G'_{ij},$$

since the functions $g, g' \in \Phi \otimes \Psi$ have coefficient matrices G and G' with respect to the basis $\boldsymbol{\phi} \otimes \boldsymbol{\psi}$. An analogous relation connects the Euclidean coefficient norm $\|.\|_{\Phi \otimes \Psi}$ with the Frobenius norm $\|.\|_F$ of the coefficient matrices.

The Frobenius inner product also furnishes the compact notation

$$f = \langle C, \boldsymbol{\phi} \otimes \boldsymbol{\psi} \rangle_F = \sum_{i=1}^{\mu} \sum_{j=1}^{\nu} C_{ij} \phi_i \psi_j, \tag{4.2}$$

where $C \in \mathbb{R}^{\mu \times \nu}$ is the coefficient matrix, for the representation of any function $f \in \Phi \otimes \Psi$ with respect to the tensor-product basis.

Combining (4.2) with the SVD of the coefficient matrix C leads to the decomposition

$$f = \sum_{r=1}^{\min(\mu,\nu)} \sigma_r \langle \mathbf{u}_r, \boldsymbol{\phi} \rangle_{\mathbb{R}^\mu} \langle \mathbf{v}_r, \boldsymbol{\psi} \rangle_{\mathbb{R}^\nu} . \tag{4.3}$$

Considering again the truncated diagonal matrix, we obtain the *rank-R approximation*

$$f_R = \langle C_R, \boldsymbol{\phi} \otimes \boldsymbol{\psi} \rangle_F = \sum_{r=1}^{R} \sigma_r \langle \mathbf{u}_r, \boldsymbol{\phi} \rangle_{\mathbb{R}^\mu} \langle \mathbf{v}_r, \boldsymbol{\psi} \rangle_{\mathbb{R}^\nu} , \tag{4.4}$$

of the function f with respect to the tensor product basis $\boldsymbol{\phi} \otimes \boldsymbol{\psi}$.

Lemma 1. *The rank-R approximation* (4.4) *solves the best approximation problem*

$$\min_{\{g\in\Phi\otimes\Psi:\ \mathrm{rank}(g)\le R\}} \|f-g\|_{\Phi\otimes\Psi}$$

and the resulting approximation error equals

$$\|f-f_R\|_{\Phi\otimes\Psi} = \|C-C_R\|_F = \sqrt{\sum_{r=R+1}^{\min(\mu,\nu)} \sigma_r^2}.$$

Proof. It is known that the approximation error of the rank-R approximation C_R of the matrix C in the Frobenius norm is given by

$$\|C-C_R\|_F = \sqrt{\sum_{r=R+1}^{\min(\mu,\nu)} \sigma_r^2} \tag{4.5}$$

and it is the best approximation of C by a matrix of rank R in the Frobenius norm, see [34, Lemma 2.30]. With our choice of norm on $\Phi\otimes\Psi$ these properties are transferred to the function f and its rank-R approximation f_R. □

We establish a simple bound on the approximation error in the L^∞ norm:

Lemma 2. *Given f as in* (4.2) *and its rank-R approximation f_R as in* (4.4) *we have*

$$\|f-f_R\|_{L^\infty(\Omega_1\times\Omega_2)} \le \Big(\max_{\xi\in\Omega_1\times\Omega_2} \|(\boldsymbol{\phi}\otimes\boldsymbol{\psi})(\xi)\|_F\Big) \sqrt{\sum_{r=R+1}^{\min(\mu,\nu)} \sigma_r^2}\,.$$

Proof. We rewrite the L^∞ norm of the approximation error and use the Cauchy-Schwarz inequality to obtain

$$\begin{aligned}
\|f-f_R\|_{L^\infty} &= \|\langle C, \boldsymbol{\phi}\otimes\boldsymbol{\psi}\rangle_F - \langle C_R, \boldsymbol{\phi}\otimes\boldsymbol{\psi}\rangle_F\|_{L^\infty} \\
&= \max_{\xi\in\Omega_1\times\Omega_2} |\langle C-C_R, (\boldsymbol{\phi}\otimes\boldsymbol{\psi})(\xi)\rangle_F| \\
&\le \Big(\max_{\xi\in\Omega_1\times\Omega_2} \|(\boldsymbol{\phi}\otimes\boldsymbol{\psi})(\xi)\|_F\Big)\, \|C-C_R\|_F.
\end{aligned}$$

The result now follows from (4.5). □

This result is useful if we are able to control the constant that appears in front of the square root. For instance, this constant can be evaluated easily when using tensor-product B-splines.

Another result can be derived for the approximation error in the L^2 norm if one uses L^2–orthonormal bases $\{\phi_i\}_{i=1}^{\mu}$ and $\{\psi_j\}_{j=1}^{\nu}$. Indeed, the coefficient-based inner products defined on Φ, Ψ and $\Phi \otimes \Psi$ are then equal to the corresponding L^2 products. Consequently, the rank-R approximation is the best approximation with respect to the L^2-norm and the error satisfies

$$\|f - f_R\|_{L^2} = \sqrt{\sum_{r=R+1}^{\min(\mu,\nu)} \sigma_r^2}. \tag{4.6}$$

In this case, the representation (4.3) is equal to the SVD of an L^2 function, which exists for any function in $L^2(\Omega_1 \times \Omega_2)$, see [34, Corollary 4.115]. In the infinite dimensional case, any function $f \in L^2(\Omega_1 \times \Omega_2)$ has a decomposition

$$f = \sum_{r=1}^{\infty} \sigma_r u_r v_r$$

where $\{u_r\}_{r=1}^{\infty}$ is an orthonormal system of $L^2(\Omega_1)$ and $\{v_r\}_{r=1}^{\infty}$ is an orthonormal system of $L^2(\Omega_2)$. The singular values are known to decay like

$$\sigma_r \lesssim r^{-\frac{s}{\min(d_1,d_2)}},$$

if $f \in H^s(\Omega_1 \times \Omega_2)$, see [33, Theorem 3.1].

4.2. Decoupling the isogeometric Galerkin discretization

In order to evaluate the trivariate integrals in the components of the system matrices in an efficient way, we will replace the components of K, $\boldsymbol{\ell}$ and m as well as the right hand side $\hat{f}$ by sums of products of bivariate and univariate functions. These components define *weight functions* that are present in the trivariate integrals. By

performing the replacement step we are able to replace the expensive trivariate numerical integration by a number of univariate and bivariate ones. Clearly, the error introduced by this operation needs to be controlled carefully.

As the first step, we project each weight function into an appropriate spline space. In some cases, the weight function is already contained in some spline space, hence the projection does not introduce any error. Otherwise we choose the spline space such that the error ϵ_Π does not exceed a given error tolerance. The resulting approximate weight function is represented by tensor–product splines.

In the second step, we decompose the approximate weight function using the results presented in Section 4.1. To this end, we consider the factorization of its domain as the Cartesian product $[0,1]^2 \times [0,1]$ (or any of the three possible splittings of $[0,1]^3$) and use the associated low-rank approximation. Given a tolerance ϵ_Λ, we obtain a rank-R approximation of the approximate weight function.

4.2.1. Spline projection

Let g be one of the weight functions, i.e., any component K_{pq} of K, any component $\boldsymbol{\ell}_p$ of $\boldsymbol{\ell}$ or either of the functions m and $\hat{f}$. We project g into some tensor–product spline space

$$\bar{\mathbb{S}} = S^{\bar{p}_1}_{\gamma_1} \otimes S^{\bar{p}_2}_{\gamma_2} \otimes S^{\bar{p}_3}_{\gamma_3} = \left(S^{\bar{p}_1}_{\gamma_1} \otimes S^{\bar{p}_2}_{\gamma_2}\right) \otimes S^{\bar{p}_3}_{\gamma_3}. \tag{4.7}$$

Note that in (4.7) we can choose each of the three possible splittings of the tensor-product space by combining two of the univariate spline spaces. In the complexity analysis performed in Section 4.5.1 we will observe that it is beneficial to choose the one resulting in the lowest rank. In general, this space will be different from the discretization space $\mathbb{S}$. Since the weight functions depend on the geometry mapping, we keep the knots of its spline representation. The choice of $\bar{\mathbb{S}}$ will be discussed later on in this section.

Using the notation of Section 4.1, the splitting (4.7) of $\bar{\mathbb{S}}$ corresponds to

$$\bar{\mathbb{S}} = \Phi \otimes \Psi, \quad \text{where} \quad \Phi = S^{q_1}_{\gamma_1} \otimes S^{q_2}_{\gamma_2}, \quad \Psi = S^{q_3}_{\gamma_3}. \tag{4.8}$$

Instead of separating the last variable of the tensor–product spline space from the first two, we could have selected each of the three variables. The potential benefits

will be analyzed experimentally in Section 4.6.

Choosing bases $\boldsymbol{\phi} = \{\bar{\beta}_{ij}\}$ of $\Phi = S^{\bar{p}_1}_{\gamma_1} \otimes S^{\bar{p}_2}_{\gamma_2}$ and $\boldsymbol{\psi} = \{\bar{\beta}_k\}$ of $\Psi = S^{\bar{p}_3}_{\gamma_3}$ results in the usual tensor-product basis functions

$$\bar{\beta}_{ijk}(\xi) = \bar{\beta}_{ij}(\xi_1, \xi_2)\bar{\beta}_k(\xi_3)$$

of the spline space $\Phi \otimes \Psi = \bar{\mathbb{S}}$. The particular choice of the basis will be determined by the error norm that we want to control. In particular we will use either B-splines or an L^2-orthonormal basis of the spline space, see next section.

We now consider a spline projection operator

$$\Pi : C([0,1]^3) \to \bar{\mathbb{S}},$$

which will be either interpolation at the Greville points or orthogonal L^2 projection. Both can be implemented efficiently by exploiting the tensor-product structure $\Pi = \Pi_1 \otimes \Pi_2 \otimes \Pi_3$, since one can realize them by sequentially applying the corresponding univariate operators. The particular choice of the projector is again related to the error norm that we want to control.

Applying the operator Π to the weight function g gives a tensor-product spline function

$$(\Pi g)(\xi) = \sum_{i=1}^{\bar{n}_1} \sum_{j=1}^{\bar{n}_2} \sum_{k=1}^{\bar{n}_3} G_{ijk} \bar{\beta}_{ij}(\xi_{12}) \bar{\beta}_k(\xi_3),$$

where the G_{ijk} form the coefficient tensor with respect to the chosen basis. We assume that the projection error satisfies

$$\|g - \Pi g\| \leq \epsilon_\Pi$$

for a given tolerance ϵ_Π. We will use the L^∞ norm if g is one of the components of K, $\boldsymbol{\ell}$ or m and the L^2 norm if $g = \hat{f}$.

The standard error bounds for spline functions can be used to analyze the asymptotic behavior of the error as the number of knots of $\bar{\mathbb{S}}$ increases. These will be used later for the theoretical analysis. In the implementation, we rely on a simple sampling-based approach in order to estimate the norm of the approximation error for any specific choice of $\bar{\mathbb{S}}$ and Π.

There is no error in some cases: The function $m = |\det(J_G)|c$, which is needed when computing the mass matrix M, is itself a spline function if c is a spline function, too. One may choose the space $\bar{\mathbb{S}}$ such that m can be represented exactly. This also applies to the components of $\boldsymbol{\ell} = |\det(J_G)|(J_G)^{-1}\boldsymbol{b}$ if the elements of b are spline functions, too.

For the stiffness matrix, every component of K is a rational function with differentiability reduced by one compared to G if the components of A are sufficiently smooth spline functions, too. We then use a high-degree tensor–product spline space for the approximation, see [64, Section 6.1], since it provides a highly accurate approximation while simultaneously requiring only very few knots.

4.2.2. Partial tensor decomposition

We apply the theory presented in Section 4.1 to decompose the spline function Πg in the space $\Phi \otimes \Psi = \bar{\mathbb{S}}$. Since we only split one of the three directions of the coefficient tensor instead of computing a full (canonical) decomposition, we call it *partial tensor decomposition*.

Taking the considered factorization (4.8) into account, the coefficients G_{ijk} of Πg form the matrix $(C_{(ij)k})$ with elements $C_{(ij)k} = G_{ijk}$, the notation (ij) again indicates that several indices are combined into a single one by a lexicographic ordering. By computing the SVD of this coefficient matrix, as described in Section 4.1, we obtain the decomposition (4.3) of Πg into a sum of products of bivariate and univariate functions,

$$(\Pi g)(\xi) = \sum_{r=1}^{\min(\bar{n}_1 \cdot \bar{n}_2, \bar{n}_3)} \mathcal{U}^r(\xi_{12}) \mathcal{V}^r(\xi_3),$$

with the bivariate and univariate factors

$$\mathcal{U}^r(\xi_{12}) = \sqrt{\sigma^r} \sum_{i=1}^{\bar{n}_1} \sum_{j=1}^{\bar{n}_2} u_{ij}^r \bar{\beta}_{ij}(\xi_{12}) \quad \text{and} \quad \mathcal{V}^r(\xi_3) = \sqrt{\sigma^r} \sum_{k=1}^{\bar{n}_3} v_k^r \bar{\beta}_k(\xi_3).$$

We truncate the sum to obtain an low-rank approximation of Πg. For a given rank

value R we define the *partial rank-R approximation*

$$(\Lambda g)(\xi) = \sum_{r=1}^{R} \mathcal{U}^r(\xi_{12})\mathcal{V}^r(\xi_3). \tag{4.9}$$

We present two error bounds for the truncation. First we consider the L^∞ norm, which is closely related to using B-splines:

Lemma 3. *The truncation error of the partial rank-R approximation satisfies*

$$||\Pi g - \Lambda g||_{\mathrm{L}^\infty} \leq \sqrt{\sum_{r=R+1}^{\min(\bar{n}_1\cdot\bar{n}_2,\bar{n}_3)} (\sigma^r)^2}$$

if the basis functions $\{\bar{\beta}_{ijk}\}$ *are tensor–product B-splines.*

Proof. Recall that the tensor-product splines form a non-negative partition of unity. Consequently, the constant in Lemma 2 does not exceed 1, since

$$\|(\boldsymbol{\phi} \otimes \boldsymbol{\psi})(\xi)\|_F = \sum_{i=1}^{\bar{n}_1}\sum_{j=1}^{\bar{n}_2}\sum_{k=1}^{\bar{n}_3}(\bar{\beta}_{ijk}(\xi))^2 \leq \sum_{i=1}^{\bar{n}_1}\sum_{j=1}^{\bar{n}_2}\sum_{k=1}^{\bar{n}_3}\bar{\beta}_{ijk}(\xi) = 1.$$

□

We use the estimate in Lemma 3 to select the smallest rank R, such that the truncation error does not exceed a given tolerance ϵ_Λ. This can be done by computing the full SVD (in this case we know all singular values σ_r) or by computing a truncated SVD incrementally (in this case we exploit the fact that the singular values are decreasing). The first method was found to be sufficient for our experimental results.

Combining the effects of approximation and truncation gives the bound

$$\|g - \Lambda g\|_{L^\infty} \leq \epsilon = \epsilon_\Pi + \epsilon_\Lambda.$$

Clearly, the L^∞-norm also provides an upper bound on the L^2-norm.

Second, we obtain an optimal bound on the L^2-norm of the truncation error by considering orthonormal bases. Such bases can be obtained by applying Gram-

Schmidt orthogonalization to B-splines. An alternative construction is described in [90].

Lemma 4. *If $\{\bar{\beta}_{ij}\}$ and $\{\bar{\beta}_k\}$ are L^2-orthonormal bases, then the truncated function Λg is the best approximation with respect to the L^2-norm among all functions of rank R with respect to the factorization $\bar{\mathbb{S}} = \Phi \otimes \Psi$. Moreover, the truncation error satisfies*

$$||\Pi g - \Lambda g||_{\mathrm{L}^2} = \sqrt{\sum_{r=R+1}^{\min(\bar{n}_1 \cdot \bar{n}_2, \bar{n}_3)} (\sigma^r)^2}.$$

Proof. Since we consider L^2-orthonormal bases, the Frobenius norm of the coefficient matrix of a function is equal to its L^2-norm. Thus we can prove the result by applying Lemma 1(i). □

Again, we select the smallest rank R, such that the approximation error does not exceed the given tolerance ϵ_Λ, now considering the L^2-norm. Combining the effects approximation and truncation gives the bound

$$\|g - \Lambda g\|_{L^2} \leq \epsilon = \epsilon_\Pi + \epsilon_\Lambda.$$

4.3. Matrix assembly

The decomposition of the integrands in (3.10), (3.11) and (3.12) of the components of the system matrix results in a decomposition of the system matrix itself. We represent the approximate mass, advection and stiffness matrix in the general form

$$\sum_{r=1}^{\varrho} X_r \otimes Y_r \tag{4.10}$$

with $X_r \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ and $Y_r \in \mathbb{R}^{n_3 \times n_3}$. We refer to the integer ϱ as the Kronecker rank of the matrix.

The low-rank approximation of the components of K makes it possible to assemble the stiffness matrix by computing bivariate and univariate integrals. Replacing K_{pq} by

$$\Lambda K_{pq} = \sum_{r=1}^{R_{pq}} \mathcal{U}_{pq}^r(\xi_{12}) \mathcal{V}_{pq}^r(\xi_3)$$

in (3.10), where R_{pq} is the rank value used for generating the approximation of the matrix element K_{pq}, gives the approximate stiffness matrix

$$\tilde{S}_{(ijk),(i'j'k')} = \sum_{p,q=1}^{3} \sum_{r=1}^{R_{pq}} \underbrace{\int_{[0,1]^2} \mathcal{U}_{pq}^r \left((1-\delta_{p3}) \frac{\partial \hat{\beta}_{ij}}{\partial \xi_p} + \delta_{p3} \hat{\beta}_{ij} \right) \left((1-\delta_{q3}) \frac{\partial \hat{\beta}_{i'j'}}{\partial \xi_q} + \delta_{q3} \hat{\beta}_{i'j'} \right) \mathrm{d}\xi_{12}}_{= X_{pq,(ij)(i'j')}^r} \tag{4.11}$$

$$\cdot \underbrace{\int_0^1 \mathcal{V}_{pq}^r \left(\delta_{p3} \frac{\partial \hat{\beta}_k}{\partial \xi_p} + (1-\delta_{p3}) \hat{\beta}_k \right) \left(\delta_{q3} \frac{\partial \hat{\beta}_{k'}}{\partial \xi_q} + (1-\delta_{q3}) \hat{\beta}_{k'} \right) \mathrm{d}\xi_3}_{= Y_{pq,kk'}^r} .$$

(4.12)

This can be verified by using the identity

$$\frac{\partial}{\partial \xi_p} \hat{\beta}_{ijk} = \left((1-\delta_{p3}) \frac{\partial \hat{\beta}_{ij}}{\partial \xi_p} + \delta_{p3} \hat{\beta}_{ij} \right) \left(\delta_{p3} \frac{\partial \hat{\beta}_k}{\partial \xi_p} + (1-\delta_{p3}) \hat{\beta}_k \right),$$

with the Kronecker delta δ_{p3}.

The approximate stiffness matrix can thus be written as a sum of $\varrho = \sum_{p,q=1}^{3} R_{pq}$ Kronecker products

$$\tilde{S} = \sum_{p,q=1}^{3} \sum_{r=1}^{R_{pq}} X_{pq}^r \otimes Y_{pq}^r$$

where the elements of the $n_1 n_2 \times n_1 n_2$ matrices X_{pq}^r and of the $n_3 \times n_3$ matrices Y_{pq}^r have been defined in (4.11) and (4.12), respectively. Consequently, we obtain an approximate representation of the stiffness matrix in the form (4.10) with Kronecker rank ϱ. The other two matrices can be dealt with analogously.

4.4. Error analysis

Next, we investigate the convergence of the discretized solution of the weak formulation using the approximate bilinear form and right hand side. Similarly to Section 3.1, we set $V = H_0^1(\hat{\Omega})$ and denote by $\hat{u} \in V$ the solution to the transformed

problem (3.8).

Furthermore, we consider a sequence of discretizations, see Section 3.2, which are based on spline spaces $V_h = \mathbb{S}$ with decreasing diameter of the elements $h \to 0$, and denote by $u_h \in \mathbb{S}$ the solution of

$$\hat{b}_h(u_h, v_h) = \hat{\lambda}_h(v_h) \text{ for all } v_h \in V_h,$$

where

$$\hat{b}_h(u_h, v_h) = \int_{\hat{\Omega}} \hat{\nabla} u_h \cdot ((\Lambda_h K)\hat{\nabla} v_h) + ((\Lambda_h \boldsymbol{\ell}) \cdot \hat{\nabla} u_h) v_h + (\Lambda_h m) u_h v_h \, \mathrm{d}\xi \tag{4.13}$$

and

$$\hat{\lambda}_h(v_h) = \int_{\hat{\Omega}} \left(\Lambda_h \hat{f}\right) w_h \, \mathrm{d}\xi. \tag{4.14}$$

We shall see that convergence can be guaranteed only by considering an associated sequence of partial rank-R approximations with h-dependent values of the rank value R. We thus use Λ_h to denote the operator defined in (4.9) that transforms a weight function into its approximation.

The analysis is based on Strang's first lemma, see [88]: Since $\hat{b}_h$ is uniformly elliptic, there is a constant C such that the overall approximation error can be estimated by

$$\begin{aligned} \|\hat{u} - u_h\|_V \leq C \Bigg(\inf_{v_h \in V_h} \Bigg(\|\hat{u} - v_h\|_V + \sup_{w_h \in V_h} \frac{|\hat{b}(v_h, w_h) - \hat{b}_h(v_h, w_h)|}{\|w_h\|_V} \Bigg) + \\ + \sup_{w_h \in V_h} \frac{|\lambda(w_h) - \lambda_h(w_h)|}{\|w_h\|_V} \Bigg). \end{aligned} \tag{4.15}$$

We control the consistency error, i.e. the second and third term of the right hand side in Strang's lemma, by suitably choosing the error tolerance as described in Section 4.2.2. First we consider the second term, which corresponds to the approximation of the bilinear form $\hat{b}$ in (3.7) by the approximate bilinear form $\hat{b}_h$ in (4.13):

Lemma 5. *The approximate bilinear form satisfies the inequality*

$$\sup_{w_h \in V_h} \frac{|\hat{b}(v_h, w_h) - \hat{b}_h(v_h, w_h)|}{\|w_h\|_V} \leq \epsilon_h \|v_h\|_V$$

for all $v_h \in V_h$, where the parameter ϵ_h on the right hand-side is determined by the error of the low rank approximation Λ_h,

$$\epsilon_h = 3 \cdot \max\{\|K_{pq} - (\Lambda_h K)_{pq}\|_{L^\infty(\hat{\Omega})}\,,\ \|\boldsymbol{\ell}_p - (\Lambda_h \boldsymbol{\ell})_p\|_{L^\infty(\hat{\Omega})}\,,\ \|m - \Lambda_h m\|_{L^\infty(\hat{\Omega})}\}.$$

Proof. Considering the first term of $|\hat{b}(v_h, w_h) - \hat{b}_h(v_h, w_h)|$, using the Cauchy-Schwarz inequality confirms that

$$\left|\int_{\hat{\Omega}} \hat{\nabla} v_h \cdot K \hat{\nabla} w_h - \hat{\nabla} v_h \cdot (\Lambda_h K) \hat{\nabla} w_h \,\mathrm{d}\xi\right| = \left|\sum_{pq=1}^{3} \int_{\hat{\Omega}} (K_{pq} - (\Lambda_h K)_{pq}) \frac{\partial v_h}{\partial \xi_p} \frac{\partial w_h}{\partial \xi_q} \,\mathrm{d}\xi\right|$$
$$\leq \sum_{p,q=1}^{3} \|K_{pq} - \Lambda_h K_{pq}\|_{L^\infty(\hat{\Omega})} \left\|\frac{\partial v_h}{\partial \xi_p}\right\|_{L^2(\hat{\Omega})} \left\|\frac{\partial w_h}{\partial \xi_q}\right\|_{L^2(\hat{\Omega})} \leq \frac{\epsilon_h}{3} \|v_h\|_V \|w_h\|_V.$$

Since we can estimate the L^2-norm by the H^1-norm, the same bound applies to the two remaining terms. □

Second we consider the third term, which corresponds to the approximation of the linear form $\hat{\lambda}$ in (3.9) by the approximate linear form $\hat{\lambda}_h$ in (4.14):

Lemma 6. *The approximate linear form satisfies the inequality*

$$\sup_{w_h \in V_h} \frac{|\hat{\lambda}(w_h) - \hat{\lambda}_h(w_h)|}{\|w_h\|_V} \leq \epsilon'_h,$$

where the right-hand side is given by

$$\epsilon'_h = \|\hat{f} - \Lambda_h \hat{f}\|_{L^2(\hat{\Omega})}.$$

Here, $\hat{f} = |\det J_G| f \circ G$ is the right hand side of the transformed problem on the parameter domain.

Proof. By using the Cauchy-Schwarz inequality,

$$\begin{aligned}|\hat{\lambda}(w_h) - \hat{\lambda}_h(w_h)| &= \int_{\hat{\Omega}} \left(\hat{f} - \Lambda_h \hat{f}\right) w_h \,\mathrm{d}\xi \\ &\leq \|\hat{f} - \Lambda_h \hat{f}\|_{L^2(\hat{\Omega})} \|w_h\|_{L^2(\hat{\Omega})} \leq \epsilon'_h \|w_h\|_V.\end{aligned}$$

□

Thus, when approximating the right-hand side, it can be beneficial to use an L^2-orthonormal basis for the projection space since this guarantees the best approximation in the L^2-norm and thus the minimal rank value. Moreover, since the right hand side f may only be in L^2 and not in L^∞ we might not be able to measure the projection error in the L^∞-norm. While the L^2-orthonormal basis guarantees the minimal rank value, we certainly can also use simply the B-spline basis for the low partial rank approximation, using the fact that the L^2-norm of functions on the unit cube can be bounded by the L^∞ norm.

When using a spline space $V_h\mathbb{S}$ of degree $p_1 = p_2 = p_3 = p$, the first term in (4.15), which represents the discretization error, can be estimated as Ch^p, where the constant C depends only on $\hat{u}$. In order to obtain an optimal rate of convergence of the overall problem, we need to make sure that the consistency error possesses the same order of convergence. This is guaranteed by choosing an appropriate sequence of operators Λ_h:

The operators Λ_h are said to be order p-convergent if there exists an h-independent constant C, such that the error bounds in Lemmas 5 and 6 satisfy $\epsilon_h \leq Ch^p$ and $\epsilon'_h \leq Ch^p$. We can now state the desired convergence result:

Theorem 7. *The solution obtained using the approximated bilinear form $\hat{b}_h$ attains the optimal rate of convergence*

$$\|\hat{u} - \hat{u}_h\|_V \leq Ch^p,$$

if the low–rank approximation operators Λ_h are order p-convergent.

The proof is obtained by combining the previous observation concerning the two sources of error (discretization and consistency error).

We conclude this section by describing a particular choice of the discretization and projection spaces that guarantees optimal rate of convergence. Let p and $\bar{p}$ denote the polynomial degrees of the three univariate factors of the tensor-product spaces $\mathbb{S}$ and $\bar{\mathbb{S}}$, respectively (for simplicity we choose uniform degrees). Similarly, we denote by n and $\bar{n}$ the numbers of degrees of freedom of these univariate factors. The total number of degrees of freedom equals n^3 and $\bar{n}^3$, respectively.

We consider an h-dependent sequence of discretization and projection spaces and the associated operators Π_h, Λ_h, where the element size h of the discretization space satisfies $h = O(1/n)$. Note that the element size $\bar{h}$ of the projection space is generally different. Nevertheless we use h to parameterize the operators.

When choosing the polynomial degree of the projection spaces as

$$\bar{p} = \mu p \tag{4.16}$$

for some constant integer $\mu \geq 1$, we obtain order p-convergent interpolation operators Π_h if

$$\bar{n} = O(n^{1/\mu}). \tag{4.17}$$

Indeed, since $\bar{h} = O(1/\bar{n})$, we may use the approximation properties of splines (see [85]) to conclude that

$$\|g - \Pi g\|_{L^\infty(\hat{\Omega})} \leq C_g \left(\frac{1}{\bar{n}}\right)^{\bar{p}+1} \leq C'_g \left(\frac{1}{n}\right)^{p} \leq C''_g h^p.$$

Furthermore, *we obtain order p-convergent low–rank approximation operators Λ_h by choosing the rank value $R = R(g, h)$ such that the truncation error has the same order of magnitude as the interpolation error.* A trivial upper bound is

$$R \leq \bar{n} = O(n^{1/\mu}), \text{ hence also } \varrho \leq O(n^{1/\mu}), \tag{4.18}$$

where ϱ is the total rank in (4.10), since the right-hand side is the maximum rank of the coefficient matrix, which has dimension $\bar{n}^2 \times \bar{n}$. We will later see that a much smaller rank can be used for most geometries. In particular, it is possible to use less degrees of freedom of the projection space if additional information on the properties of the weight function is available, see Section 4.2.1.

4.5. Computational complexity

We analyze the computational complexity of our method with respect to n, the number of degrees of freedom in each direction, and p, the polynomial degree. We compare with the complexity of the classical element-wise Gauss quadrature and the complexity of the low-rank tensor decomposition method based on HOSVD, which is described in [64]. For all methods, the computational effort is bounded from below by the number of its non-zero entries, which is $O(n^3p^3)$.

In the end of the section we analyze the complexity of the method's application to $d+1$-dimensional space-time problems.

4.5.1. Partial tensor decomposition method (PDM)

We briefly recall the algorithm. The input consists of the geometry map, the discretization basis, the coefficients in (3.1), and a consistency tolerance. In order to perform the matrix assembly, we execute the following steps:

1. Perform the projection of the weight functions into the spline space $\bar{\mathbb{S}}$ chosen as described in Section 4.4. When exploiting the tensor-product structure, the complexity of this step is equal to

$$O(\bar{n}\bar{p}^2) = \mu^2 O(n^{1/\mu}p^2),$$

   cf. [64].

2. Perform partial low rank approximation of the weight functions by applying partial SVD on the resulting spline function. The complexity of computing the SVD including the relevant singular vectors equals

$$O(\bar{n}^4) = O(n^{4/\mu}),$$

   see [31]. this can be reduced to $O(\varrho\bar{n}^3) = O(\varrho n^{3/\mu})$ if either the total rank ϱ (or, more precisely, the rank of all weight functions) is known in advance or it is estimated based on the error bound in Lemma 3 during the computation.

3. Assemble the matrices X_r and Y_r for $r = 1, \ldots, \varrho$ using bivariate (tensor-product) and univariate Gauss quadrature, respectively. This step is dominated by the bivariate Gauss quadrature[1], whose total complexity is

$$O(\varrho n^2 p^6).$$

4. Generate the full system matrix by evaluating the sum of Kronecker products (4.10). Since each entry is the sum of ϱ products, the complexity of this step equals

$$O(\varrho n^3 p^3).$$

As analyzed in the end of the previous section we can choose $\bar{n} = O(n^{\frac{1}{\mu}})$, where the positive integer μ determines the degree $\bar{p} = \mu p$. Therefore, in cases where the number of degrees of freedom n is much bigger than the polynomial degree p, the generation of the global matrix is dominated by the last step, that is, the computation of the sum of Kronecker products. The total complexity of PDM is then

$$O(\varrho n^3 p^3).$$

Consequently, our method is optimal if the rank ϱ does not increase with h-refinement. Even if we do not truncate the partial tensor decomposition, we get $\varrho = \bar{n} = O(n^{\frac{1}{\mu}})$ and thus the computational complexity equals $O(n^{3+\frac{1}{\mu}} p^3)$. This means that PDM is especially efficient in connection with h-refinement.

Note that there are three different split directions for the partial tensor decomposition, each resulting in a potentially different rank value for the given accuracy. Since the overall complexity is governed by the computation of the sum of Kronecker products, we choose the split direction that provides the lowest rank even if it is not the one which results in the smallest number of bivariate integrals. In some cases it is obvious which direction will result in the lowest rank, e.g. if the domain is created by a linear sweep. Otherwise, all three splits are computed as computing the SVD is a problem of low complexity relative to the algorithm.

[1] We use Gauss quadrature with $p + 1$ Gauss nodes per knot span in each coordinate direction. Strictly speaking, this corresponds to another approximation of the bilinear and linear forms. The overall accuracy of the simulation is again preserved, since the assumptions of Strang's first lemma are again satisfied.

4.5.2. Element-wise Gauss quadrature

For an element-wise Gauss quadrature we need to iterate over $O(n^3)$ elements and compute $O(p^6)$ values and derivatives of basis functions on each quadrature point. Since the number of quadrature points in each direction is in $O(p)$, we arrive at a total complexity of

$$O(n^3 p^9).$$

4.5.3. Comparison with the full tensor decomposition method (FDM)

In FDM, which is described in [64], the case of arbitrary dimensions is considered. The weight functions are projected to a spline space in the same way as described above and then decomposed into products of d univariate components using higher order SVD.

In the same way as in PDM, this decomposition is then truncated with respect to a rank value $\tilde{\varrho}$ depending on the given error tolerance ϵ. In general, the rank $\tilde{\rho}$ in the full decomposition is larger than the rank ρ obtained in the partial tensor decomposition.

In the 3-dimensional case, FDM leads to a complexity of $O(\bar{n}\bar{p}^2 + \bar{n}^4 + \tilde{\varrho} n p^3)$ for the assembly in the Kronecker format. The assembly of the full matrix from the Kronecker format is the same as in PDM and thus its complexity is in

$$O(\tilde{\varrho} n^3 p^3).$$

The quadrature in FDM is less expensive as we only have to integrate univariate integrals but the dominating step of computing the Kronecker products remains of the same complexity and is proportional to the rank needed to fulfill the given tolerance for the consistency error.

One advantage of PDM over FDM is the optimality of the rank with respect to the Euclidean norm of the coefficient tensor as described in Section 4.1. If we find

a full rank $\tilde{\varrho}$ tensor decomposition

$$\bar{\Lambda} g = \sum_{r=1}^{\tilde{\varrho}} \mathcal{U}_r \otimes \mathcal{V}_r \otimes \mathcal{W}_r = \sum_{r=1}^{\tilde{\varrho}} (\mathcal{U}_r \otimes \mathcal{V}_r) \otimes \mathcal{W}_r,$$

we also have a rank $\tilde{\varrho}$ partial tensor decomposition by combining the first two vectors into a matrix. Since the rank value obtained by using partial SVD is optimal, it follows that

$$\varrho \leq \tilde{\varrho},$$

where ϱ is the rank needed to satisfy the same error tolerance in the partial decomposition.

Summing up, we conclude that – for $n \gg p$ – the complexity of PDM is at most of the same order as the one of FDM and has the potential to be much smaller. Additionally, the total rank ϱ in the partial tensor decomposition is bounded by $O(\bar{n})$, while the rank $\tilde{\varrho}$ that appears in the full decomposition is bounded by $O(\bar{n}^2)$.

4.6. Numerical experiments

We present results of numerical experiments, in order to demonstrate the behavior of PDM. We choose a test suite of different patches that possess different rank values for the Jacobian of the geometry mapping as well as the matrix-valued factor in the stiffness matrix. Figure 4.1 visualizes the patches that we used to test our method, as well as the degrees of freedom and polynomial degrees of the geometry mapping[2].

We will compare the rank values and the computing times with those of FDM. As an additional benchmark, we use the computing times of a classical element-wise Gauss quadrature. All numerical experiments were performed on a standard PC using the G+Smo C++ library [45, 65].

[2]The parametrizations used in this section can be found in `https://github.com/gismo/data/`.

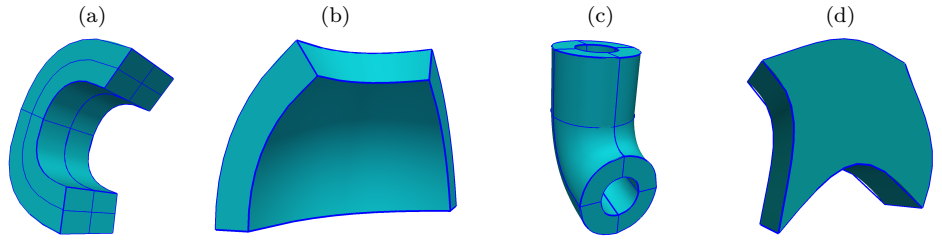


Figure 4.1.: The patches of the test suite.

4.6.1. Rank comparison

Consider the operator (3.1) with $A = I$, $b = 0$ and $c = 1$. We analyze the ranks of the weight functions defined by the Jacobian determinant m and the matrix K, see (3.4) and (3.6). In particular, we will study the stability of the rank as the prescribed truncation error ϵ_Λ tends to zero.

The experiments were done using the B-spline basis of the spline space $\bar{\mathbb{S}}$ which is used for interpolating the weight functions. Consequently, as discussed in Section 4.2.2, we measure the approximation error in the L^∞-norm using Lemma 3.

Table 4.1 shows the rank values that are needed to approximate the Jacobian determinant for varying accuracy ϵ_Λ, using the three different split directions. No projection error is present since the Jacobian determinant can be represented exactly in a tensor–product spline space $\bar{\mathbb{S}}$. In addition we provide information about the full tensor rank.

As analyzed in Section 4.5.3, the rank obtained by the partial decomposition in all directions never exceeds the tensor rank. The Jacobian determinant of model (a) has tensor rank 1 which is recognized by both methods. For the volumes (b) and (d), which were generated by approximate offsetting, the rank obtained by decomposing the parametrization with respect to the offset direction is the lowest one. This effect is particularly strong for (d).

We now proceed to the ranks of the elements of the matrix K, see (3.4). This matrix does not admit an exact representation in the spline space $\bar{\mathbb{S}}$ (except for trivial geometries). To minimize the influence of the spline approximation error, we choose the interpolation spaces $\bar{\mathbb{S}}$ such that the L^∞-norm of the interpolation

Table 4.1.: Ranks needed to approximate the Jacobian determinant m with varying accuracy. The bold font indicates the lowest ranks among the different split directions.

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	**1**	**1**	**1**
rank for split direction 2	**1**	**1**	**1**
rank for split direction 3	**1**	**1**	**1**
tensor rank	**1**	**1**	**1**

(a)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	1	2	3
rank for split direction 2	1	3	3
rank for split direction 3	**1**	**2**	**2**
tensor rank	1	3	4

(b)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	2	2	3
rank for split direction 2	**2**	**2**	**2**
rank for split direction 3	2	2	3
tensor rank	4	4	5

(c)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	6	7	7
rank for split direction 2	6	7	7
rank for split direction 3	**1**	**1**	**1**
tensor rank	6	7	7

(d)

error does not exceed $\epsilon_\Pi = 10^{-10}$.

Table 4.2 shows the *total rank* for different prescribed tolerances in the low-rank approximation, i.e. the sum of the ranks of the components of K. Again, we compare the rank values of the full decomposition with the partial decomposition in all three directions.

Table 4.2.: The total ranks needed to approximate the matrix-valued function K with varying accuracy. The bold font indicates the lowest ranks among the different split directions.

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	**5**	**5**	**5**
rank for split direction 2	**5**	**5**	**5**
rank for split direction 3	**5**	**5**	**5**
tensor rank	**5**	**5**	**5**

(a)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	13	22	31
rank for split direction 2	13	28	38
rank for split direction 3	**9**	**16**	**16**
tensor rank	13	29	42

(b)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	27	31	35
rank for split direction 2	18	22	24
rank for split direction 3	**18**	**18**	**19**
tensor rank	38	44	49

(c)

$\log_{10}(\epsilon_\Lambda)$	-4	-8	-10
rank for split direction 1	48	85	101
rank for split direction 2	48	85	101
rank for split direction 3	**5**	**5**	**8**
tensor rank	48	85	101

(d)

Since we consider the total rank, the difference in the rank values between the full and partial decompositions is more prominent in the case of the stiffness matrix. Similar to the case of the mass matrix, we observe that the splitting with respect to the direction of the sweep gives the lowest values of the total rank for volumes (b) and (d).

For the mass matrix, both methods result in quite stable rank values. For the stiffness matrix, however, the rank may increase as the error tolerance goes to zero. However, while the worst-case upper bound for the tensor rank is $\bar{n}^2$, it is only $\bar{n}$ for the rank generated by the partial decomposition.

In the following sections, when comparing PDM with FDM and with the element-wise Gauss quadrature, we always choose the split direction which provides the smallest rank value. This is motivated by the observation that the overall complexity is dominated by this rank.

4.6.2. Decomposition step

We investigate the computational costs of the decomposition step (HOSVD or SVD), in relation to the total cost of the matrix assembly. These costs are reported in Table 4.3 for the mass and the stiffness matrix.

Table 4.3.: Efficiency of the decomposition step when assembling the mass matrix (top) and the stiffness matrix (bottom) for the coarsest discretization.

mass matrix

Model	Disc. DOF	Intpl. DOF	HOSVD time(s)	Assembly time(s)	Ratio	SVD time(s)	Assembly time(s)	Ratio
(a)	$4 \times 4 \times 4$	$11 \times 11 \times 11$	$1.6 \cdot 10^{-3}$	$8.1 \cdot 10^{-3}$	20%	$2.7 \cdot 10^{-4}$	$8.0 \cdot 10^{-3}$	3%
(b)	$3 \times 3 \times 2$	$6 \times 6 \times 3$	$6.3 \cdot 10^{-4}$	$2.1 \cdot 10^{-3}$	30%	$7.5 \cdot 10^{-5}$	$1.5 \cdot 10^{-3}$	5%
(c)	$9 \times 2 \times 5$	$24 \times 3 \times 12$	$3.2 \cdot 10^{-3}$	$1.3 \cdot 10^{-2}$	25%	$6.1 \cdot 10^{-4}$	$1.0 \cdot 10^{-2}$	6%
(d)	$5 \times 5 \times 2$	$12 \times 12 \times 3$	$1.0 \cdot 10^{-3}$	$7.4 \cdot 10^{-3}$	14%	$9.5 \cdot 10^{-5}$	$4.8 \cdot 10^{-3}$	2%

stiffness matrix

Model	Disc. DOF	Intpl. DOF	HOSVD time(s)	Assembly time(s)	Ratio	SVD time(s)	Assembly time(s)	Ratio
(a)	$4 \times 4 \times 4$	$31 \times 31 \times 31$	$1.2 \cdot 10^{-1}$	$3.7 \cdot 10^{-1}$	32%	$5.5 \cdot 10^{-2}$	$2.8 \cdot 10^{-1}$	20%
(b)	$3 \times 3 \times 2$	$13 \times 13 \times 9$	$8.2 \cdot 10^{-3}$	$2.2 \cdot 10^{-2}$	37%	$2.0 \cdot 10^{-3}$	$1.4 \cdot 10^{-2}$	13%
(c)	$9 \times 2 \times 5$	$64 \times 12 \times 32$	$1.6 \cdot 10^{-1}$	$2.9 \cdot 10^{-1}$	55%	$7.4 \cdot 10^{-2}$	$3.8 \cdot 10^{-1}$	19%
(d)	$5 \times 5 \times 2$	$32 \times 32 \times 20$	$1.3 \cdot 10^{-1}$	$4.0 \cdot 10^{-1}$	33%	$4.1 \cdot 10^{-2}$	$3.1 \cdot 10^{-1}$	13%

We choose the projection spaces $\bar{\mathbb{S}}$ for the mass matrix and the stiffness matrix

such that the interpolation error accuracy is below 10^{-8}. The resulting numbers of degrees of freedom are reported in the third column of both tables. The same accuracy is used when performing the rank truncation.

We then compare the decomposition time with the total time needed for the matrix assembly, for the coarsest possible approximation as determined by the geometry mapping (reported in the second column of the tables). Consequently, the number $\bar{n}$ is always larger than n in this experiment. Even in this situation, the total cost is dominated by the assembly step.

More precisely, the tables report the decomposition time needed for both methods (columns 4 and 7), the associated assembly times (columns 5 and 8), and their ratio. For the partial decomposition, the decomposition time never contributes more than 20% to the total time, even for these very sparse discretizations. In addition we observe that the SVD is always significantly faster than HOSVD.

Even for this experiment, where $\bar{n} \geq n$, the decomposition step only takes up a relatively small part of the total computation time. As we will see in the next section, the assembly time increases linearly with the number of degrees of freedom of the discretization space, while the refinement of $\mathbb{S}$ has no effect on the decomposition. Indeed, the latter step only depends on the dimension of the projection space $\bar{\mathbb{S}}$. Thus, the contribution of the decomposition step becomes even less significant as n is increased.

4.6.3. Order of convergence

The consistency error in Strang's first lemma – and consequently the overall error of the isogeometric discretization – can be controlled by the combined error of spline projection and rank truncation, $\epsilon = \epsilon_\Pi + \epsilon_\Lambda$. Fig. 4.2 reports the convergence rates for the solution of the Poisson problem

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega, \\ u &= u_0 && \text{on } \partial\Omega, \end{aligned}$$

with known exact solution

$$u(x, y, z) = \sin(\pi x) \sin(\pi y) \sin(\pi z).$$

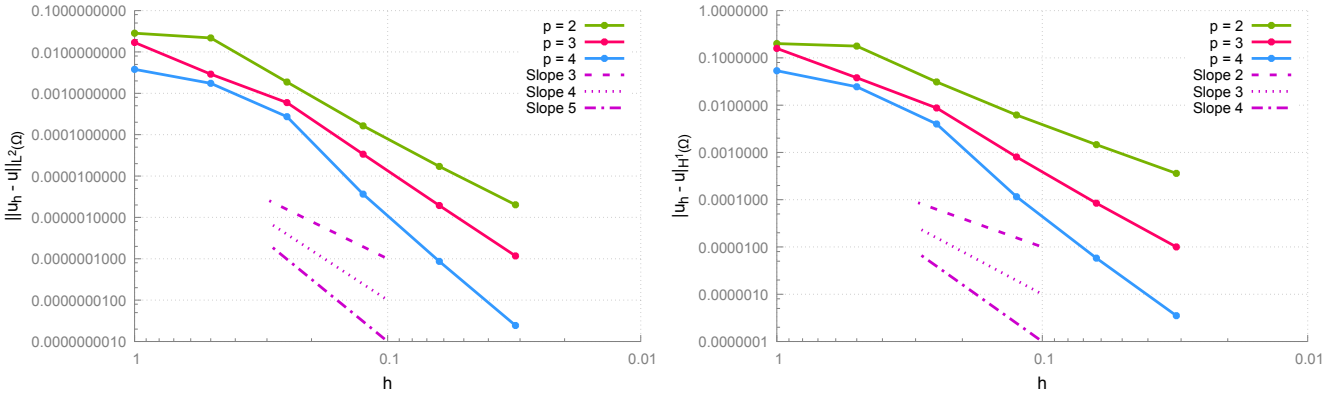


Figure 4.2.: Numerical errors for the solution to a Poisson problem on model (b) for degrees $p = 2, 3, 4$..

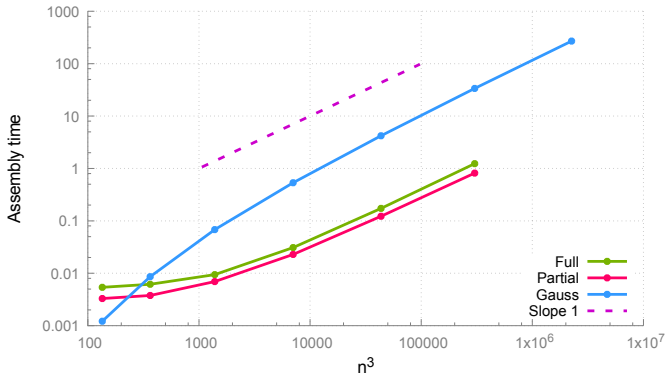
on the domain Ω represented by patch (b) in Fig. 4.1, using PDM. We consider discretizations of degree $p = 2, 3, 4$ and an error tolerance $\epsilon = 10^{-10}$, which results in rank-16 approximations of the stiffness matrices. These realize the optimal rates of convergence for the considered range of n, both with respect to the L^2 norm (left) and H^1 seminorm (right).

4.6.4. Computational complexity of matrix assembly

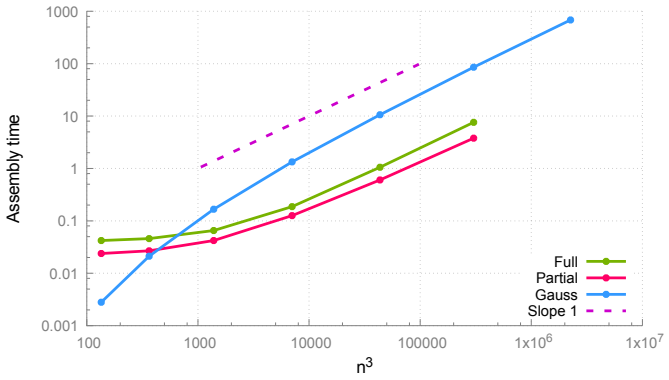
We investigate the dependence of the computation times on n (the number of degrees of freedom per direction used for the discretization) and p (the corresponding polynomial degrees). The experiments were performed on the computational domain represented by patch (c), using different polynomial degrees for the discretization. The overall accuracy was chosen to be $\epsilon = 10^{-8}$, and the degrees of the projection spline space $\bar{\mathbb{S}}$ were kept constant, $(5, 2, 5)$ in the mass case and $(11, 7, 11)$ in the stiffness case.

First we explore the dependence on the number n^3 of degrees of freedom. Figure 4.3 reports the computation times (including interpolation, decomposition, numerical quadrature and sum of Kronecker products) required for assembling the mass and stiffness matrices for various values of n and degrees $p = 2, 3, 4$.

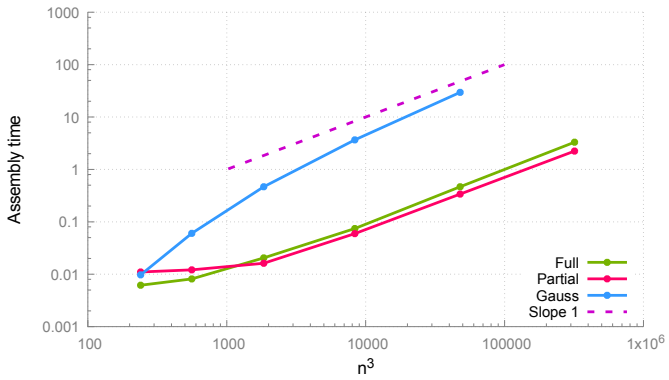
All three methods scale linearly with the dimension n^3 of the discretization space $\mathbb{S}$. Both decomposition-based methods are much faster than the element-wise Gauss



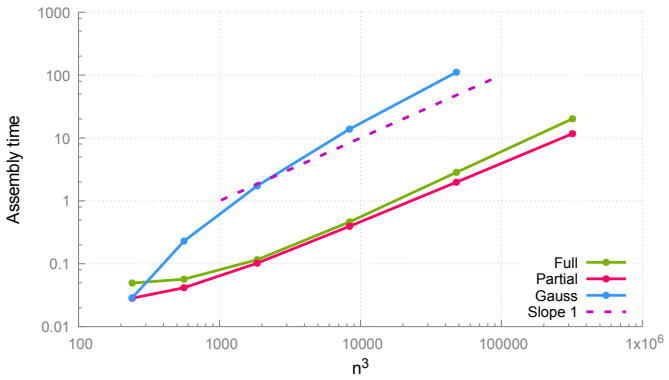
(a) Mass, $p = 2$



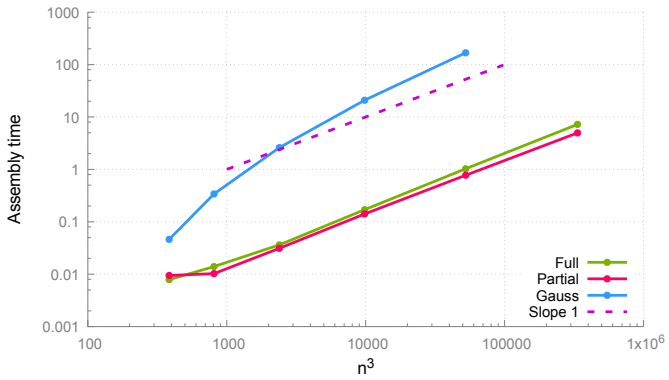
(b) Stiffness, $p = 2$



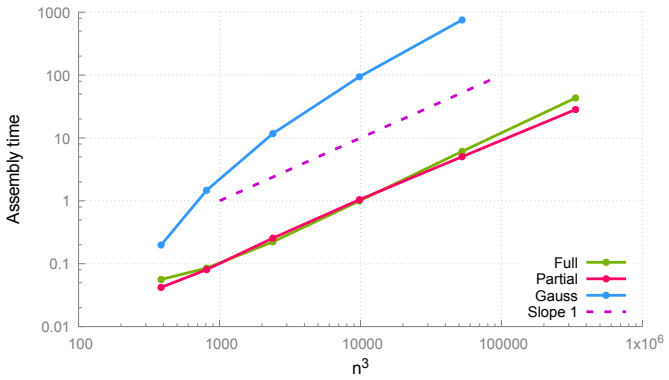
(c) Mass, $p = 3$



(d) Stiffness, $p = 3$



(e) Mass, $p = 4$



(f) Stiffness, $p = 4$

Figure 4.3.: n-dependence of the assembly times of the mass and stiffness matrices on model (c) for different polynomial degrees p.

quadrature, even for small polynomial degrees p. PDM is faster than FDM for sufficiently large values of n, since the rank is smaller. For small values of n and larger degrees p, however, the overall effort of PDM is dominated by the bivariate quadrature, and hence FDM performs slightly better.

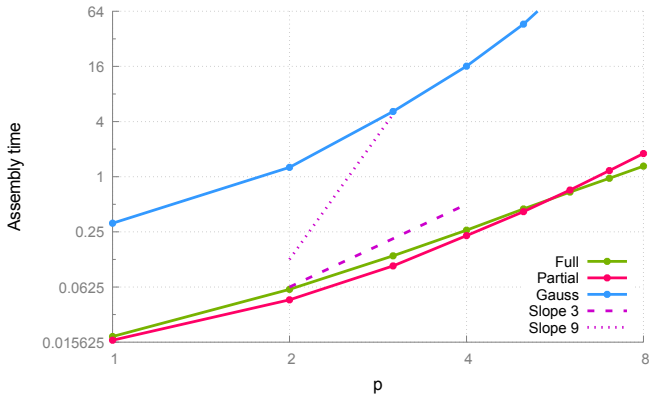
Now we continue with the dependence on the polynomial degree p. Figure 4.4 reports the computation times required for assembling the mass and stiffness matrices for three different values of n and degrees $p = 1, \dots, 8$.

These plots confirm the expected asymptotic scaling with p^3 for both of the decomposition-based methods, while the computational costs of the element-wise Gauss quadrature grow much faster. We also note that the complexity (of order 6 with respect to the degree) of the bivariate quadrature dominates the overall effort for large values of p, hence FDM becomes slightly faster than PDM. In fact, if n has lower order of magnitude than p^3, the overall complexity is dominated by the bivariate quadrature and FDM is potentially more efficient. Consequently, PDM and FDM are better suited for h- and p-refinement respectively.

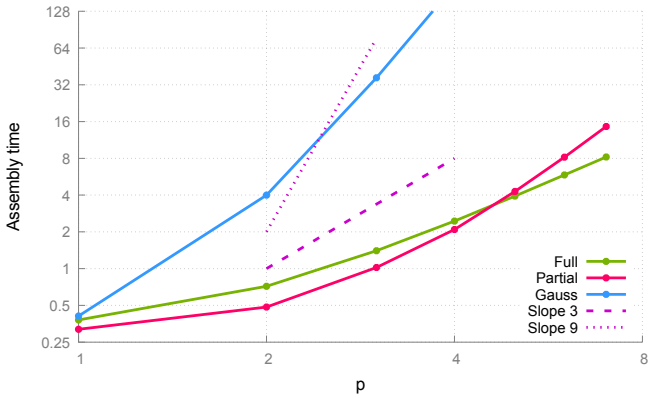
Finally we address the Kronecker format. For certain applications (for instance, when using iterative solvers that require only matrix-vector multiplications), it suffices to represent the matrices in Kronecker format (4.10). This can be achieved simply by omitting the last step of the algorithm. The computation time is then dominated by the Gauss quadrature needed when evaluating the bivariate and univariate integrals. Consequently, PDM and FDM have complexity $O(\rho n^2 p^6)$ and $O(\tilde{\rho} n p^3)$, respectively. This is confirmed by the numerical experiments reported in Fig. 4.5. However, the cost of matrix-vector multiplications using the Kronecker format grows linearly with the rank of the representation, and hence PDM (which typically generates a much lower rank value) has an advantage when considering the overall computational effort.

4.6.5. Worst-case comparison with FDM

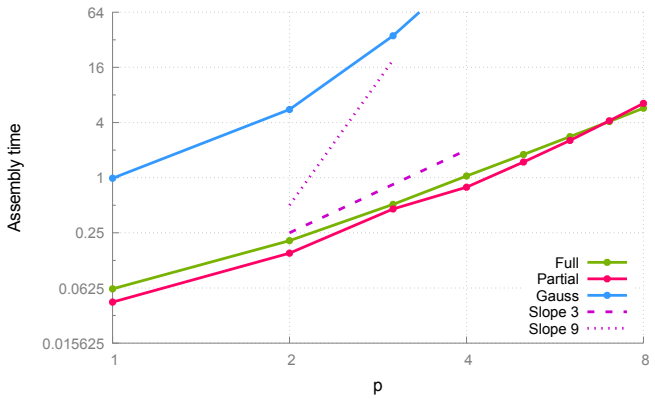
In Section 4.5.3 we concluded that the rank obtained by PDM is bounded by $O(\bar{n})$ and is always lower than the rank obtained by FDM which is bounded by $O(\bar{n}^2)$. This means that even for very irregular patches PDM will still have an advantage over the classical element-wise Gauss quadrature while FDM might perform poorly,



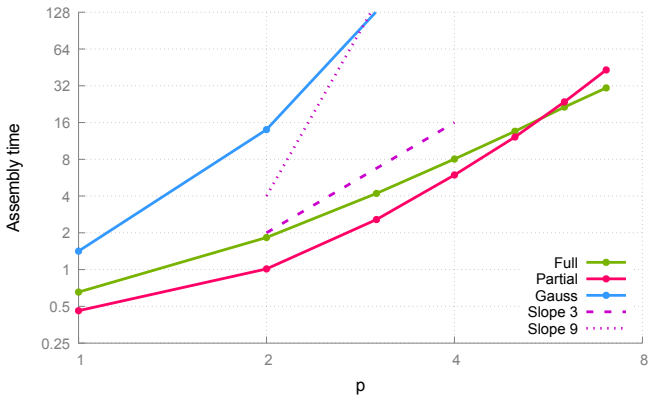
(a) Mass, $25 \times 25 \times 25$



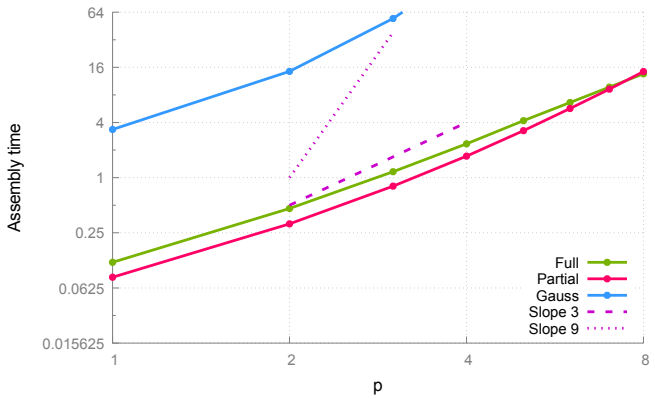
(b) Stiffness, $25 \times 25 \times 25$



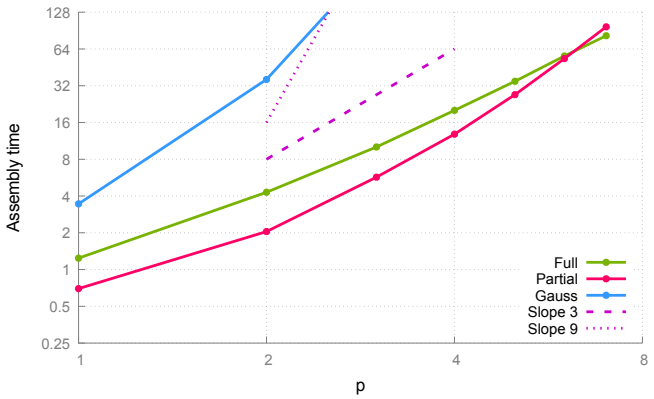
(c) Mass, $37 \times 37 \times 37$



(d) Stiffness, $37 \times 37 \times 37$

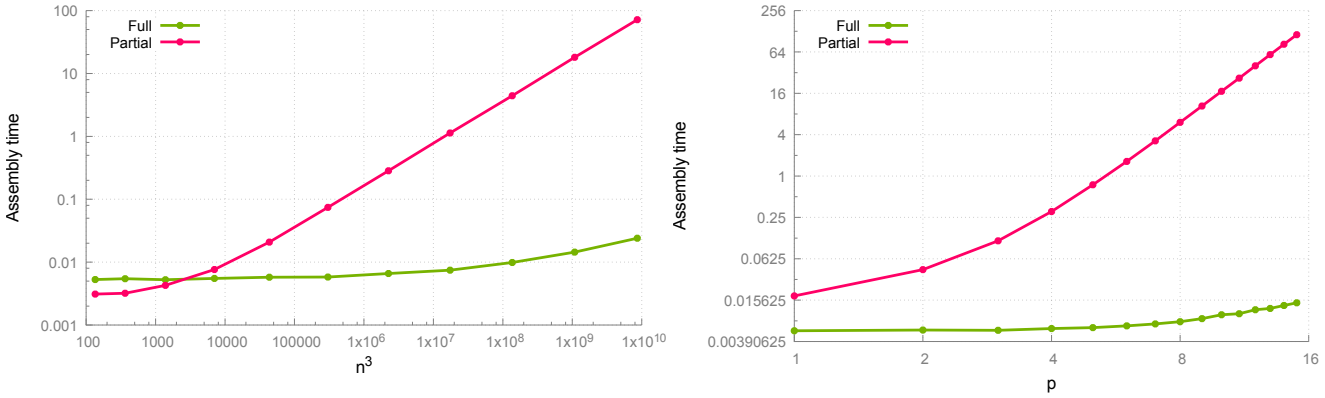


(e) Mass, $50 \times 50 \times 50$



(f) Stiffness, $50 \times 50 \times 50$

Figure 4.4.: p-dependence of the assembly times of the mass and stiffness matrices on model (c) for different values of n.



(a) n-dependence for $p = 2$

(b) p-dependence for $50 \times 50 \times 50$ DOF

Figure 4.5.: n- and p-dependence of the computation time for the Kronecker format of the mass matrix on model (c).

especially since the cost of the nonlinear optimization in the decomposition step increases with the rank.

In order to demonstrate this advantage experimentally, we perform another experiment on a patch which is expected to exhibit maximal rank for both methods. We use a cubic parametrization of a cube with two inner knots in each direction and disturb its control net using pseudo-random numbers. The displacement of each control point was bounded to maintain the regularity of the parametrization. Figure 4.6 shows the resulting perturbed cube. First we analyze the behavior of the ranks in both methods depending on the chosen tolerance ϵ_Λ. Figure 4.7 shows the resulting rank when approximating the Jacobian determinant for decreasing tolerance in both FDM and PDM. In this experiment we chose the spline space $\bar{\mathbb{S}}$ such that the Jacobian determinant is represented exactly, result in an interpolation space of dimensions $23 \times 23 \times 23$. This means that the maximum rank in PDM is 23 while the maximum rank in FDM is 529. The results comply with these bounds.

Second we compare the computation times for evaluating the mass matrix using FDM, PDM, and the classical element-wise Gauss quadrature on the perturbed cube. We choose the tolerance depending on the number of degrees of freedom of the discretization space such that it decays as fast as the theoretical discretization error of the L^2-projection, thereby maintaining the overall accuracy of the numerical simulation. Table 4.4 shows the resulting total computation times as well as the

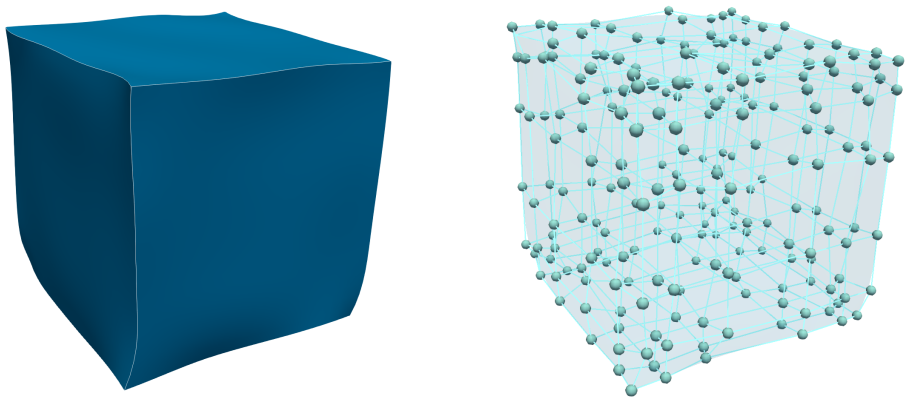


Figure 4.6.: The randomly perturbed cube (left) and its control grid (right), $n = 6$ and $p = 3$.

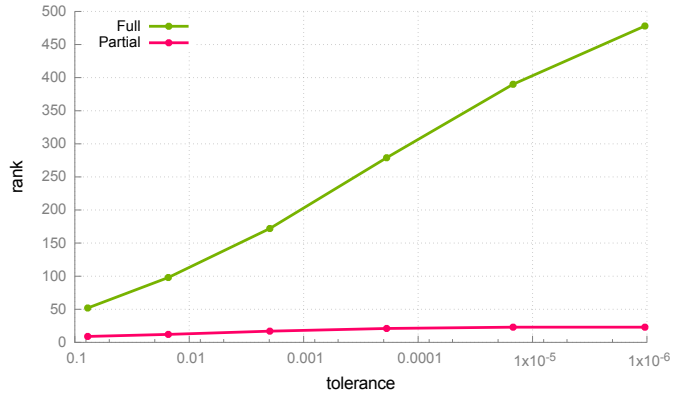


Figure 4.7.: Ranks of the Jacobian determinant of the perturbed cube

Table 4.4.: Computation times for assembling the mass matrix on the perturbed cube using FDM, PDM and Gauss quadrature.

		rank		decomposition time (s)		total time (s)		
#DOF	tolerance	FDM	PDM	FDM	PDM	FDM	PDM	Gauss
216	$7.7 \cdot 10^{-2}$	52	9	0.16	0.0017	0.32	0.16	**0.02**
729	$1.5 \cdot 10^{-2}$	98	12	1.84	0.0017	2.09	0.18	**0.13**
3375	$1.9 \cdot 10^{-3}$	172	17	27.36	0.0017	28.85	**0.37**	1.05
19683	$1.9 \cdot 10^{-4}$	279	21	250.67	0.0017	267.88	**1.56**	8.39
132651	$1.5 \cdot 10^{-5}$	390	23	896.87	0.0017	1083.72	**9.02**	67.75
970299	$1.0 \cdot 10^{-6}$	478	23	2288.38	0.0017	4526.56	**58.35**	566.81

time needed for the (full or partial) tensor decomposition. We observe that the decomposition time in PDM is constant while the decomposition time in FDM grows significantly as the rank increases. The comparison with the classical element-wise Gauss quadrature shows that PDM still achieves a large speed-up even in this worst-case example. On the other hand, FDM performs worse than Gauss quadrature in this case. This is not in contradiction to the complexity result presented in Section 4.5; a higher degree p would be required to realize the asymptotic advantage of FDM with respect to Gauss quadrature.

5. Partial tensor decomposition in space-time methods

In Chapter 4 we focused on the computation of the mass and stiffness matrix in elliptic partial differential equations. Another natural application of the partial tensor decomposition method is the evaluation of the system matrix appearing in time-dependent equations that are discretized using isogeometric space-time methods. In this type of method, the solution is approximated in a function space over the complete space-time cylinder consisting of product of a spatial domain and the time interval.

We start by introducing our model problem, its discrete variational form and the error estimates for spline discretizations in Section 5.1. In Section 5.2 we achieve the efficient computation of the system matrix, first for a constant diffusion coefficient and then, using the partial tensor decomposition method from the previous chapter, for varying diffusion. We analyze the complexity in Section 5.3 and present numerical experiments in Section 5.4.

5.1. Parabolic equation with varying coefficients

We present an isogeometric space-time formulation for a parabolic equation with coefficients that depend on both space and time. In particular, the method does not involve any time-stepping but treats space and time equally as a four-dimensional domain. We use this problem as a model problem for the efficient matrix assembly by partial tensor decomposition and only cite the theoretical results for the weak problem and its discretized analogue without deriving them. The details can be found in [66, Section 2].

5.1.1. Model problem and isogeometric space-time formulation

We consider a physical domain $\Omega \subset \mathbb{R}^d$, where $d = 2, 3$, and the space-time cylinder $Q = \Omega \times [0, T]$ for some final time $T > 0$. The initial and final time slices are denoted by $\Sigma_0 = \Omega \times \{0\}$ and $\Sigma_T = \Omega \times \{0\}$. We assume that Ω is parameterized by a regular diffeomorphism (geometry mapping)

$$G : \hat{\Omega} \longrightarrow \Omega$$

over the parametric domain $\hat{\Omega} = [0, 1]^d$. We obtain a parameterization

$$\mathbf{\Phi} : \hat{Q} \longrightarrow Q$$

of the space-time cylinder over the parametric domain $\hat{Q} = [0, 1]^{d+1}$ by lifting G linearly, i.e.,

$$\mathbf{\Phi}(\hat{x}_1, \ldots, \hat{x}_d, \hat{t}) = \Big(G(\hat{x}_1, \ldots, \hat{x}_d), \hat{t}\, T\Big), \tag{5.1}$$

where instead of $\hat{x}_{d+1}$ we use $\hat{t}$ to denote the time variable.

Let $\rho : Q \to \mathbb{R}$ be a smooth function that fulfills

$$\rho_{\max} \geq \rho(x, t) \geq \rho_{\min} > 0$$

for positive constants $\rho_{\min}$ and $\rho_{\max}$. The goal is to approximate the solution to the initial value problem

$$\begin{cases} \partial_t u - \operatorname{div}_x(\rho(x,t)\nabla_x u) &= f \text{ in } Q \\ u(x,t) &= 0 \text{ on } \Sigma, \\ u(\cdot, 0) &= u_0 \text{ on } \Sigma_0, \end{cases} \tag{5.2}$$

where $f \in L^2(Q), u_0 \in L^2(\Omega)$.

Using the standard approach of multiplying this equation with a test function and applying integration by parts one arrives at the weak space-time formulation of the problem: Find $u \in H_0^{1,0}(Q)$ that fulfills

$$a(u, v) = \ell(v), \quad \text{for all} \;\; v \in H_{0,\tilde{0}}^{1,1}(Q), \tag{5.3}$$

where the bilinear form is given by

$$a(u,v) = -\int_Q u(x,t)\partial_t v(x,t)\ \mathrm{d}x\ \mathrm{d}t + \int_Q \rho(x,t)\,\nabla_x u(x,t)\ \cdot\ \nabla_x v(x,t)\ \mathrm{d}x\ \mathrm{d}t$$

and the linear right-hand side is given by

$$\ell(v) = \int_Q f(x,t)v(x,t)\ \mathrm{d}x\ \mathrm{d}t + \int_\Omega u_0(x)v(x,0)\ \mathrm{d}x.$$

Here, the employed Sobolev spaces are defined as

$$H_0^{1,0}(Q) = \{v \in L^2(Q) : \nabla_x v \in [L^2(Q)]^d,\ v = 0\ \text{ on }\ \Sigma\}$$

and

$$H_{0,\bar{0}}^{1,1}(Q) = \{v \in L^2(Q) : \nabla_x v \in [L^2(Q)]^d,\ \partial_t v \in L^2(Q), v = 0\ \text{ on }\ \Sigma,\ v = 0\ \text{ on }\ \Sigma_T\}.$$

5.1.2. Isogeometric discretization

Similar to the case of elliptic partial differential equations that was presented in Section 3.2, the solution to (5.3) is approximated using a tensor-product spline space

$$\mathbb{S} = \bigotimes_{k=1}^{d+1} S_{\boldsymbol{\tau}_k}^{p_k}$$

on the parameter domain, where τ_k are the knot vectors and p_k the polynomial degrees for each direction. In particular, the discretization space for the problem on the physical domain is chosen as the space of *isogeometric functions*

$$V_h = \mathrm{span}\{\beta_{\boldsymbol{i}} = \hat{\beta}_{\boldsymbol{i}} \circ \Phi^{-1} : \boldsymbol{i} \in \{1,\dots,n_1\} \times \dots \times \{1,\dots,n_{d+1}\}\}.$$

In order to incorporate the boundary conditions and initial conditions, we define

$$V_{0h} = \{v_h \in V_h : v_h|_{\Sigma \cup \Sigma_0} = 0\}.$$

However, in contrast to the discrete variational form for the elliptic PDE, a non-symmetric bilinear form has to be used in order to arrive at a stable discrete

formulation. It is derived by multiplying (5.2) with a discrete test function $w_h = v_h + \theta h \frac{\partial v_h}{\partial t}$, where $v_h \in V_{0h}$. Here, $\theta > 0$ is a given positive constant.

The discrete weak problem reads as follows: Find $u_h \in V_{0h}$, such that

$$a_h(u_h, v_h) = l_h(v_h), \quad \text{for all} \;\; v_h \in V_{0h}, \tag{5.4}$$

where

$$\begin{aligned} &a_h(u_h, v_h) \\ &= \int_Q \partial_t u_h \, (v_h + \theta \, h \partial_t v_h) + \rho(x,t) \Big(\nabla_x u_h \cdot \nabla_x v_h + \theta \, h \nabla_x u_h \cdot \nabla_x \partial_t v_h \Big) \, \mathrm{d}x \, \mathrm{d}t \end{aligned} \tag{5.5}$$

and

$$l_h(v_h) = \int_Q f \, (v_h + \theta \, h \partial_t v_h) \, \mathrm{d}x \, \mathrm{d}t.$$

Evaluating the bilinear and linear form in (5.4) on the basis functions $\beta_{\boldsymbol{i}} = \hat{\beta}_{\boldsymbol{i}} \circ \Phi^{-1}$ of V_h results in a linear system

$$K^T u = F.$$

Here, the entries of K are of the form $K_{\boldsymbol{ij}} = a_h(\beta_{\boldsymbol{i}}, \beta_{\boldsymbol{j}})$, while the entries of the right-hand side are given by $F_{\boldsymbol{i}} = l_h(\beta_{\boldsymbol{i}})$.

The details for the derivation and the analysis of the discrete formulation are not part of this thesis and can be found in [66, Section 2]. There, it is shown that with respect to the appropriate norm,

$$\|v\|_h = \left(\|\rho^{\frac{1}{2}} \, \nabla_x v\|^2_{L^2(Q)} + \theta \, h \, \|\partial_t v\|^2_{L^2(Q)} + \frac{1}{2} \|v\|^2_{L^2(\Sigma_T)} \right)^{\frac{1}{2}}, \tag{5.6}$$

the solution of (5.4) in V_{0h} converges with order p to the solution of (5.3). It is assumed that $p = p_1 = \ldots p_{d+1}$ and that the solution of (5.3) is smooth enough, see [66, Theorem 5].

5.2. Splitting the integration directions

We apply the partial tensor decomposition method to the parabolic PDE with varying diffusion presented in Section 5.1. The goal of this application is to decouple the integration in space and time and thereby improving the computational complexity of assembling the system matrix based on evaluating the discrete bilinear form (5.5) on the basis functions.

Analogously to the elliptic case, the integrals appearing in the system matrix and right-hand side of the discrete system are transformed to the parametric domain $\hat{Q} = [0,1]^{d+1}$. In Section 5.2.1 we show that if the diffusion coefficient ρ in (5.2) is constant, e.g. $\rho \equiv 1$, then the decoupling of the integration in time and in space follows naturally from the tensor-product structure of the B-Spline basis and the parameterization (5.1). Afterwards, we consider the case of a general diffusion coefficient and use the partial tensor decomposition method from Chapter 4 to decouple the assembly. The resulting decomposition of the system matrix is derived in Section 5.2.2.

5.2.1. Fast assembly for constant diffusion coefficients

In the case of a constant diffusion coefficient $\rho \equiv 1$ we can exploit the tensor product structure of the B-Spline basis and the corresponding structure of the parameterization (5.1) directly and perform the integration in space and time separately without performing a (partial) low-rank approximation. For assembling the system matrix corresponding to the discrete weak formulation (5.4), we need to evaluate the bilinear form defined in (5.5), on the basis functions of the discretization space. Using the parameterization (5.1), we transform each appearing integral over the $(d+1)$-dimensional space-time domain Q to the parametric domain $\hat{Q} = [0,1]^{d+1}$. As a consequence of the B-Spline basis' tensor-product structure, each resulting $(d+1)$-variate integral is split into a product of a d-variate integral over the spatial parametric domain and a univariate integral over the unit interval.

Let $\boldsymbol{i} = (i_1, \ldots, i_d, i_{d+1})$ and $\boldsymbol{j} = (j_1, \ldots, j_d, j_{d+1})$ be two multi-indices. In the rest of this section, we consider each of the four terms in the representation of $a_h(\beta_{\boldsymbol{i}}, \beta_{\boldsymbol{j}})$ in (5.5) separately.

For the first term, the transformation of the integral yields

$$\begin{aligned}\int_Q \partial_t \beta_{\boldsymbol{i}}\, \beta_{\boldsymbol{j}}\ \mathrm{d}x\ \mathrm{d}t &= \int_{\hat{Q}} |\det J_G| \partial_{\hat{t}} \hat{\beta}_{\boldsymbol{i}} \hat{\beta}_{\boldsymbol{j}}\, d\hat{x}\, d\hat{t} \\ &= \int_{(0,1)^d} |\det J_G| \hat{\beta}_{i_1 \dots i_d} \hat{\beta}_{j_1 \dots j_d}\, d\hat{x} \cdot \int_0^1 \partial_{\hat{t}} \hat{\beta}_{i_{d+1}} \hat{\beta}_{j_{d+1}}\, d\hat{t},\end{aligned} \tag{5.7}$$

where J_G is the Jacobian of the spatial geometry mapping G. For the second term we arrive at

$$\begin{aligned}\theta h \int_Q \partial_t \beta_{\boldsymbol{i}}\, \partial_t \beta_{\boldsymbol{j}}\ \mathrm{d}x\ \mathrm{d}t &= \theta \frac{h}{T} \int_{\hat{Q}} |\det J_G| \partial_{\hat{t}} \hat{\beta}_{\boldsymbol{i}} \partial_{\hat{t}} \hat{\beta}_{\boldsymbol{j}}\, d\hat{x}\, d\hat{t} \\ &= \theta \frac{h}{T} \int_{(0,1)^d} |\det J_G| \hat{\beta}_{i_1 \dots i_d} \hat{\beta}_{j_1 \dots j_d}\, d\hat{x} \cdot \int_0^1 \partial_{\hat{t}} \hat{\beta}_{i_{d+1}} \partial_{\hat{t}} \hat{\beta}_{j_{d+1}}\, d\hat{t}.\end{aligned} \tag{5.8}$$

Transforming the third term leads to

$$\begin{aligned}\int_Q \nabla_x \beta_{\boldsymbol{i}} \nabla_x \beta_{\boldsymbol{j}}\ \mathrm{d}x\ \mathrm{d}t &= T \int_{\hat{Q}} |\det J_G| \nabla_{\hat{x}} \hat{\beta}_{\boldsymbol{i}} J_G^{-1} J_G^{-\top} \nabla_{\hat{x}} \hat{\beta}_{\boldsymbol{j}}\, d\hat{x}\, d\hat{t} \\ &= T \int_{(0,1)^d} |\det J_G| \nabla_{\hat{x}} \hat{\beta}_{i_1 \dots i_d} J_G^{-1} J_G^{-\top} \nabla_{\hat{x}} \hat{\beta}_{\boldsymbol{j}}\, d\hat{x} \cdot \int_0^1 \hat{\beta}_{i_{d+1}} \hat{\beta}_{j_{d+1}}\, d\hat{t}.\end{aligned} \tag{5.9}$$

Finally, the fourth integral becomes

$$\begin{aligned}\theta h \int_Q \nabla_x \beta_{\boldsymbol{i}} \cdot \nabla_x \partial_t \beta_{\boldsymbol{j}}\ \mathrm{d}x\ \mathrm{d}t &= \theta h \int_{\hat{Q}} |\det J_G| \nabla_{\hat{x}} \hat{\beta}_{\boldsymbol{i}} J_G^{-1} J_G^{-\top} \nabla_{\hat{x}} \hat{\beta}_{\boldsymbol{j}}\, d\hat{x}\, d\hat{t} \\ &= \theta h \int_{(0,1)^d} |\det J_G| \nabla_{\hat{x}} \hat{\beta}_{i_1 \dots i_d} J_G^{-1} J_G^{-\top} \nabla_{\hat{x}} \hat{\beta}_{j_1 \dots j_d}\, d\hat{x} \cdot \int_0^1 \hat{\beta}_{i_{d+1}} \partial_{\hat{t}} \hat{\beta}_{j_{d+1}}\, d\hat{t}.\end{aligned} \tag{5.10}$$

We observe that these representations only consist of entries of the stiffness and mass matrices for space and time, as well as those of the matrix containing the mixed time derivatives.

The decomposition of the integrals implies that we can write the system matrix $(K_{ij}) = (a_h(\beta_{\boldsymbol{j}}, \beta_{\boldsymbol{i}}))$ as the sum of Kronecker products

$$K = X_1 \otimes (Y_1 + Y_2) + X_2 \otimes \left(Y_3 + Y_1^\top\right), \tag{5.11}$$

where X_k are $n_1 \cdots n_d \times n_1 \cdots n_d$-matrices containing the d-variate integrals given in (5.7)-(5.10) and Y_k are $n_{d+1} \times n_{d+1}$-matrices containing the corresponding univariate

integrals. This representation (5.11) is also called the *Kronecker format*. We define the Kronecker rank of a matrix to be the number of summands in the Kronecker format, that is, in our case the Kronecker rank of K is 2.

5.2.2. Fast assembly for space-time dependent diffusion

Next, we consider the case where the diffusion coefficient ρ is a smooth function depending on both x and t. Now, the terms (5.9) and (5.10) no longer decompose directly. In order to decouple integration in space and time in this case, we use the partial tensor decomposition method to decompose the parametric diffusion coefficient $\hat{\rho}(\hat{x},\hat{t}) = (\rho \circ \mathbf{\Phi})(\hat{x},\hat{t})$ into d-variate and univariate functions by projecting into a spline space and computing the singular value decomposition of the coefficient tensor. This results in an approximation

$$\hat{\rho}(\hat{x},\hat{t}) \approx \sum_{r=1}^{R} \mathcal{U}_r(\hat{x})\mathcal{V}_r(\hat{t}),$$

where $\mathcal{U}$ and $\mathcal{V}$ are d-variate and univariate spline functions respectively and R is the smallest rank that guarantees a given error tolerance.

This decomposition of $\hat{\rho}$ leads to a decomposition of the $(d+1)$-variate integrals. In particular, the third term in (5.5) becomes

$$\int_Q \rho(x,t)\nabla_x \beta_{\boldsymbol{i}} \nabla_x \beta_{\boldsymbol{j}} \,\mathrm{d}x\,\mathrm{d}t \approx \tag{5.12a}$$
$$T\sum_{r=1}^{R} \int_{(0,1)^d} |\det J_G| \mathcal{U}_r(\hat{x}) \nabla_{\hat{x}} \hat{\beta}_{i_1\dots i_d} J_G^{-1} J_G^{-\top} \nabla_{\hat{x}} \hat{\beta}_{\boldsymbol{j}} \,d\hat{x} \int_0^1 \mathcal{V}_r(\hat{t}) \hat{\beta}_{i_{d+1}} \hat{\beta}_{j_{d+1}} \,d\hat{t},$$

while the fourth term becomes

$$\theta h \int_Q \rho(x,t) \nabla_x \beta_{\boldsymbol{i}} \cdot \nabla_x \partial_t \beta_{\boldsymbol{j}} \,\mathrm{d}x\,\mathrm{d}t \approx \tag{5.12b}$$
$$\theta h \sum_{r=1}^{R} \int_{(0,1)^d} |\det J_G| \mathcal{U}_r(\hat{x}) \nabla_{\hat{x}} \hat{\beta}_{i_1\dots i_d} J_G^{-1} J_G^{-\top} \nabla_{\hat{x}} \hat{\beta}_{j_1\dots j_d} \,d\hat{x} \int_0^1 \mathcal{V}_r(\hat{t}) \hat{\beta}_{i_{d+1}} \partial_{\hat{t}} \hat{\beta}_{j_{d+1}} \,d\hat{t}.$$

We arrive at a Kronecker format representation of rank $R+1$ of the system matrix:

$$K \approx X_1 \otimes (Y_1 + Y_2) + \sum_{r=1}^{R} U^r \otimes (V_1^r + V_2^r), \tag{5.13}$$

where the matrices X_1, Y_1, Y_2 are defined as in (5.11) and U^r, V_1^r, V_2^r are the matrices corresponding to (5.12).

If the diffusion coefficient is matrix-valued, let us say ρ_{pq}, then the same method can be applied by decomposing each of the components of the matrix. In this case the rank R in the Kronecker format is the total rank, i.e. the sum of the ranks of all components.

5.3. Computational complexity

As we did in Section 4.5, we assume that the degrees of freedom and polynomial degrees in each direction $\hat{x}_k$, $k = 1, \ldots, d+1$ are the same, i.e., $n = n_1 = \ldots = n_{d+1}$ and $p = p_1 = \ldots = p_{d+1}$. We apply the analysis from the previous sections to the application of PDM to the isogeometric discretization of the parabolic equation with varying coefficients.

The complexity of assembling the system matrix is bounded from below by the number of its non-zeros, which is $O(n^{d+1}p^{d+1})$. The classical assembly method using element-wise Gauss quadrature rules has complexity $O(n^{d+1}p^{3(d+1)})$.

In the proposed method we compute d-variate and univariate integrals by element-wise Gauss quadrature exploiting the decomposition of the integrals. The complexity of computing each matrix X_1, U^r is thus $O(n^d p^{3d})$ while the complexity of computing the matrices Y_1, Y_2, V^r is clearly dominated by this. Thus, the complexity of the quadrature step is

$$O(Rn^d p^{3d}).$$

Generating the global matrix K by computing the Kronecker product (5.13) then costs

$$O(Rn^{d+1}p^{d+1}).$$

This shows us that it depends on the dimension d which step of the assembly

method is dominating in terms of computational complexity. The overall complexity is either dominated by the d-variate quadrature or by the final step of summing up all the Kronecker products in (5.13). Since usually $n \gg p$, for $d = 2$ the complexity of the sum of Kronecker products (5.13) is dominating while for $d = 3$, the complexity of the quadrature step dominates. Thus, intuitively speaking, we are able to perform the matrix assembly for a four-dimensional space-time problem for the prize of a matrix assembly for a three-dimensional problem. For a three-dimensional space-time problem (i.e. with a two-dimensional spatial domain), the speed-up corresponds to the speedup in the case of three-dimensional elliptic problems analyzed in Section 4.5.

5.4. Numerical experiments

We show our numerical experiments for the application of the partial tensor decomposition method to the parabolic space-time problem, both with constant diffusion and with a diffusion coefficient that depends on space and time. First, we consider the three-dimensional case and then we continue with the four-dimensional case.

We perform each experiment on a single patch. For a multi-patch domain one can apply continuous or discontinuous discretization techniques and ultimately treat the problem patch-wise. Hence the methods developed in this work can be applied for handling the resulting local problems. For instance, the isogeometric tearing and interconnecting (IETI) method from [52] can be applied if the multi-patch discretization is continuous. Otherwise a discontinuous Galerkin approach can be used as is described in [37] for constant coefficient $\rho = 1$. The method produces a block-bidiagonal system matrix which can be solved sequentially.

For solving the linear system, a parallel GMRES solver from Trilinos [26] is used with tolerance set to 10^{-8} and Krylov subspace dimension chosen as 200.

In the experiments we analyze the convergence of the overall approximation error as well as the computational complexity. As a benchmark we compare with the classical element-wise Gauss quadrature.

First example, $Q \subset \mathbb{R}^{2+1}$. For our first numerical example, the space-time domain Q is the quarter annulus in space extended into the unit interval in time. We consider the function

$$u(x_1, x_2, t) = (\cos(2\pi(x-y)) - \cos(2\pi(x+y)))\,\sin(2\pi t) \tag{5.14}$$

to be the exact solution of the problem. The right-hand-side f, the boundary data and the initial data are computed accordingly. Besides the system matrix K, we also compute the load vector using the partial tensor decomposition method. The parameter θ is set to 1.

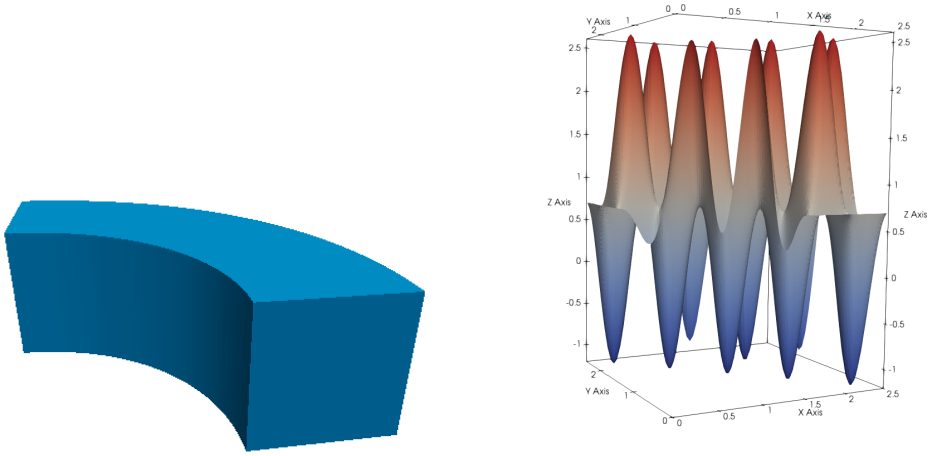


Figure 5.1.: The space-time domain for $d = 2$ and a time slice of the exact solution at $t = 0.7$. On the left, the spatial domain is an annulus and the time direction is parallel to the z-axis.

Table 5.1 shows the error in the appropriate norm (5.6) as well as the computation times for assembling the matrix in the case of a constant diffusion coefficient $\rho \equiv 1$. We observe that our method results in the optimal convergence rate from [66, Theorem 5] as presented in Section 5.1.2 and that the assembly times given in last column are very low compared to the size of the system.

Next, we study the behavior of the method when we have a space-time dependent diffusion coefficient. We consider again the same exact solution given in (5.14), but

	h	#DOF	error	error rate	assembly time
$p = 3$	0.25	343	8.30949		0.0061s
	0.125	1331	5.09944	0.70	0.0052s
	0.0625	6859	0.492096	3.37	0.0259s
	0.03125	42875	0.034556	3.83	0.0926s
	0.01562	300763	0.00392612	3.14	0.374241s
$p = 4$	0.25	512	7.97706		0.0111s
	0.125	1728	5.78954	0.46	0.0613s
	0.0625	8000	0.2335	4.63	0.4567s
	0.03125	46656	0.00601485	5.28	1.9032s
	0.01562	314432	0.000286093	4.39	11.2130s

Table 5.1.: 3D experiment on the example space-time domain (see Fig. 5.1) with a constant diffusion coefficient.

the diffusion coefficient is defined as

$$\rho(x_1, x_2, t) = (x_1 - 4)^2(x_2 - 4)^2(t - 4)^2 + (x_1 - 4)^4(x_2 - 4)^4(t - 4)^4. \tag{5.15}$$

As tolerance for the projection and truncation error in the low-rank approximation we chose $\epsilon = 10^{-6}$ which results in an approximation of rank 2. Table 5.2 shows that

	h	#DOF	error	error rate	assembly time
$p = 3$	0.25	343	8.40222		0.0425s
	0.125	1331	5.11743	0.72	0.0250s
	0.0625	6859	0.492269	3.38	0.1312s
	0.03125	42875	0.034559	3.83	0.9555s
	0.01562	300763	0.00366286	3.24	6.2758s
$p = 4$	0.25	512	8.03895		0.01667s
	0.125	1728	5.81114	0.47	0.1474s
	0.0625	8000	0.233569	4.64	0.2993s
	0.03125	46656	0.00601493	5.28	1.9955s
	0.01562	314432	0.000293	4.36	16.8282s

Table 5.2.: 3D experiment on the example space-time domain (see Fig. 5.1) with a space-time dependent diffusion coefficient (5.15).

the error estimates are still upheld in this case and that the assembly is performed

in a very short time.

For $p \geq 5$, we only show the computation times of the assembly, because the resulting systems are not only non-symmetric but also quite large and dense, and therefore very hard to solve. Figure 5.2 shows the dependence of the computation times on the number of degrees of freedom for the 3D problem for our method and for the classical element-wise Gauss quadrature. As it can be seen in the presented computation times, the assembly method using partial low-rank tensor approximation is very efficient. In particular, it outperforms a classical element-wise Gauss rule approach by far. This coincides with the results from Section4.6.4 for the mass and stiffness matrix.

In some cases it can be beneficial to avoid evaluating the sum of Kronecker products in (5.13) by using the *Kronecker format* representation directly. Since matrix–vector multiplication can be implemented easily for a matrix in this format, it can be used for solving the system iteratively. The main advantage is the reduction of the needed memory for storing the system matrix which allows us to assemble up to a very large number of degrees of freedom. The assembly is also very fast, since we only have to compute bivariate and univariate integrals. Figure 5.3 shows the computation times for the bivariate and univariate integrals in the matrices of the right hand side in (5.13). The maximum number of degrees of freedom that were computed in this experiment are over 136 million for $p = 3$.

Second example, $Q \subset \mathbb{R}^{3+1}$. In this example the space-time cylinder Q is the product of a volumetric shell shape (see Fig. 5.4) in space and the unit interval in time. The diffusion coefficient is chosen to be

$$\begin{aligned}\rho(x_1, x_2, x_3, t) =&(x_1 - 4)^2(x_2 - 4)^2(x_3 - 4)^2(t - 4)^2 + \\ &+(x_1 - 4)^4(x_2 - 4)^4(x_3 - 4)^4(t - 4)^4. \end{aligned} \tag{5.16}$$

We focus on the computation times for matrix assembly. Figure 5.5 shows the dependence of the assembly time on the number of degrees of freedom for the $\rho(x_1, x_2, x_3, t)$ given in (5.16). We observe that our method leads to a large speed-up compared to the classical element-wise Gauss quadrature.

We note that in the four-dimensional case the overall complexity is no longer

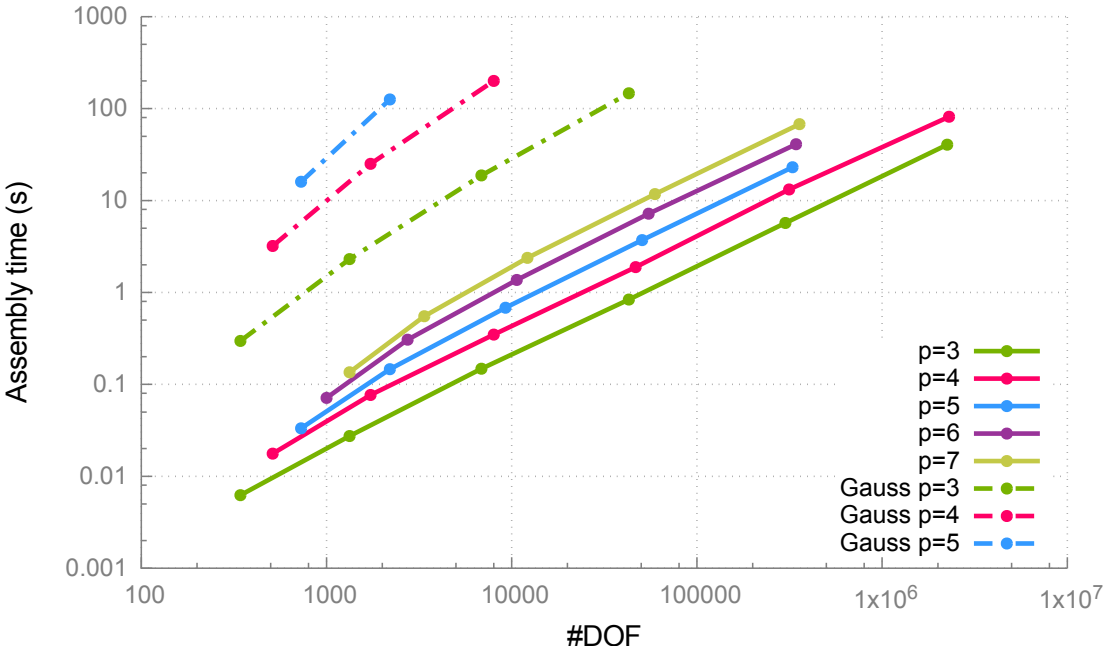


Figure 5.2.: Computation times for assembling the *global system matrix* on the three-dimensional example domain (Fig. 5.1) for the space-time dependent diffusion coefficient (5.15).

dominated by the sum of Kronecker products (5.13) but by the trivariate quadrature. For this reason, the computation of the matrix in Kronecker format is no longer significantly faster than the computation of the global matrix by performing the sum of Kronecker products in (5.13). However, the advantages stemming from the reduction in memory still apply to the four-dimensional case and we can assemble up to a much higher number of degrees of freedom using the same amount of memory. Figure 5.6 shows the computation times for the system matrix in Kronecker format. They are dominated by the trivariate spatial integrals. For $p = 2$ we assembled for a maximum number of about 19 million degrees of freedom.

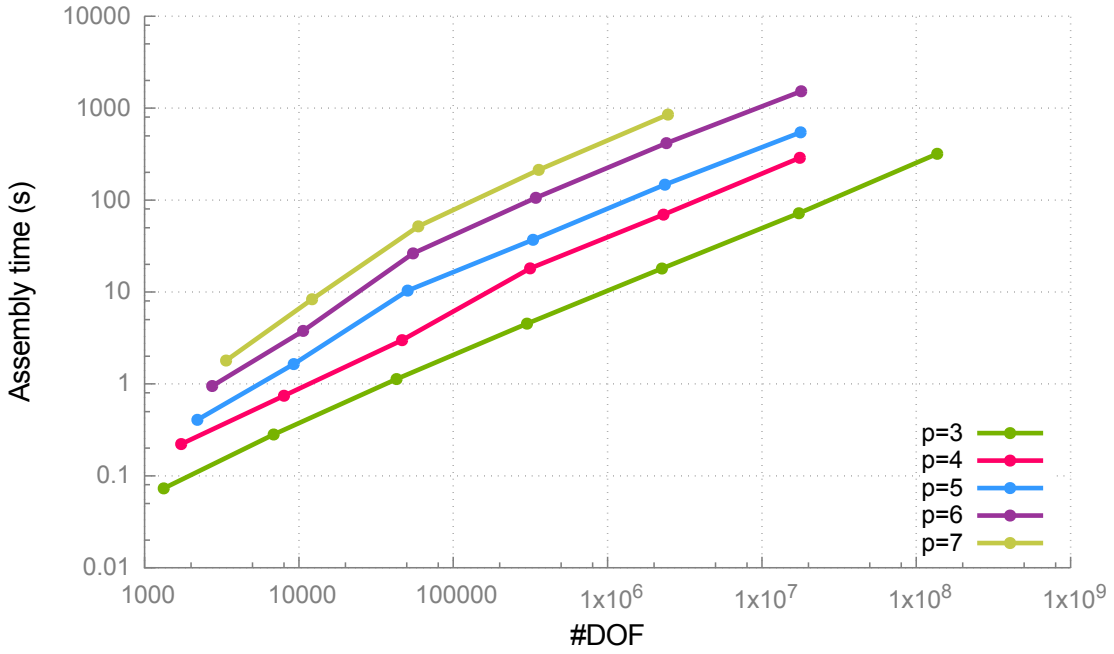


Figure 5.3.: Computation times for assembling the *Kronecker format* on the three-dimensional example domain (Fig. 5.1) for the space-time dependent diffusion coefficient (5.15).

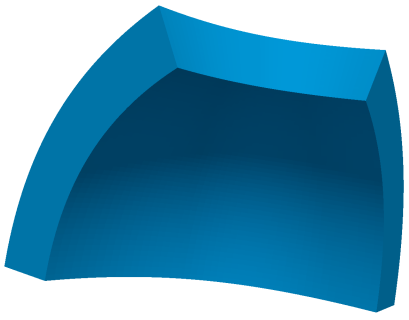


Figure 5.4.: The volumetric shell shape of Example 2.

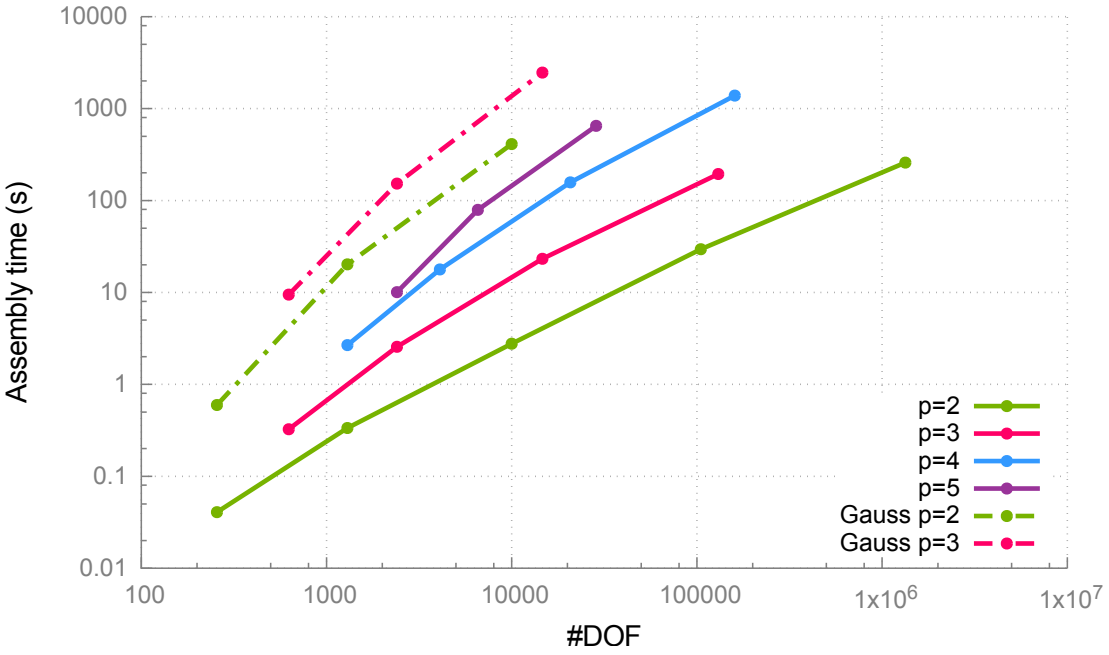


Figure 5.5.: Computation times for assembling the *global system matrix* on the four-dimensional space-time domain (with spatial part as in Fig. 5.4) for the space-time dependent diffusion coefficient (5.16).

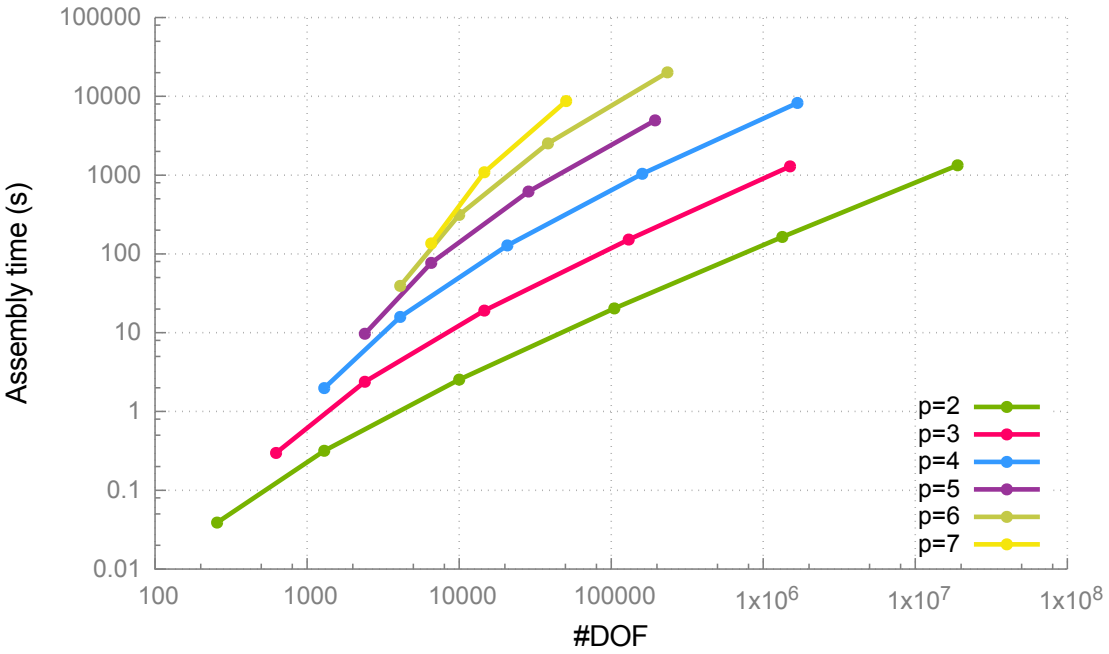


Figure 5.6.: Computation times for assembling the *Kronecker format* on the four-dimensional space-time domain (with spatial part as in Fig. 5.4) for the space-time dependent diffusion coefficient (5.16).

Part B.

Assembly on trimmed domains

6. Contribution and outline

In Part B of this thesis, we develop an efficient and accurate quadrature rule for trimmed domains.

The flexibility of NURBS surface representations in computer-aided design is greatly enhanced by the use of trimming. More precisely, only a part of the parameter domain is active, while a region defined by a set of trimming curves is discarded. For example, the trimming curves can be the result of a surface-surface intersection. Analogously, a trimmed volume is defined by a tensor-product NURBS patch and a set of trimming surfaces separating the active and inactive part of the volume.

Since IgA aims at a unification of the representations used in CAD and in numerical simulation, it is necessary treat trimmed patches directly. We remark that different CAD representations are possible, e.g. subdivision surface-based models or T-spline models, see [7, 10, 46].

Besides its the application in IgA, trimming also plays a role in the context of unfitted finite element methods [13], such as the finite cell method [75]. In this class of methods, the solution is approximated using a background finite element mesh that is intersected with the underlying geometry. Most of the results that will be presented in part of the thesis can be also applied directly to this class of methods.

There are multiple challenges that arise when using IgA on trimmed domains. One issue is the efficient coupling between adjacent trimmed patches. To this end, a finite cell method with weak coupling has been proposed in [77], a tearing and interconnecting approach was recently studied in [94] as well as Discontinuous Galerkin methods [38, 39].

Another issue is the numerical stability of the trimmed basis functions, since basis functions with a tiny support can appear around the trimmed boundary.

This instability can be addressed using modified bases such as extended B-Splines [42, 69, 67] or immersed B-Splines [78]. Other approaches are the addition of a stabilization term (ghost penalty) to the bilinear form [15] and the omission of basis functions with sufficiently small support [24].

Finally, the problem of applying numerical quadrature on a trimmed patch is a challenge in its own right. For IgA, efficient and accurate quadrature rules are needed to perform numerical integration on the trimmed elements of the discretization spaces. They are required in order to assemble the Galerkin matrices as well as the right-hand sides (load vectors) of the discrete systems.

Several approaches to numerical integration on trimmed domains have been considered. Most of this earlier work is limited to the two-dimensional case, see [68] for an overview. We discuss a number of related results.

A first idea is to place quadrature points on the full surface and set the weights of the points lying outside the trimmed domain to zero. However, this method has no guarantees and the integration error cannot be controlled reliably. In engineering practice local adaptive quadrature is used on top of it, that is, subdivision is performed around the trimmed region and quadrature nodes are placed in each sub-cell [27, 47, 86]. This approach can generate an extensive number of quadrature points, thus creating efficiency barriers.

Another approach to numerical quadrature on trimmed domains is to subdivide the trimmed elements into simpler objects and to find local approximations that can be dealt with by high-order quadrature rules. The accuracy of the overall procedure is then governed by the geometric approximation error. The principle of approximate reparameterization of trimmed domains is also known as *untrimming*.

In [51], the authors use base cases for two-dimensional trimmed elements and perform local untrimming, using the intersection points of the trimmed curve and the boundary; see also [87] for some applications of this machinery in optimization. In [81] a local untrimming on the element level is performed by a projection method, which can be interpolation or least-squares fitting.

When the trimming curves are complicated and have high degree, it is typical to compute a piecewise linear approximation of the boundary to simplify further processing [7, 11, 54, 76]. When it comes to numerical integration, the geometry approximation error accumulates in the final result, and deteriorates the overall

approximation order.

In the volumetric case, this approach has been described in [3], where the trimming surface is approximated by piece-wise polynomials, and in [43], where a marching cube-type method is used. Tetrahedral elements have also been used for the discretization of partial differential equations on trimmed volumes [92]. In [70], the trimmed patch is subdivided into a number of singular tensor-product B-Spline patches whose boundaries approximate the trimming surface. Clearly, the success of these approaches depends on the ability to compute surface-surface intersections, which is well-known to be a delicate problem [73]. Among other approaches, the use of interval arithmetics has been proposed to achieve robustness in degenerate or nearly-degenerate situations [40].

Several works on trimmed quadrature by local approximation of the geometry have been published in the context of unfitted finite element methods. In [59], the author uses a piecewise linear approximation of the boundary and applies a transformation of the finite element mesh in order to improve the approximation order. For the case of Cut Finite Element Methods (CutFEM, [25]), a method for numerical integration using boundary integrals and Taylor approximation in a Nitsche formulation was developed in [16]. For the finite cell method, the smart octree method for quadrature on volumetric domains has been introduced in [53]. It is based on an extended adaptive subdivision of the elements into octants. More precisely, the octant nodes are moved to the trimming surface, in order to increase the accuracy of the result.

Yet another interesting approach has been presented in [71], where the authors show how to construct a specific quadrature rule for each trimmed element with the help of non-linear optimization. As a prerequisite, the method evaluates the integrals of polynomials on a piece-wise linear approximation to the trimmed domain using a subdivision into convex subdomains, and this determines the overall accuracy. The numerical experiments demonstrate that the method is suitable for two-dimensional problems.

Finally, we note that several techniques exist that can help to avoid trimming. Besides using the result of untrimming also for the analysis, these include discontinuous Galerkin methods on overlapping patches [93] and Schwarz-type domain decomposition methods [12, 48].

6. Contribution and outline

In the present work we develop an *efficient and accurate* quadrature rule for the approximation of integrals over trimmed domains. We begin the presentation of our new method by considering the case of a trimmed two-dimensional domain. The trimming curve is given implicitly by a real-valued level function on the whole domain. This does not impose any restrictions, since any trimming curve can be converted into this format by employing implicitization techniques, which can be either exact or approximate ones, see, e.g., [20, 89]. An implicit representation of a trimming curve can also be the result of a Boolean operation on two surfaces. Our method is based on an error correction approach that is applied to local quadrature cells. In the case of IgA, these cells are the elements of a spline discretization. In a first step, we obtain an adaptive subdivision of the cell in such a way that each subcell falls in a predefined base case. We then extend the classical approach of linear approximation of the trimming curve by adding an error correction term based on a Taylor expansion of the blending between the linearized implicit trimming curve and the original one.

In terms of *accuracy*, adding the error correction term improves the local quadrature error in each cell by two orders of magnitude compared to the piecewise linear approximation of the trimming curve, thus providing an extra order of convergence globally. In particular, cubic order of convergence is achieved for the compound quadrature rule with a negligible additional computational cost. The *efficiency* of the method is achieved by the fact that it requires solely the evaluation of the trimming function at the vertices of the cells and the quadrature nodes, and refrains from any kind of point inversion or non-linear solving. Furthermore, our method is *easy to implement*, since the resulting nodes for the correction term are simply one-dimensional Gauss nodes and their corresponding weights are given by a direct computation. Moreover, we do not need to test for quadrature points outside the integration domain or treat them in a different way. Overall, it is straight-forward to incorporate the method into existing codes. Our treatment of two-dimensional domains has been published in [84].

Next, we generalize this idea to the case of trimmed volumetric domains. Again, we assume that the trimming surface is given implicitly as a level set of a function, for example as the result of a Boolean operation on two domains or the result of an implicitization method. The trivariate trimming function is approximated by a

linear function using constrained least squares optimization, thereby ensuring a consistent topology. The resulting quadratic order of convergence of the quadrature rule is then increased by adding a simple error correction term. Since this error correction term only consists of a bivariate integral, it preserves the overall computational complexity, making the method very efficient. Our method for numerical quadrature on trimmed three-dimensional domains has been published as a technical report [82].

In the numerical experiments, we apply our method to the problems of L^2-projection and isogeometric analysis on trimmed volumes. To address the problem of numerical stability, we implemented the extended B-Spline basis, which recovers stability by eliminating unstable functions and adding them to a number of stable ones with appropriate weights, thereby ensuring that the approximation power stays the same.

7. Preliminaries

In the remainder of Part B of this thesis, we present our method for numerical quadrature on trimmed two-dimensional and three-dimensional domains. The content of this chapter applies to both, the two-dimensional and the three-dimensional case. We begin the presentation in Section 7.1 by formally stating the considered problem of finding a quadrature rule for a trimmed quadrature cell. This local rule can be used for global quadrature in a compound quadrature rule, see Section 7.2. We then give an outline of the overall method in Section 7.3. In Section 7.4, we describe the algorithm that we use in order to arrive at a suitable subdivision of the integration domain for the compound quadrature rule.

7.1. Problem formulation

Throughout this part of the thesis, we consider integrals of functions on a trimmed axis-aligned box $B \subset [0,1]^d$, where d is either 2 or 3. We assume that the integrand is a smooth function

$$f : B \to \mathbb{R}$$

and that the trimming curve or surface is defined implicitly as the zero-level set of another smooth function

$$\tau : B \to \mathbb{R}.$$

This means that the integration region is defined as

$$B_\tau = \{\boldsymbol{x} \in B : \tau(\boldsymbol{x}) > 0\}.$$

We develop a method for the numerical evaluation of the integral

$$I_{B,\tau} f = \int_{B_\tau} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \tag{7.1}$$

More precisely, we approximate the exact integral using a weighted quadrature rule

$$Q_{B,\tau} f = \sum_i w_i f(\boldsymbol{x}_i) \approx I_{B,\tau} f \tag{7.2}$$

with quadrature nodes $\boldsymbol{x}_i$ and weights w_i.

Remark 8. *Usually, the trimming functions in CAD and not given implicity but by low degree parametric curves. It is then possible to convert these curves into implicit form by invoking suitable implicitization techniques, which can be either exact or approximate ones, see e.g. [20, 89].*

7.2. Compound quadrature rule

The problem (7.1) is motivated by its application in isogeometric analysis, where we need to compute integrals over the parametric domain $[0,1]^d$ in order to compute the system matrix for isogeometric discretizations of partial differential equations. For elliptic problems, we derived in Section 3.2 that the integrand then takes the form (3.10-3.12). Typically, the numerical quadrature for computing the integrals in the system matrix is performed element-wise, which means that the axis-aligned box is an element of the spline discretization and both, the f and τ are defined on the whole parametric domain $\hat{\Omega} = [0,1]^d$. This means that in the case of a trimmed domain we approximate the integral

$$\int_{\hat{\Omega}_\tau} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \sum_{B \in \mathcal{B}} \int_B f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \tag{7.3}$$

by applying the quadrature rule (7.2) on each element B of the spline discretization. Here,

$$\hat{\Omega}_\tau = \{\boldsymbol{x} \in \hat{\Omega} : \tau(\boldsymbol{x}) > 0\}$$

is the global trimmed parameter domain and $\mathcal{B}$ denotes the set of elements. On each element, the function $f : B \to \mathbb{R}$ consists of a product of polynomial segments of tensor-product B-splines and a smooth kernel that reflects the influence of the geometry mapping (i.e., the parameterization of the computational domain by a NURBS surface) and the coefficient functions of the PDE.

More generally, we can apply the method to approximate the integral of any (piece-wise) smooth function over the trimmed parametric domain $\hat{\Omega}_\tau$ by summing up the local results of (7.2) over the cells of a subdivision $\mathcal{B}$ of the parametric domain $\hat{\Omega} = [0, 1]^d$. The resulting quadrature rule

$$\overline{Q}_\tau f = \sum_{B \in \mathcal{B}} Q_{B,\tau} f \approx \int_{\hat{\Omega}_\tau} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \tag{7.4}$$

is called *compound quadrature rule* for the subdivision $\mathcal{B}$. We control the quadrature error with respect to a step-size h corresponding to the resolution of the subdivision.

7.3. Description of the method

Our approach to finding a local quadrature rule of the form (7.2) consists of three main steps:

1. We subdivide the box B into smaller boxes until the sign distribution of the trimming function τ at the boxes' vertices is an instance of a base cases. The number of possible base cases depends on the dimension, they will be presented in the next chapters of this thesis.

2. We approximate the trimming function τ by a linear polynomial σ and perform an approximate evaluation of the integral

   $$I_\sigma f = \int_{B_\sigma} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

   by Gauss quadrature.

3. We improve this initial approximation to $I_\tau f$ by adding an error correction term that is derived from the Taylor expansion of the linear interpolant

$I_{\sigma+u(\tau-\sigma)}f$ at $u = 0$. This results in a $d-1$-variate integral, which is again evaluated via Gauss quadrature.

Summing up, the quadrature rule Q_τ consists of two types of quadrature nodes and weights in each cell: the nodes and weights of the d-variate tensor-product Gauss quadrature performed in the second step and the nodes and weights of the $d-1$-variate Gauss quadrature needed to evaluate the error correction term in the third step.

7.4. Adaptive subdivision

In order to define a compound quadrature rule (7.4) based on the local quadrature rules, we need to find a subdivision $\mathcal{B}$ of the whole domain $\hat{\Omega}$ into boxes that are suitable for our method.

Our method is designed to be able to treat a number of base cases, which are characterized by the sign distribution of the trimming function on the vertices of B. The base cases for the quadrature rule on two-dimensional and three-dimensional domains are described in detail in Sections 8.1 and 9.1, respectively. Intuitively, the sign distribution captures how the trimming surface cuts the box B. In addition, we only treat boxes, whose maximum edge length does not exceed the user-defined constant h, in order to maintain the accuracy of the numerical integration.

The sign configurations that do not correspond to a base case are dealt with by further subdividing the box B, depending on the dimension into 2^d sub-boxes. The

Data: τ, $h > 0$
Let the initial cell be $B = [0,1]^d$.
Set $\mathcal{B} = \varnothing$.
while *there is an untreated cell* B*:* **do**
  if *Size*$(B) < h$ *and BaseCase*(τ, B) **then**
    Add B to $\mathcal{B}$.
  else
    $B_1, \ldots, B_{2^d} =$ *Subdivide*(B)
    Mark $B_1, \ldots, B_{2^d}$ as untreated.
  end
end

Algorithm 1: QuadratureCells

splitting is applied recursively until each generated box is an instance of one of the base cases. Similarly, we split the box B into 2^d sub-boxes if the maximum edge length exceeds h. The total number of generated boxes satisfies $N_h = O(\frac{1}{h^d})$ if the trimming curve or surface is regular within B.

Applying this procedure to the whole parametric domain $\hat{\Omega} = [0,1]^d$ leads to a subdivision $\mathcal{B}$ that consists of cells $B \in \mathcal{B}$ which all correspond to a base case of our method and whose size is smaller than the given step-size. We summarize this algorithm for finding a subdivision of $\hat{\Omega}$ that is suitable for the compound quadrature rule (7.4) based on our method in Algorithm 1.

8. Quadrature on trimmed two-dimensional domains

In this chapter we present our quadrature rule for trimmed two-dimensional domains. As outlined in Section 7.3, we first describe the base cases of our method in Section 8.1. Then, we find a first approximation of the integral's value by approximating the trimming function τ with a linear polynomial. In Section 8.3, we define the first order error correction term, which increases the approximation power. Finally, we analyze the convergence property of the resulting quadrature rule theoretically in Section 8.4 and conclude the chapter with an analysis of the computational complexity and the presentation of our numerical results.

8.1. Base cases

As we described in Section 7.4, our method is able to handle a number of base cases, which depend on the dimension. These base cases are defined by the sign distribution of the trimming function on the vertices of the axis-aligned box B, similar to the marching cubes algorithm [72]. We treat zero-values at the vertices as positive values. Therefore, for $d = 2$, there are $2^4 = 16$ possible configurations, which can be reduced to a set of six different ones by only defining them up to rotations.

The base cases are defined in such a way that they either correspond to a single component of the trimming curve inside B or to an untrimmed box. We arrive at five configurations that are depicted in Figure 8.1, we call those configurations the *base cases* of our method. Consequently, the method does not detect branches of the trimming curve that leave and re-enter the cell within the same edge. In order to illustrate this fact, Figure 8.2 shows two instances of each of the three

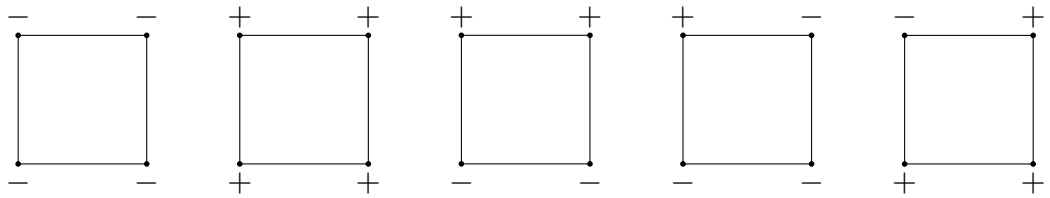


Figure 8.1.: Sign distributions of the trimming function τ for the five base cases (up to rotation).

trimmed base cases. Our method also results in the correct approximation order if this situation occurs.

In the two-dimensional case, there is only sign distributions (up to rotation) that does not represent a base case, see Figure 8.3. As described in Section 7.4, these situations are dealt with by uniformly subdividing the corresponding cell. This process is guaranteed to terminate if no singularities of the trimming curve are present (which is always the case in practice).

Figure 8.4 shows an instance of quadrature cells for the compound rule (7.4) generated by Algorithm 1. The zero set of the trimming function consists of two parallel lines, which are parallel to one of the square's diagonals. The parameter h was chosen as $\frac{1}{2}$. For this specific instance of τ, the algorithm adds up to two additional subdivision steps at the northwest and southeast corners of the domain.

8.2. Linearized trimmed quadrature

We consider the quadrature on a two-dimensional axis-aligned quadrature cell $B \subset \mathbb{R}^2$. Thus, we need to approximate the integral

$$\int_{B_\tau} f(x, y) \, \mathrm{d}y \, \mathrm{d}x, \tag{8.1}$$

where

$$B_\tau = \{(x, y) \in B : \tau(x, y) > 0\}.$$

This approximation is trivial for the first two base cases in Figure 8.1: The value of the integral equals zero in the first case, and it is simply approximated by a

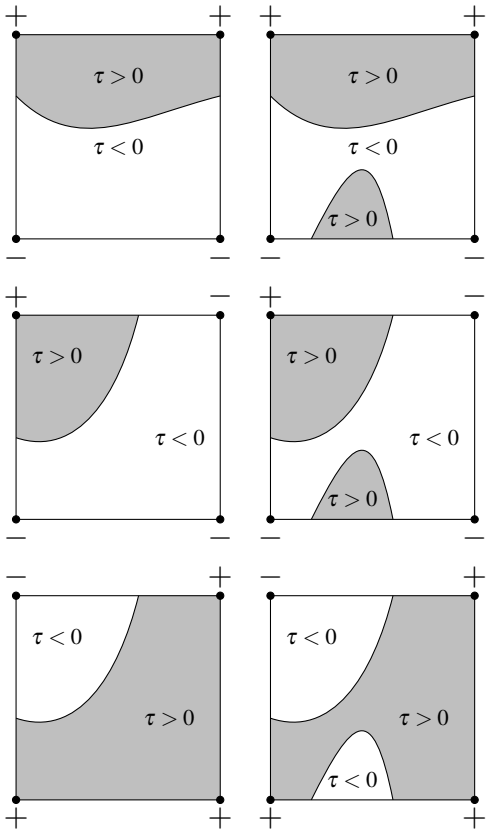


Figure 8.2.: Two instances of each trimmed base case (curved quadrilateral, triangle and pentagon).

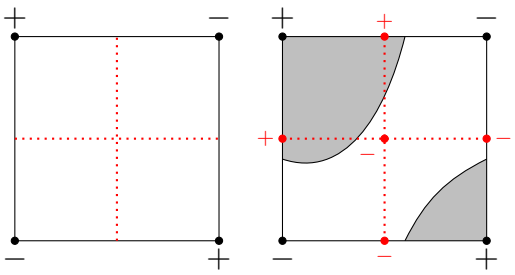


Figure 8.3.: Left: Sign distribution that does not represent a base case. Right: Uniform subdivision (quadsection) of this cell.

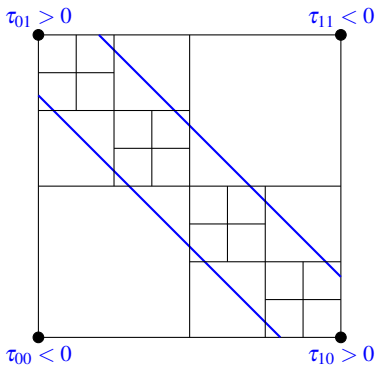


Figure 8.4.: Quadrature cells for $h = \frac{1}{2}$. The trimming curve is shown in blue.

tensor-product Gauss rule in the second one.

In order to perform this approximation in the remaining three base cases, we replace τ by another function σ. This function is chosen such that the integral

$$\int_{B_\sigma} f(x, y)\, \mathrm{d}y\, \mathrm{d}x \tag{8.2}$$

over the region

$$B_\sigma = \{(x, y) \in B : \sigma(x, y) > 0\}$$

enclosed by the zero-level set of σ admits a simple evaluation. We achieve this by using a suitable linear approximation of τ, that is, the level set $\sigma(x, y) = 0$ is simply a straight line segment.

The evaluation of (8.2) using numerical quadrature is based on the intersections of $\sigma(x, y) = 0$ with the boundary of the cell. We determine these intersections by linear interpolation of the function values of τ at the vertices of the cell. Figure 8.5 shows examples for this approximation. For this choice of σ, the linearized integral (8.2) belongs to the same base case as the original integral (8.1).

For the quadrilateral case we proceed as follows: First, we construct a bilinear parameterization of B_σ. Second, we transform the integral to the associated parameter domain and evaluate its value using Gauss quadrature with n evaluations per parametric direction, where n is chosen by the user. The triangular case is dealt with analogously, by using a parameterization with a singularity at the involved

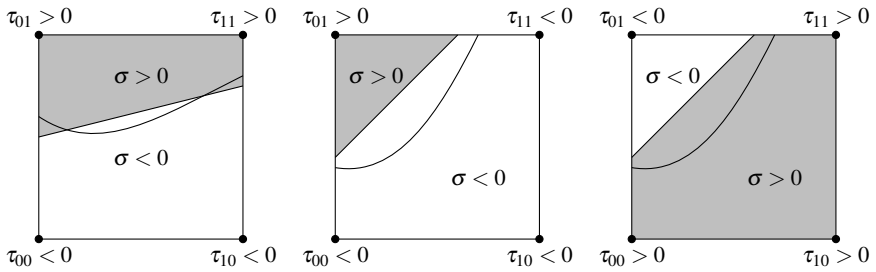


Figure 8.5.: A simple approximation of the three trimmed base cases.

cell vertex. In the pentagonal case, the identity

$$B_\sigma = B \setminus B_{-\sigma}$$

allows to evaluate (8.2) by combining results for the untrimmed and the triangular case.

The resulting linearized trimmed quadrature rule will be referred to as

$$\mathrm{LT}[B, \tau](n),$$

where n denotes the number of Gauss nodes. For the compound quadrature rule based on LT and the subdivision with maximum cell size h created by Algorithm 1 we use

$$\overline{\mathrm{LT}}[\tau](h, n)f = \sum_{B \in \mathcal{B}} \mathrm{LT}[B, \tau](n)f.$$

Clearly, the values generated by the LT rule converge to the true integral as h is decreased. As we shall see later, however, the approximation of the quadrature domain by a polygon limits the order of convergence.

8.3. Corrected linearized trimmed quadrature

We improve the order of convergence of the LT rule by adding an error correction term if B is in one of the three trimmed base cases. This term is found by performing a Taylor expansion.

Throughout this section, we assume that the box B is given as

$$B = [x_0, x_0 + h] \times [y_0, y_0 + h] \in \mathcal{B}$$

and that it is either a quadrangular or triangular base case, see Figure 8.5. Recall that the pentagonal case is solved by considering the complementary triangular domain.

Linear blending of the trimming function τ and its linear approximation σ leads to the subsets

$$B_{\sigma+u(\tau-\sigma)} = \{(x, y) \in B : \sigma(x, y) + u(\tau(x, y) - \sigma(x, y)) > 0\}, \quad u \in [0, 1].$$

We define the function

$$F(u) = \int_{B_{\sigma+u(\tau-\sigma)}} f(x, y)\,\mathrm{d}y\,\mathrm{d}x, \tag{8.3}$$

it attains the exact value of (7.3) for $u = 1$, while the LT rule is based on the approximate evaluation of $F(0)$. We improve the accuracy by adding a correction term that is based on the first two terms of the Taylor series

$$F(1) = F(0) + F'(0) + R_1(1) \tag{8.4}$$

of F around $u = 0$, where R_1 denotes the remainder.

In order to compute $F'(0)$, we observe that the level set of the function obtained by linear blending defines a function $y = c_u(x)$ or $x = c_u(y)$ for sufficiently small values of u, where the projection of the trimming curve onto the x or y axis specifies the domain

$$[a_u, b_u],$$

respectively. If both choices are possible, we choose the one with the larger domain for $u = 0$.

Without loss of generality, we consider the first case

$$[a_u, b_u] \subset [x_0, x_0 + h]$$

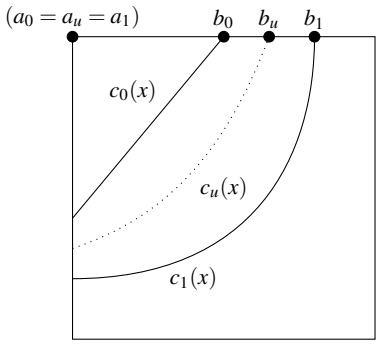


Figure 8.6.: Linear blending of τ and σ in the triangular base case.

where the function satisfies

$$\sigma(x, c_u(x)) + u(\tau(x, c_u(x)) - \sigma(x, c_u(x))) = 0, \tag{8.5}$$

see Figure 8.6.

Lemma 9. *The first order error correction term is given by*

$$F'(0) = \frac{1}{\frac{\partial}{\partial y}\sigma} \int_{a_0}^{b_0} f(x, c_0(x))\tau(x, c_0(x))\,\mathrm{d}x \tag{8.6}$$

Proof. By differentiating (8.5) we observe that

$$\frac{\partial}{\partial u} c_u(x) = \frac{-\tau(x, c_u(x)) + \sigma(x, c_u(x))}{\frac{\partial \sigma}{\partial y}(x, c_u(x)) + u(\frac{\partial \tau}{\partial y}(x, c_u(x)) - \frac{\partial \sigma}{\partial y}(x, c_u(x)))}. \tag{8.7}$$

The function F in (8.3) can be rewritten as

$$F(u) = \int_{a_u}^{b_u} \int_{c_u(x)}^{y_0+h} f(x, y)\,\mathrm{d}y\,\mathrm{d}x.$$

Its first derivative thus evaluates to

$$\begin{aligned} F'(u) \quad = \quad & -\int_{a_u}^{b_u} f(x, c_u(x)) \frac{\partial}{\partial u} c_u(x)\,\mathrm{d}x \\ & -\frac{\mathrm{d}}{\mathrm{d}u} a_u \int_{c_u(a_u)}^{y_0+h} f(a_u, y)\,\mathrm{d}y + \frac{\mathrm{d}}{\mathrm{d}u} b_u \int_{c_u(b_u)}^{y_0+h} f(b_u, y)\,\mathrm{d}y \end{aligned} \tag{8.8}$$

The integration limit satisfies $b_u = x_0 + h$ or

$$c_u(b(u)) = y_0 + h.$$

Consequently, the third term in (8.8) vanishes since either the integral or the factor in front of it take value zero. Similarly, the second term vanishes as well.

Finally we use (8.7), (8.8) and the fact that σ vanishes on the graph of c_0,

$$\sigma(x, c_0(x)) = 0,$$

to rewrite the first order correction term as a univariate integral over the linearized trimming curve, thus arriving at (8.6). □

An approximate value of (8.6) is computed using a Gauss rule.

Remark 10. *In Lemma 17 we will present a more general form of the correction term for the three-dimensional case, based on Reynold's transport theorem* (9.9)*. The representation* (8.6) *follows from using a specific choice of parameterization of the linearized trimming curve.*

Note that in Section 8.5 we only used the zero level set of σ. The value of the correction term depends on the gradient of the linearized trimming function, which we did not discuss so far. In fact, we would like to choose $\nabla\sigma$, and more specifically $\frac{\partial\sigma}{\partial y}$, so that it approximates sufficiently well $\nabla\tau$. A good choice is to set $\frac{\partial\sigma}{\partial y}$ using finite differences over the values of τ at the cell's vertices. In the quadrilateral case we use the average

$$\frac{\partial\sigma}{\partial y} = \frac{1}{2}\left(\frac{\tau_{01} - \tau_{00}}{h} + \frac{\tau_{11} - \tau_{10}}{h}\right) = \frac{\tau_{01} - \tau_{00} + \tau_{11} - \tau_{10}}{2h}$$

of the finite differences over the two edges that intersect the trimming curve. In the triangular case we use the finite difference

$$\frac{\partial\sigma}{\partial y} = \frac{\tau_{01} - \tau_{00}}{h}$$

over the edge which intersects the trimming curve.

The resulting corrected linearized trimmed quadrature rule (with first order correction term) will be referred to as

$$\mathrm{CLT}[B,\tau](n,k),$$

where n resp. k are the numbers of bivariate resp. univariate quadrature nodes.

In addition, we use

$$\overline{\mathrm{CLT}}[\tau](h,n,k) = \sum_{B\in\mathcal{B}} \mathrm{CLT}[B,\tau](n,k)$$

to denote the compound quadrature rule based on CLT and the subdivision $\mathcal{B}$ defined by Algorithm 1 for the maximum step size h.

If the trimming function τ is linear and thus $\sigma = \tau$, then

$$\tau(x, c_0(x)) = 0.$$

Consequently, the correction term (8.6) vanishes. In this case, CLT and LT give equivalent results.

8.4. Convergence result

In this section we will show that the first order error correction in the CLT rule improves the convergence by one order with respect to the non-corrected LT rule. More precisely, we will prove this result for a slightly modified version of $\overline{\mathrm{CLT}}$, obtained by adapting the quadrature cells, which we denote as $\overline{\mathrm{CLT}}^{\star}$. Intuitively, we construct the modified version of $\overline{\mathrm{CLT}}$ as follows: We look at all cells where the gradient of the trimming function is "almost" horizontal or vertical, for some point. In this case we fuse two adjacent cells in such a way that the trimmed cell becomes of the quadrilateral base case which is easier to treat theoretically than the triangular case.

We analyze the convergence of the compound rule (7.4) based on LT and CLT with respect to h-refinement of the underlying modified subdivision created by Algorithm 1.

First we prove two technical results about the local errors in the trimmed cells of the quadrilateral base case (Lemma 11) and of the triangular base case (Lemma 12). Second we combine them with the known approximation properties of the employed Gauss rules to estimate the global quadrature error in Theorem 13.

Both lemmas consider a rectangular cell (not necessarily a square) B of size h and a trimming function τ defined on it. We derive error bounds that applies to all cells of these base cases.

We say that the cell satisfies the assumptions (about the base cases) *in the strong sense* if the trimming curve crosses the boundary in exactly two points.

Lemma 11. *Assume that a rectangular cell B fulfills the assumptions of the quadrilateral base case in the strong sense, and the trimming function τ and its linear approximation σ satisfy the inequalities*

$$\left|\frac{\partial \tau}{\partial y}(x,y)\right| \geq C_1 \quad , \quad \forall (x,y) \in B \tag{8.9}$$

and

$$\|\sigma - \tau\|_{L^\infty(B)} \leq C_2 h^2, \tag{8.10}$$

$$\|\nabla\sigma - \nabla\tau\|_{L^\infty(B)} \leq C_3 h \tag{8.11}$$

for certain positive constants C_1, C_2, C_3. Then there exists a constant

$$C_{\text{quad}}(C_1, C_2, C_3, f)$$

which depends solely on these three constants and f, such that the corrected trimmed quadrature on this cell fulfills for $n = k = 2$

$$|I_{B,\tau} f - \text{CLT}[B,\tau](2,2) f| \leq C_{\text{quad}} h^4,$$

where h is the size of B.

Proof. We denote the rectangular cell by $B = [x_0, x_0 + \alpha h] \times [y_0, y_0 + \beta h]$ where (x_0, y_0) is the lower left vertex and $\alpha, \beta > 0$. Since by the monotonicity assumption (8.9) the trimming curve $\tau(x,y) = 0$ can be written as a graph, the same is

true for all intermediate curves if h is sufficiently small. By differentiating (8.8) and using that both $a_u = x_0$ and $b_u = x_0 + \alpha h$ are constant, we obtain, for all $u \in [0,1]$,

$$F''(u) = -\int_{a_u}^{b_u} \frac{\partial}{\partial y} f(x, c_u(x)) \left(\frac{\partial}{\partial u} c_u(x)\right)^2 + f(x, c_u(x)) \frac{\partial^2}{\partial u^2} c_u(x)\, \mathrm{d}x \tag{8.12}$$

We observe that under the assumptions (8.9) - (8.11)

$$\left|\frac{\partial}{\partial u} c_u(x)\right| = \frac{|\tau(x, c_u(x)) - \sigma(x, c_u(x))|}{|(1-u)\frac{\partial}{\partial y}\sigma + u\frac{\partial}{\partial y}\tau(x, c_u(x))|} \leq C' h^2. \tag{8.13}$$

where C' depends on C_1, C_2, C_3. By differentiating (8.5) twice we obtain

$$\frac{\partial^2}{\partial u^2} c_u(x) = \frac{-2\frac{\partial}{\partial u} c_u \left(\frac{\partial}{\partial y}\tau - \frac{\partial}{\partial y}\sigma\right) - u\left(\frac{\partial}{\partial u} c_u\right)^2 \frac{\partial^2}{\partial y^2}\tau}{(1-u)\frac{\partial}{\partial y}\sigma + u\frac{\partial}{\partial y}\tau(x, c_u(x))}$$

and thus using again the assumptions on τ and σ we get

$$\left|\frac{\partial^2}{\partial u^2} c_u(x)\right| \leq C'' h^3 \tag{8.14}$$

where C'' depends again on C_1, C_2, C_3. By estimating the integral by the supremum we conclude that for all $u \in [0,1]$

$$F''(u) \leq C''' h^4. \tag{8.15}$$

The result is obtained by combining Taylor's theorem with the approximation properties of the employed Gauss rules for the bi- and univariate quadrature. □

Lemma 12. *Assume that a rectangular cell B satisfies the assumptions of the triangular base case (in the strong sense) and that in addition to the assumptions (8.9)-(8.11) in Lemma 11 there is a constant C_4 independent of h, such that*

$$\left|\frac{\partial \tau}{\partial x}(x, y)\right| \geq C_4 > 0 \quad , \quad \forall (x, y) \in B \tag{8.16}$$

Then, there exists a constant $C_{\text{triangle}}(C_1, C_2, C_3, C_4, f)$, such that the corrected

trimmed quadrature on this cell fulfills

$$|I_{B,\tau} f - \mathrm{CLT}[B,\tau](2,2) f| \leq C_{\mathrm{triangle}} h^4,$$

where h is the size of B.

Proof. Again, we write $B = [x_0, x_0 + \alpha h] \times [y_0, y_0 + \beta h]$. In the triangular case, b_u in (8.8) is not constant but defined implicitly by

$$c_u(b_u) = y_0 + \beta h.$$

For the second derivative of F this means that an additional term appears which depends on the derivative of b_u:

$$\begin{aligned} F''(u) &= -\int_{a_u}^{b_u} \frac{\partial}{\partial y} f(x, c_u(x)) \left(\frac{\partial}{\partial u} c_u(x) \right)^2 + f(x, c_u(x)) \frac{\partial^2}{\partial u^2} c_u(x) \, \mathrm{d}x \\ &\quad - f(b_u, c_u(b_u)) \frac{\partial}{\partial u} c_u(b_u) \frac{\mathrm{d}}{\mathrm{d}u} b_u. \end{aligned} \tag{8.17}$$

In order to estimate the last term in (8.17), we compute

$$\frac{\mathrm{d}}{\mathrm{d}u} b_u = -\frac{\frac{\partial}{\partial u} c_u(b_u)}{\frac{\partial}{\partial x} c_u(b_u)}.$$

Differentiating (8.5) with respect to x leads to

$$\frac{\partial}{\partial x} c_u(x) = -\frac{\frac{\partial}{\partial x}\sigma + u\left(\frac{\partial}{\partial x}\tau(x, c_u(x)) - \frac{\partial}{\partial x}\sigma\right)}{\frac{\partial}{\partial y}\sigma + u\left(\frac{\partial}{\partial y}\tau(x, c_u(x)) - \frac{\partial}{\partial y}\sigma\right)}.$$

Using assumption (8.16) we conclude

$$\left| \frac{\partial}{\partial x} c_u(x) \right| \geq C'''' > 0$$

and thus

$$\left| \frac{\mathrm{d}}{\mathrm{d}u} b_u \right| \leq C'''' h^2.$$

Therefore, in view of (8.13) and (8.14) the last term in (8.12) satisfies

$$f(b_u, c_u(b_u))\frac{\partial}{\partial u}c_u(b_u)\frac{\mathrm{d}}{\mathrm{d}u}b_u \leq C''''''h^4$$

and the result follows. □

Unfortunately, even with these two results at hand, we cannot analyze quadrature rule $\overline{\mathrm{CLT}}$ directly. This is due to two reasons: First, we cannot guarantee that all the triangular cells in the subdivision $\mathcal{B}$ satisfy the assumption of Lemma 12. Second, there may be trimmed cells which are not base cases in the strong sense. Both problems are resolved by suitably modifying the quadrature rule.

More precisely, we construct a modified quadrature rule $\overline{\mathrm{CLT}}^\star$ by replacing the subdivision $\mathcal{B}$ of Ω with a new subdivision $\mathcal{B}^\star$. We begin by using creating a uniform grid of cells of size $\frac{h}{2}$. We assume that h is small enough, such that all cells belong to one of the base cases. The subdivision $\mathcal{B}^0$ consists of all cells that possess a non-empty intersection with the integration domain. Note that this also includes cells where the trimming curve crosses the same edge twice.

We will obtain $\mathcal{B}^\star$ by merging some of the trimmed cells in $\mathcal{B}^0$.

First, we define the constants C_1 and C_4 that are to be used in Lemmas 11 and 12 as

$$C_1 = C_4 = \frac{1}{4}\min_{\tau(x,y)=0}\|\nabla\tau(x,y)\|_2. \tag{8.18}$$

If h is sufficiently small this means that each *trimmed* cell $B \in \mathcal{B}^0$ belongs to one of three classes:

1. Class H (horizontal gradient): For all $(x, y) \in B$ we have

$$\left|\frac{\partial}{\partial x}\tau(x,y)\right| \geq C_4, \text{ and } \left|\frac{\partial}{\partial y}\tau(x_0,y_0)\right| < C_1 \text{ for at least one } (x_0,y_0) \in B. \tag{8.19}$$

2. Class V (vertical gradient): For all $(x, y) \in B$ we have

$$\left|\frac{\partial}{\partial y}\tau(x,y)\right| \geq C_1, \text{ and } \left|\frac{\partial}{\partial x}\tau(x_0,y_0)\right| < C_4 \text{ for at least one } (x_0,y_0) \in B. \tag{8.20}$$

3. Class D (diagonal gradient): For all $(x, y) \in B$ we have

$$\left|\frac{\partial}{\partial x}\tau(x,y)\right| \geq C_4 \text{ and } \left|\frac{\partial}{\partial y}\tau(x,y)\right| \geq C_1. \tag{8.21}$$

This is illustrated in Figure 8.7. If a cell B belongs to class H (resp. class V), then

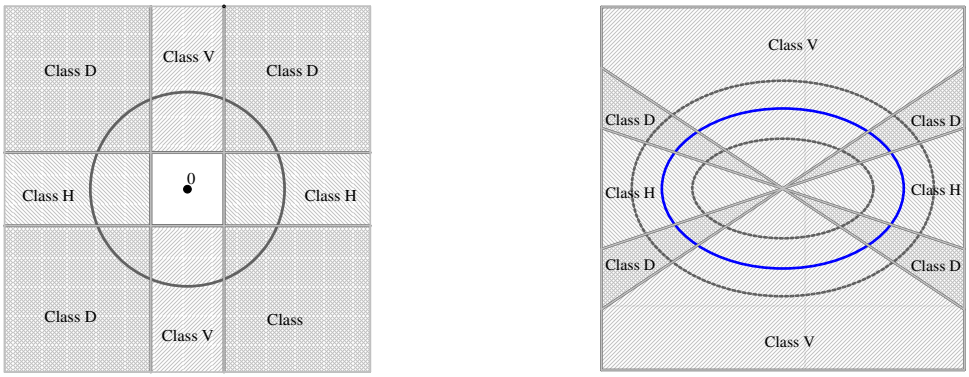


Figure 8.7.: Illustration of the regions of cell classes V, H and D. Left: The regions defined by $\nabla\tau(x, y)$ for (x, y) over the union of the trimmed cells. The black circle has radius $4C_1 = \min_{\tau(x,y)=0} \|\nabla\tau(x, y)\|_2$. It is visible that $\nabla\tau(x, y)$ lies outside the square in the middle. Right: An example trimming curve (an ellipse, shown in blue). The dotted offsets enclose the region that contains trimmed cells.

also all of its neighbors are either in class H (resp. V) or in class D. To obtain the modified subdivision $B^\star$ we merge all pairs of vertically adjacent cells where one of them is in class H. Similarly, we merge all pairs of horizontally adjacent cells where one of them is in class V. The remaining cells are kept. Note that this results in rectangular cells of maximum size h, since at most two cells will be merged, due to the restricted range of the gradients. The modified rule $\overline{\text{CLT}}^\star$ is obtained by applying CLT to the modified subdivision $\mathcal{B}^\star$.

Next, we prove the convergence result for the modified rule $\overline{\text{CLT}}^\star$. The modified quadrature cells in $\overline{\text{CLT}}^\star$ ensure that all triangular cells belong to class D, where we have a bound on both partial derivatives, cf. (8.21). Moreover, all cells satisfy the assumptions about the base cases in the strong sense.

Theorem 13. *Assume $\tau \in C^2([0, 1]^2)$ such that the constant $C_1 = C_4$ as defined in (8.18) is positive, and $f \in C^4([0, 1]^2)$. Then, there exists a constant $C_{\tau,f}$, such*

that the $\overline{\mathrm{CLT}}^\star$ *rule with* $n = 2$ *and* $k = 2$ *satisfies*

$$|I_{[0,1]^2,\tau} f - \overline{\mathrm{CLT}}^\star[\tau](h, 2, 2)f| \leq C_{\tau,f} h^3. \tag{8.22}$$

Proof. By the construction of $\overline{\mathrm{CLT}}^\star$, the subdivision $\mathcal{B}^\star$ consists of untrimmed quadrature cells, trimmed quadrilateral cells of classes H, V and D, and trimmed triangular and pentagonal cells of class D. In the trimmed quadrilateral cells we can always apply Lemma 11, while in the trimmed triangular cells of class D we can apply Lemma 12. Moreover, we can treat the trimmed pentagonal cells of class D by applying Lemma 12 to the complement of the quadrature domain. In both cases the constants C_2 and C_3 are obtained by linear approximation of τ. They depend on the second derivative of τ whose norm is bounded.

The number of trimmed cells does not exceed $C_5 \frac{1}{h}$ for some constant C_5. Moreover, we can use the same constants C_{quad} and C_{triangle} for all trimmed cells. Indeed, these constants depend on the values $C_1, \ldots, C_4$, which are determined by derivatives of the trimming function. Consequently, a global upper bound for these constants exists and depends solely on the trimming function τ. Moreover, we may use an upper bound on the derivatives of f. We conclude

$$\sum_{B \in \mathcal{B}^\star_{\mathrm{trimmed}}} |I_{B,\tau} f - \mathrm{CLT}[B, \tau](2, 2)f| \leq C_5 \max\{C_{\mathrm{quad}}, C_{\mathrm{triangle}}\} h^3.$$

Since we use $n = 2$ Gauss nodes in each direction for the untrimmed cells, the local error is bounded by $C_{\mathrm{Gauss}} h^5$ in each of these cells for some constant C_{Gauss}. Since there are at most $\frac{1}{h^2}$ untrimmed cells, we have

$$\sum_{B \in \mathcal{B}^\star_{\mathrm{untrimmed}}} |I_{B,\tau} f - \mathrm{CLT}[B, \tau](2, 2)f| \leq C_{\mathrm{Gauss}} h^3.$$

The result (8.22) is implied by these two inequalities, since the various constants depend on τ and f only.

□

We conjecture that in the triangular base case the influence of the last term in (8.17) is canceled by the corresponding term in the adjacent cell. Consequently, in

practice it suffices to use $\overline{\mathrm{CLT}}$ instead of $\overline{\mathrm{CLT}}^\star$. This is supported by our numerical experiments.

8.5. Computational complexity

The error correction term in CLT only consists of a univariate integral. As we will show in the complexity analysis in this section, this leads to a very efficient method. More precisely, CLT has the same asymptotic complexity as LT which is moreover the same complexity as the one of a simple Gauss quadrature.

In the analysis, we assume that each evaluation of f and τ takes constant time. We also assume that we use the same number q of Gauss nodes in each direction both for the bivariate quadrature in LT and for the univariate quadrature in the correction term for CLT.

Theorem 14. *LT requires $O(q^2)$ function evaluations.*

Proof. The determination of the intersection points of $\sigma = 0$ with the boundary by interpolation for LT is performed in constant time. Thus, the complexity for finding the approximations σ is $O(1)$.

Therefore, the complexity is dominated by the bivariate quadrature which is performed up to two times per cell. Since we use a q^2-point Gauss rule, the overall complexity is $O(q^2)$. □

Theorem 15. *The computational complexity of CLT is the same as the computational complexity of LT.*

Proof. The first order correction term (9.8) consists of a univariate integral which is approximated using the same amount of Gauss nodes per direction as for the bivariate integral. Since the number of function evaluations in the computation of the weights (9.10) is constant in h, the overall complexity is governed by the bivariate integral in LT and the complexities are asymptotically the same. □

The correspondent computational complexity for the compound quadrature rules follows immediately:

Theorem 16. *The overall complexity of the compound quadrature rules* $\overline{\mathrm{LT}}$ *and* $\overline{\mathrm{CLT}}$ *is* $O(N_h q^2)$*, where* N_h *is the number of cells in the subdivision* $\mathcal{B}$ *with maximum cell size* h*.*

8.6. Numerical experiments

The method was implemented in C++ using the G+Smo library [65]. In this section, we show experiments for the linearized trimmed quadrature rule CLT as well as the linearized trimmed quadrature rule LT on a number of trimmed geometries, both two-dimensional and three-dimensional.

We analyze several properties, such as the approximation power and the complexity, of CLT and compare it to LT.

Implementing the modified rule $\overline{\mathrm{CLT}}^\star$ that was constructed in Section 8.4 for the theoretical analysis of the compound two-dimensional quadrature rule can be computationally inefficient in practice since it is necessary to estimate the gradient of τ in all cells for deciding which cells are to be joined. Additionally, one needs to find an upper bound of the gradient's norm along the trimming curve. However, we also implemented $\overline{\mathrm{CLT}}^\star$ in a particular case in order to make a direct comparison with CLT. In the numerical experiments we observe that the theoretical error estimate for $\overline{\mathrm{CLT}}^\star$ (Theorem 13) still holds for the original CLT quadrature rule.

Ellipse As a first example, we use our method to compute the volume of an ellipse implicitly defined by

$$\tau(x,y) = -\frac{(x-0.5)^2}{a^2} - \frac{(y-0.5)^2}{b^2} + 1 > 0, \tag{8.23}$$

where we set $a = 0.45$ and $b = 0.2$ in our experiment. Figure 8.8 shows the result of the subdivision of this ellipse after some steps of refinement. In each cell, the trimming function was approximated by a linear function as described in Section 8.2. Note that in the pentagonal case we integrate over the remaining triangle and subtract from the full integral. In the left plot in Figure 8.9 we show the quadrature error for different values of h when computing the area enclosed by the ellipse with the simple trimmed quadrature $LT(h,1)$ described in Section 8.2 and with the

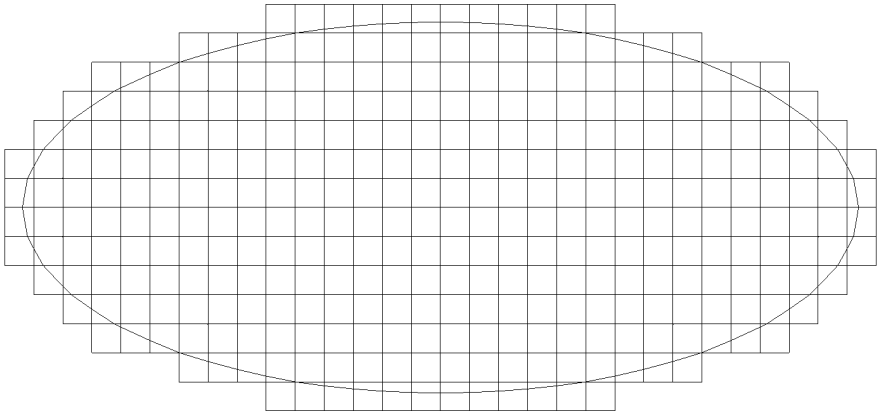


Figure 8.8.: Subdivision of the ellipse after some steps of refinement with approximate linear trimming curve in each cell.

trimmed quadrature by first order correction $CLT(h, 1, 2)$ described in Section 8.3. We observe that the first order error correction results in an additional order of convergence with respect to h, confirming the theoretical result from Theorem 13.

Next, we show the computation times for both quadrature rules in Figure 8.10. We observe that applying the error correction does not result in a significant increase in complexity compared to the linearized trimmed quadrature. The complexity for both LT and CLT increases linearly with the number of cells.

Finally, we compare the CLT quadrature rule with the modified quadrature rule $\overline{\text{CLT}}^\star$ which was defined in Section 8.4 for the theoretical analysis of the convergence. For the ellipse given by the trimming function (8.23) the exact value of the constants $C_1 = C_4$ in the conditions (8.19), (8.20) and (8.21) is

$$C_1 = \frac{1}{4} \min_{\tau(x,y)=0} \|\nabla \tau(x, y)\|_2 = \frac{10}{9}.$$

We sample the gradient of τ on a fine grid in each cell in order to check these conditions. In Figure 8.11 we show the modified subdivision of the ellipse after some steps of refinement. Eight cells were joined in this case, they are highlighted in the picture. Figure 8.12 shows the error of both CLT and $\overline{\text{CLT}}^\star$ when computing the area of the ellipse. We observe that the quadrature errors for both methods

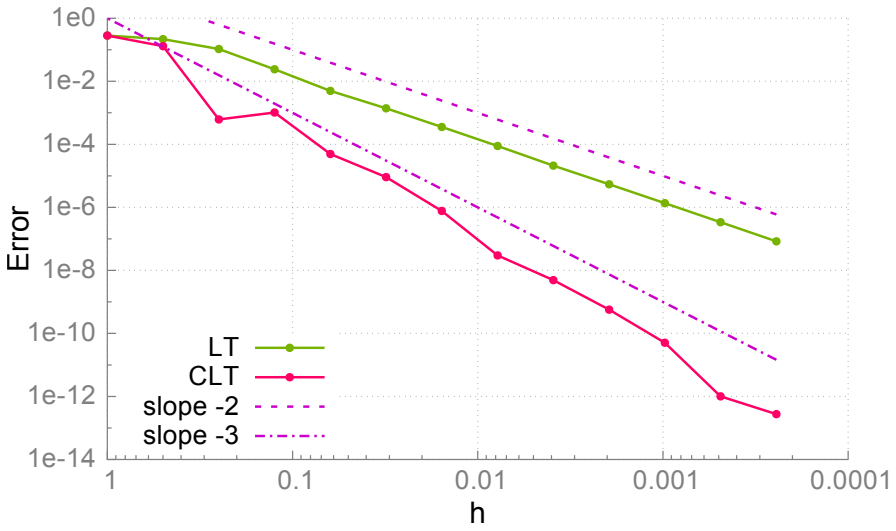


Figure 8.9.: Absolute error in LT and CLT for the area enclosed by the ellipse.

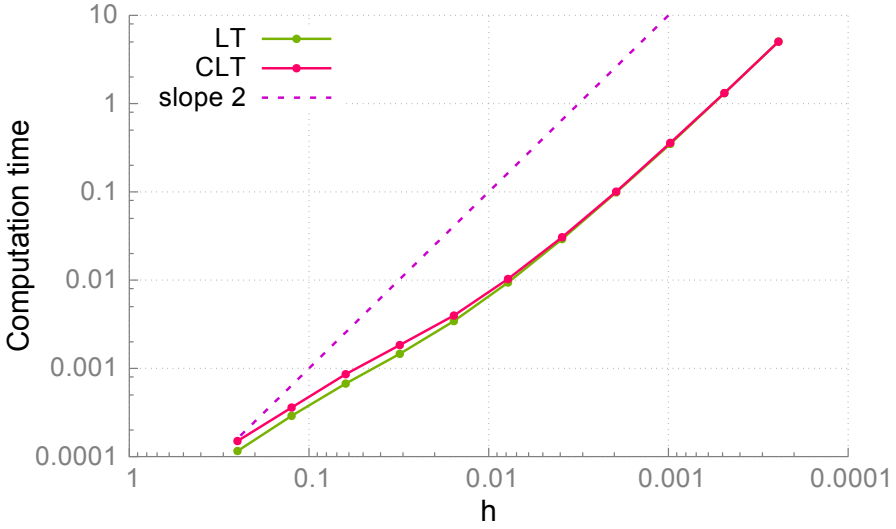


Figure 8.10.: Computation times for the approximation of the area enclosed by the ellipse

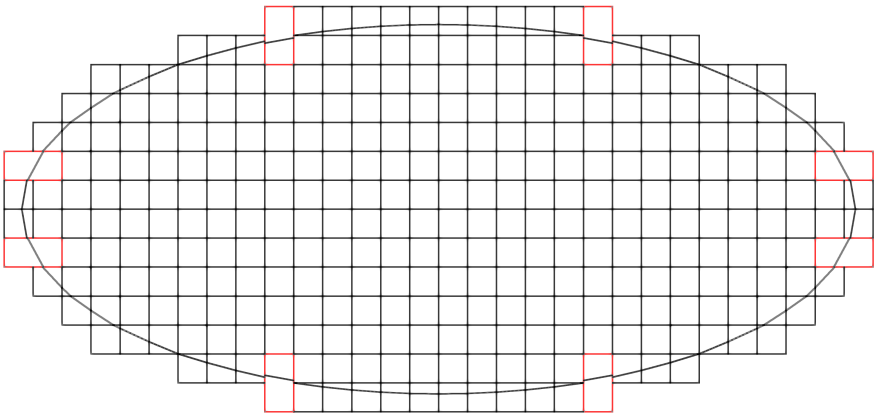


Figure 8.11.: Modified subdivision of the ellipse for $\overline{\text{CLT}}^{\star}$ after five steps of refinement. Joined cells are highlighted.

have the same asymptotic behavior.

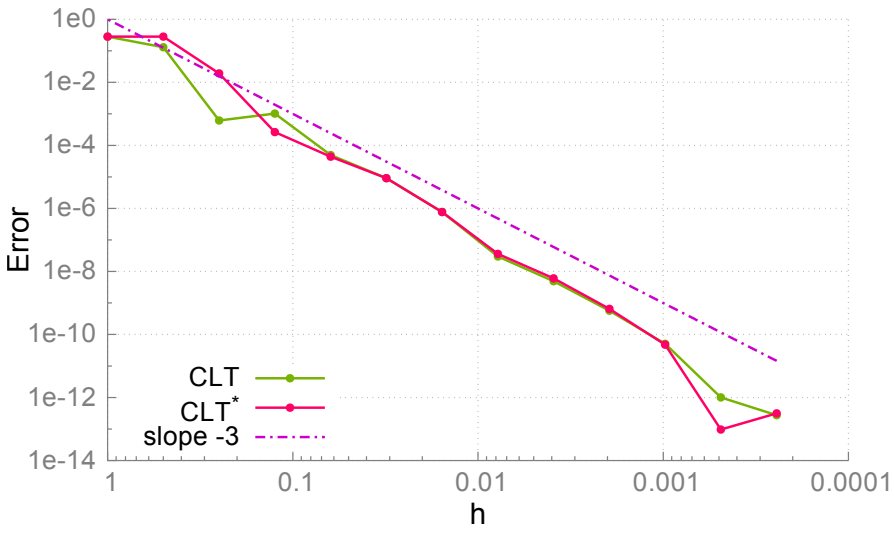


Figure 8.12.: Absolute error in CLT and $\overline{\text{CLT}}^{\star}$ for the area enclosed by the ellipse.

Perforated quarter annulus In our next example, we will approximate the area of a quarter annulus which is trimmed with three circles. The domain is represented by Non-Uniform Rational B-Splines (NURBS), which is a powerful technique for the representation of complex geometric shapes, see [74] for more details. The

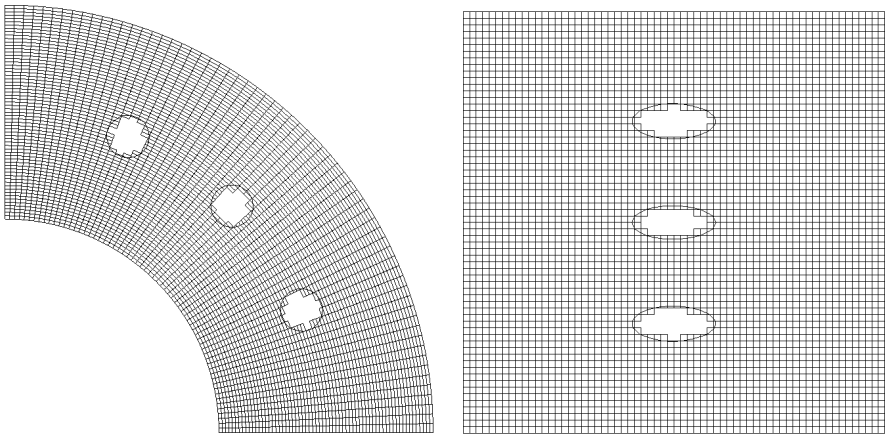


Figure 8.13.: Subdivision of the perforated quarter annulus and its corresponding parametric domain after some steps of refinement with approximate linear trimming curve in each cell.

quarter annulus is represented exactly as a NURBS domain of polynomial degrees $(p_1, p_2) = (1, 2)$ with no interior knots [1]. Figure 8.13 shows the piecewise linear approximation of the trimming curve in the computational and in the parametric domain after some steps of refinement.

Since we perform the computation on the parametric domain, we approximate the integral

$$\int_{\Omega_{\gamma \circ G}} |\det J_G(x, y)| \, \mathrm{d}y \, \mathrm{d}x,$$

where $G : [0.1]^2 \to \mathbb{R}^2$ is the NURBS parameterization of the quarter annulus and $J_G(x, y)$ is the Jacobian matrix of G at the point (x, y). The trimming function $\gamma : \mathbb{R}^2 \to \mathbb{R}$ on the physical domain is defined as the product of the implicit representations of the three circles.

In the first plot in Figure 8.14 we show the convergence rate of the quadrature rules with and without error correction. We chose $n = 2$ Gauss nodes for the linearized quadrature and $k = 2$ Gauss nodes for the correction term. As in the case of the ellipse, we observe that the first order error correction term in CLT

[1] The weights were chosen as $(\omega_{0,0}, \omega_{1,0}, \omega_{0,1}, \omega_{1,1}, \omega_{0,2}, \omega_{1,2}) = \left(1, 1, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 1, 1\right)$ and the corresponding control points as $(1, 0), (2, 0), (1, 1), (2, 2), (0, 1), (0, 2)$.

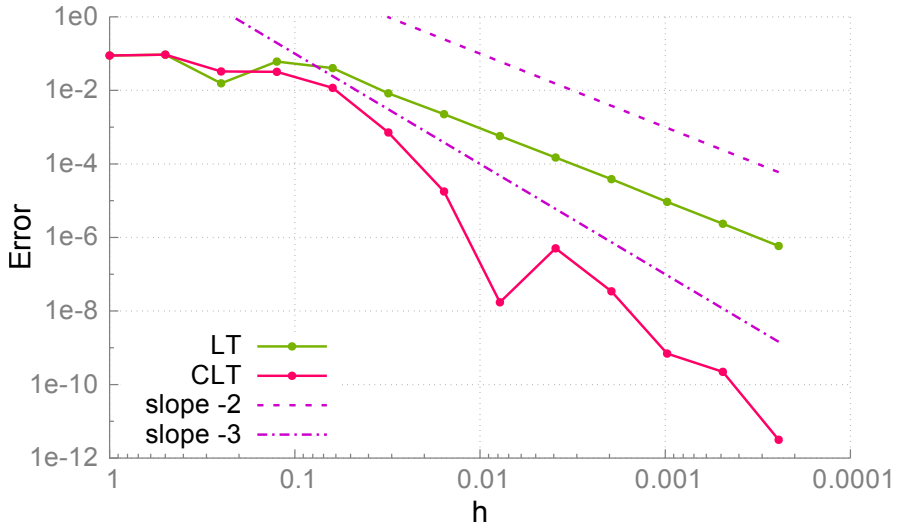


Figure 8.14.: Absolute error in LT and CLT for the area of the perforated quarter annulus for $n = 2, k = 2$.

results in an additional order of convergence compared to the linearized quadrature in LT.

Since we only use one error correction term, the convergence error cannot be improved by additional Gauss nodes in the bivariate and univariate quadrature. This is confirmed by the second plot in Figure 8.15 which shows the same experiment as in the first plot but for $n = 3$ and $k = 3$.

Singular case: Bicuspid curve Figure 8.16 shows a linear approximation of the bicuspid curve which is an algebraic curve given by

$$(x^2 - a^2)(x - a)^2 + (y^2 - a^2)^2 = 0$$

for some $a > 0$. It has two cusps that hinder the improvement of the approximation order by the error correction term. In Figure 8.17 we show the error convergence of the approximation of the area enclosed by the bicuspid, where the reference value was computed with a lower value of h. We observe that the error correction in CLT does not improve the convergence rate in this case, however, the absolute error is lower.

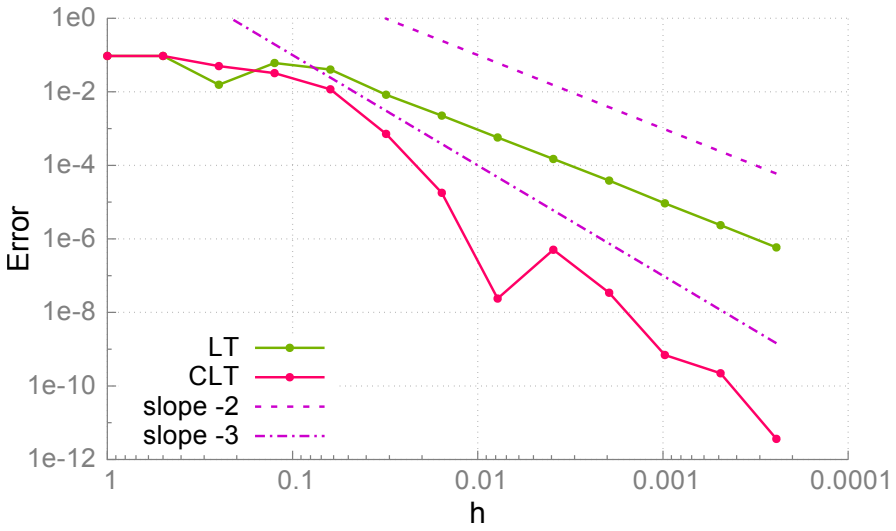


Figure 8.15.: Absolute error in LT and CLT for the area of the perforated quarter annulus for $n = 3, k = 3$.

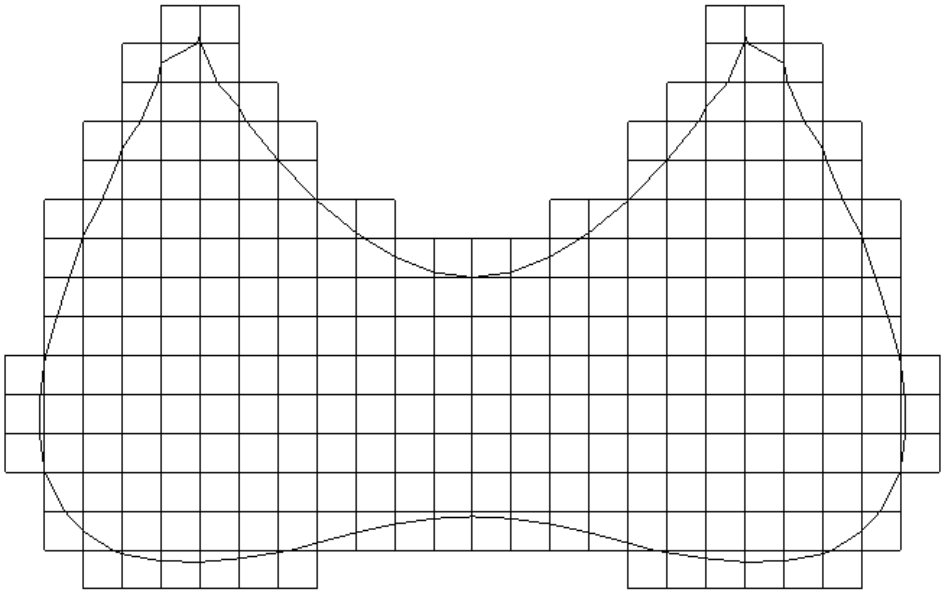


Figure 8.16.: Subdivision of the bicuspid with approximate linear trimming curve in each cell.

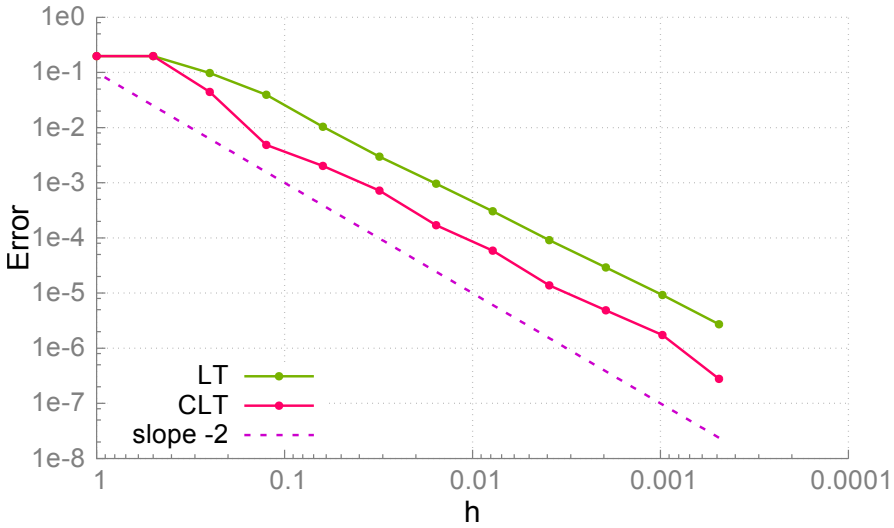


Figure 8.17.: Absolute error for the computation of the area enclosed by the bicuspid curve using LT and CLT.

9. Quadrature on trimmed three-dimensional domains

We generalize the method for quadrature on trimmed surfaces to quadrature on trimmed volumes. The main idea of the algorithm is the same as in the two-dimensional case and this chapter follows the outline of Chapter 8. However, the treatment of trimmed volumes is much more complicated because of the much higher number of possibilities how a trimming surface might intersect any given box. This calls for a more thorough reduction of the number of base cases and more involved techniques have to be applied in order to find a linear approximation of the trimming surface that locally preserves the topology.

9.1. Base cases

For $d = 2$, there were only 16 different sign distributions, if we distinguish between positive and non-positive signs of τ at the vertices of B. For $d = 3$ this number is much higher, there are $2^8 = 256$ possible cases. However, this number can be reduced significantly by considering the 48 isometries of the cube (i.e. the rotations and reflections of the cell) and possibly inverting the signs of τ at the vertices. This results in the 15 cube configurations of the marching cube algorithm [72].

Five of these configurations, which are shown in Figure 9.1, correspond to a single component of the trimming surface within B. More precisely, if the trimming surface has a single component within B, then the sign distribution belongs to one of these configurations. These are the base cases of our method for the three-dimensional case. They are handled directly. In addition we also handle the case of only positive signs, which corresponds to an untrimmed box $B = B_\tau$. As in the

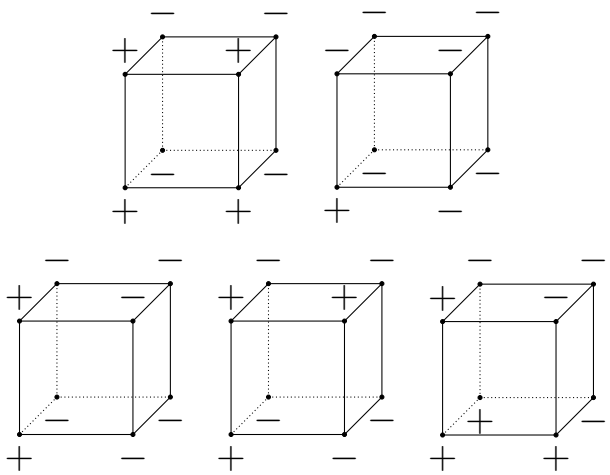


Figure 9.1.: The base cases defined up to rotations, reflections and sign inversion.

two-dimensional case, this case does not require any particular treatment, simple Gauss quadrature is sufficient.

The remaining nine configurations are dealt with by splitting the box B into eight sub-boxes as described in Section 7.4.

Again, it should be noted that the trimming surface may possess several components within B even if the sign distribution belongs to one of the base cases. As we shall see later, our technique gives correct results also in these situations.

9.2. Linearized trimmed quadrature

Consider a quadrature cell

$$B = [x_0, x_1] \times [y_0, y_1] \times [z_0, z_1].$$

We approximate the trimming function τ by a linear function

$$\sigma(x, y, z) = \sigma_1 x + \sigma_2 y + \sigma_3 z + \sigma_4 \tag{9.1}$$

and apply Gauss quadrature to the integral of f over

$$B_\sigma = \{(x, y, z) \in \Omega : \sigma(x, y, z) > 0\}.$$

The approximation ensures that the signs of σ on the vertices of B agree with the signs of τ. While in the two-dimensional case such an approximation can be obtained quite easily, we need to solve a constrained optimization problem in the three-dimensional case. This is described in the next section.

Since the zero-level set of the linear function σ is a plane, its intersection with B is a polygon. Hence, B_σ is a polyhedron. In order to get quadrature points and weights for B_σ we need to find a multi-patch parameterization over the reference domain $[0,1]^3$. This is described in section 9.2.

Constrained least squares approximation

The approximation of the trimming function τ by a linear function σ of the form (9.1) with the additional condition that the plane $\sigma(x,y,z)=0$ intersects B in the same edges as τ leads to a constrained least squares problem:

$$\begin{aligned} &\min_{\sigma\in\mathbb{R}^4} \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1}(\sigma_1 x_i + \sigma_2 y_j + \sigma_3 z_k + \sigma_4 - \tau_{ijk})^2 \\ &\text{subject to } S\sigma \geq 0. \end{aligned} \tag{9.2}$$

Here, τ_{ijk} is the value of τ on the vertex (x_i, y_j, z_k) of B.

The matrix S represents the conditions on the intersections of the plane with the edges of B. Since we allow for the zero-level set of σ to go through the vertices, the sign of $\sigma(x,y,z)$ on each vertex of B needs to be either the same as the sign of τ_{ijk} or zero. This gives the eight inequalities

$$\begin{cases} \sigma_1 x_i + \sigma_2 y_j + \sigma_3 z_j + \sigma_4 \geq 0 & \text{for } \tau_{ijk} \geq 0 \\ -\sigma_1 x_i - \sigma_2 y_j - \sigma_3 z_j - \sigma_4 \geq 0 & \text{for } \tau_{ijk} \leq 0. \end{cases}$$

Depending on the sign distribution of τ on B this number can be further reduced by omitting redundant conditions. For example, the matrix S for the second base

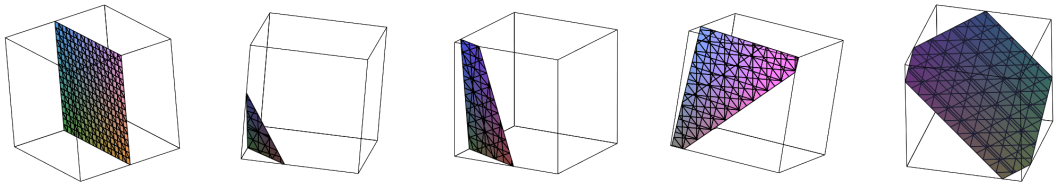


Figure 9.2.: Example level sets for the five base cases.

case in Figure 9.1 (the tetrahedral case) equals

$$S_{\text{tet}} = \begin{pmatrix} x_0 & y_0 & z_0 & 1 \\ -x_0 & -y_0 & -z_1 & -1 \\ -x_0 & -y_1 & -z_0 & -1 \\ -x_1 & -y_0 & -z_0 & -1 \end{pmatrix}.$$

The hexagonal base case (the fifth in Figure 9.1), has the maximum number of 8 constraints.

In order to solve this optimization problem computationally, we formulate it as a quadratic programming problem

$$\begin{aligned} &\min_{\sigma} \frac{1}{2}\sigma^{\intercal} A\sigma - b^{\intercal}\sigma \\ &\text{subject to } S\sigma \geq 0, \end{aligned} \tag{9.3}$$

where A and b are defined by the objective function (9.2). Several algorithms exist for solving quadratic programming problems. In our implementation we use the one presented in [30]. Since the size of the optimization problem is bounded, the time needed to compute the solution is bounded by a constant, independent of the specific configuration.

Figure 9.2 shows examples for the intersections of B with the zero-level sets of linear functions obtained by solving problem (9.3) for all five base cases, Figure 9.3 shows the approximation of a ball (which is defined by a trimming function) obtained by considering uniformly sized boxes with decreasing diameter.

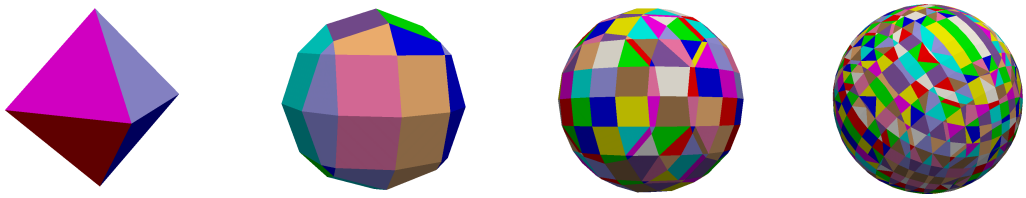


Figure 9.3.: Several steps of refinement of the linear approximation to the ball.

Parameterization of the approximate domain

In order to find the quadrature nodes, we need to define a parameterization of the polyhedral domain $B_\sigma = \{(x, y, z) \in B : \sigma(x, y, z) \geq 0\}$. In the first, second and third base case in Figure 9.1, the resulting polyhedron is a cuboid, a tetrahedron or a prism[1], respectively. A parameterization can be obtained by a single, trilinear parameterization, which is degenerate for the tetrahedron and the prism. For the fourth case, we create a multi-patch parameterization that consists of two degenerate trilinear patches representing prisms. In the fifth base case, we obtain three trilinear patches (two prisms and a four-sided pyramid). Instances of base cases 2-5 and the splitting into trilinear patches for cases 3 and 4 are shown in Figure 9.4.

If B is an instance of a rotation or a reflection of one of the base cases, the parameterization is simply composed with the respective transformation matrix. Whenever the box is in an inverted base case, i.e. one of the base cases with all signs flipped, we first compute the integral over the entire box B and then subtract the integral defined by the base case.

Numerical integration

We use a $q \times q \times q$-point Gauss rule for the approximation of the integral

$$I_\sigma f = \int_{B_\sigma} f(x, y, z) \, \mathrm{d}z \, \mathrm{d}y \, \mathrm{d}x. \tag{9.4}$$

This will be denoted as *Linearized Trimmed (LT)* quadrature rule.

[1] More precisely, it is a pentahedron, which is topologically equivalent to a triangular prism.

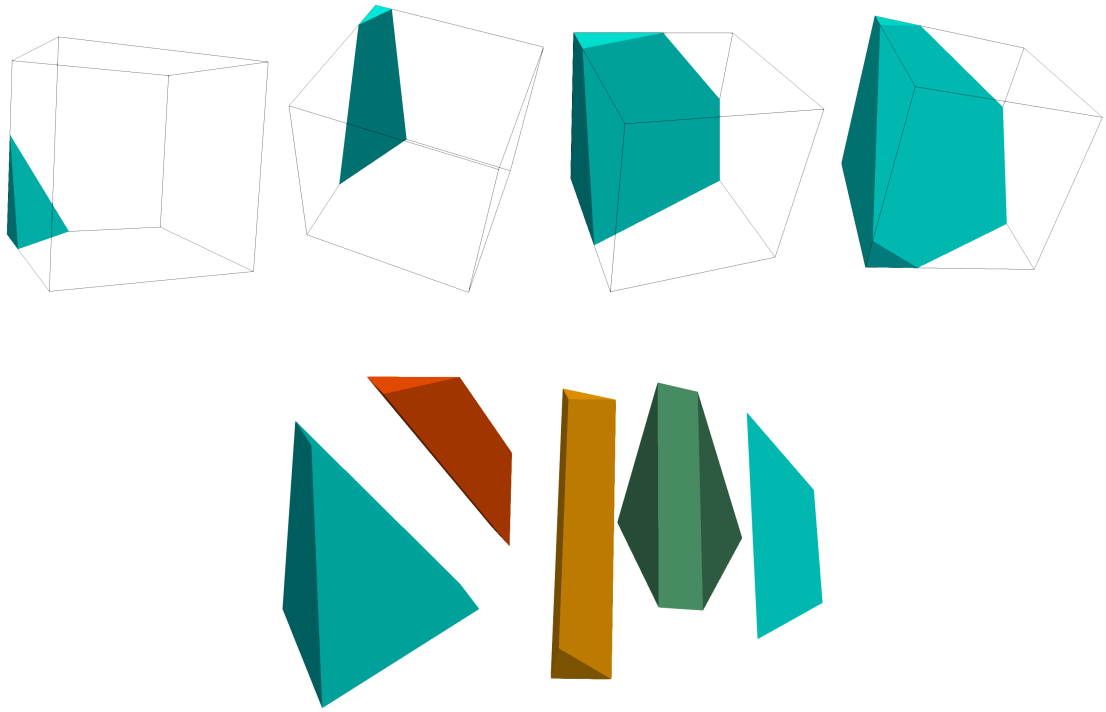


Figure 9.4.: Parameterization of the linearized domain: Base cases 2-5 (top row) and the subdivision into trilinear patches for cases 4 and 5 (bottom row).

If B is an instance of a base case without sign inversion, we obtain

$$\mathrm{LT}[B,\tau]f = \sum_{i=1}^{\#\text{patches}} \underbrace{\sum_{j=1}^{q^3} w_j^i f(x_j^i, y_j^i, z_j^i)}_{(\star)}, \tag{9.5}$$

where the nodes (x_j^i, y_j^i, z_j^i) are the images of the tensor-product Gauss nodes (ξ_j, η_j, ζ_j) for the corresponding patch G^i of the parameterization of B_σ and

$$w_j^i = w_j^{\text{Gauss}} |\det \nabla G^i(\xi_j, \eta_j, \zeta_j)|$$

with the tensor-product Gauss weights w_j^{Gauss}. Otherwise, if B is an inversion of an instance of a base case (i.e. the signs of τ at the cell vertices are flipped), then the signs of the weights w_j^i are flipped and the sum $(\star)$ that corresponds to the trilinear parameterization of the entire box is added to (9.5).

9.3. Corrected linearized trimmed quadrature

In the final step of our method, we add the error correction term to the cell-wise LT quadrature rule in order to alleviate the error caused by the linear approximation of the integration domain.

The first order correction term

First, we define the linear interpolant between σ and τ for $u \in [0, 1]$:

$$\eta_u(x, y, z) = \sigma(x, y, z) + u(\sigma(x, y, z) - \tau(x, y, z)). \tag{9.6}$$

We then consider the integral over the intermediate integration domains

$$B_{\eta_u} = \{(x, y, z) \in B : \eta_u(x, y, z) > 0\}$$

as a function of a parameter $u \in [0, 1]$, i.e.

$$F(u) = \int_{B_{\eta_u}} f(x, y, z) \, \mathrm{d}z \, \mathrm{d}y \, \mathrm{d}x.$$

Since σ and τ share the sign distribution at the box' vertices, this function is expected to be C^2-smooth for $u \in [0, 1]$. More precisely, it is C^2-smooth if the level sets of η_u intersect the boundary of the box transversally and do not cross any vertex or edge as u varies from 0 to 1. A two-dimensional illustration of this is shown in Figure 9.5.

By construction, $F(0) = I_\sigma f$ and $F(1) = I_\tau f$. In order to approximate $F(1)$, we perform a Taylor expansion of the function F around $u = 0$ and evaluate it at $u = 1$:

$$F(1) \approx F(0) + F'(0). \tag{9.7}$$

We call $F'(0)$ the *first order error correction term*. The right-hand side, where we use Gaussian quadrature to evaluate both contributions, defines the *Corrected Linearized Trimmed* (CLT) quadrature rule.

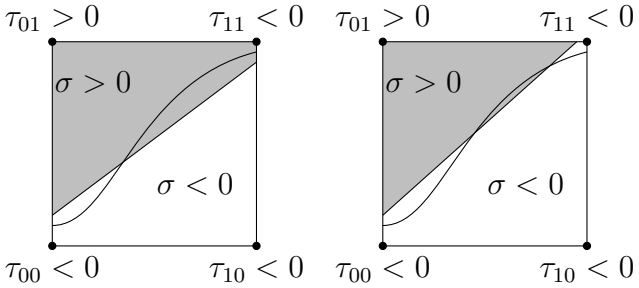


Figure 9.5.: Two-dimensional visualization of the continuity of F. For the approximation on the left-hand-side, the function $F(u)$ is smooth. For the approximation on the right-hand-side, it is only C^0-smooth, because the boundary of the integration domain B_{η_u} crosses the north-east corner.

Evaluation of the correction term

The first order error correction term admits a simple analytic representation.

Lemma 17. *The first order error correction term is equal to*

$$F'(0) = -\int_{\{\sigma=0\}\cap B} f \frac{\tau}{\|\nabla\sigma\|} \,\mathrm{d}s. \tag{9.8}$$

Proof. We evaluate the derivative of the integral $F(u)$ with respect to the parameter u with the help of Reynold's transport theorem and obtain

$$F'(u) = \int_{\{\eta_u=0\}\cap B} f v_u \,\mathrm{d}s, \tag{9.9}$$

where the normal velocity v_u is given by

$$v_u = -\frac{\frac{\mathrm{d}}{\mathrm{d}u}\eta_u}{\|\nabla\eta_u\|}.$$

From the definition (9.6) of η_u we infer

$$\frac{\partial\eta_u}{\partial u} = \tau - \sigma$$

and

$$\nabla \eta_u = \nabla \sigma + u(\nabla \tau - \nabla \sigma).$$

Setting $u = 0$ and using that σ vanishes on the zero-level we arrive at the representation (9.8). □

For the numerical evaluation of the bivariate integral (9.8), we consider the subdivision of the integration domain $\{\sigma = 0\} \cap B$, which is defined by the patches covering the polyhedral domain B_σ, see Section 9.2. This subdivision consists of at most two triangular and quadrilateral facets, which admit bilinear parameterizations. We use a $q \times q$ point Gauss rule to evaluate the contributions of the facets to the overall integral. Consequently, the CLT quadrature rule takes the form

$$\mathrm{CLT}[B,\tau]f = \mathrm{LT}[B,\tau]f + \sum_{i=1}^{\#\text{surface patches}} \sum_{j=1}^{q^2} v_j^i f(x_j^i, y_j^i, z_j^i),$$

where (x_j^i, y_j^i, z_j^i) are the Gauss nodes mapped to the i-th planar patch using the parameterization H^i and

$$v_j^i = \frac{v_j^{\mathrm{Gauss}} \tau(x_j^i, y_j^i, z_j^i) \sqrt{\det \nabla H^{i^T}(\xi_j, \eta_j) \nabla H^i(\xi_j, \eta_j)}}{\|\nabla \sigma(x_j^i, y_j^i, z_j^i)\|_2} \tag{9.10}$$

with the (2D) tensor-product Gauss weights v_j^{Gauss}.

In the experiments we will observe that the optimal number of Gauss nodes in each direction when using the first order error correction term is $q = 2$. This is expected, since otherwise the overall approximation order is dominated by the linear approximation and the correction term.

Higher order correction terms, which are beyond the scope of this thesis, involve edge integrals and point evaluations. Clearly, higher order Gauss rules are needed to achieve higher overall accuracy.

9.4. Computational complexity

The computational complexity in the three-dimensional case can be analyzed in the same way as we did in Section 9.4 for the two-dimensional case.

Theorem 18. *Let q be the number of Gauss nodes per direction for the trivariate quadrature in LT and the bivariate quadrature in the correction term for CLT.*

1. *Both LT and CLT require*

$$O(q^3)$$

   functions evaluations.

2. *The complexity of the compound rule is*

$$O(N_h q^3),$$

   where N_h is the number of cells in the subdivision $\mathcal{B}_h$.

Proof. The constrained least squares approximation for the three-dimensional LT rule is performed in constant time. Thus, the result follow using the same arguments as for the analogous results in Theorems 14-16 for the two-dimensional case. □

Remark 19. *The total number of generated boxes satisfies $N_h = O(\frac{1}{h^3})$ if the trimming surface is regular within $\hat{\Omega}$.*

9.5. Numerical experiments

Next, we present the numerical experiment for the quadrature rule on trimmed volumes. For solving the quadratic programming problem (9.3) in the three-dimensional CLT quadrature rule, we use QuadProg++ [21]. As we did in Section 8.6 for the case two-dimension domains, we analyze the approximation power and the complexity of CLT and compare it to LT and a simple Gauss quadrature over the untrimmed elements. Additionally, we apply our method for the three-dimensional trimmed quadrature to the problems of L^2-projection into a spline space and approximating the Poisson equation using an isogeometric discretization.

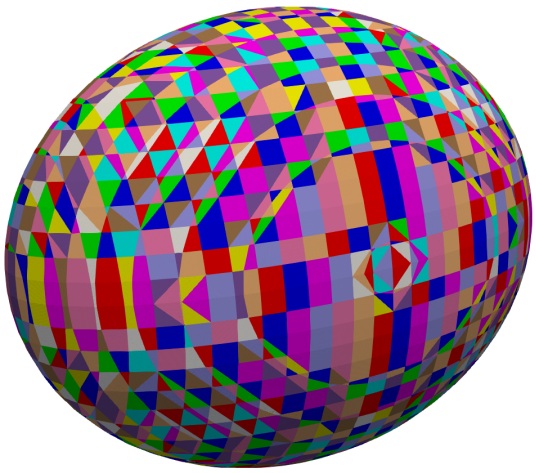


Figure 9.6.: Piece-wise linear approximation of the ellipsoid after five steps of refinement

9.5.1. Convergence of the quadrature

Like in the last Section, we analyze the behavior of our method with respect to h-refinement. This means that we first subdivide a given geometry into boxes of size h and then apply different quadrature rules to them.

Ellipsoid example In our first numerical example we use our method to compute the volume of the region enclosed by an ellipsoid given implicitly by

$$\tau(x, y, z) = -\frac{(x-0.5)^2}{a^2} - \frac{(y-0.5)^2}{b^2} - \frac{(z-0.5)^2}{c^2} + 1.$$

Its piece-wise linear approximation after five steps of refinement is shown in Figure 9.6. We compare the convergence of LT and CLT when using two Gauss nodes per direction in both the trivariate integral in LT and the bivariate integral in the first order error correction term for CLT. Additionally, we compare the two methods with simply using Gauss quadrature for all cells that lie completely inside the integration domain. This method will be denoted as the inner cell (IC) method.

In the plot shown in Figure 9.7 we observe that the use of error correction increases the order of convergence by one, resulting in cubic convergence. On the other hand, simply omitting all trimmed cells gives only linear convergence.

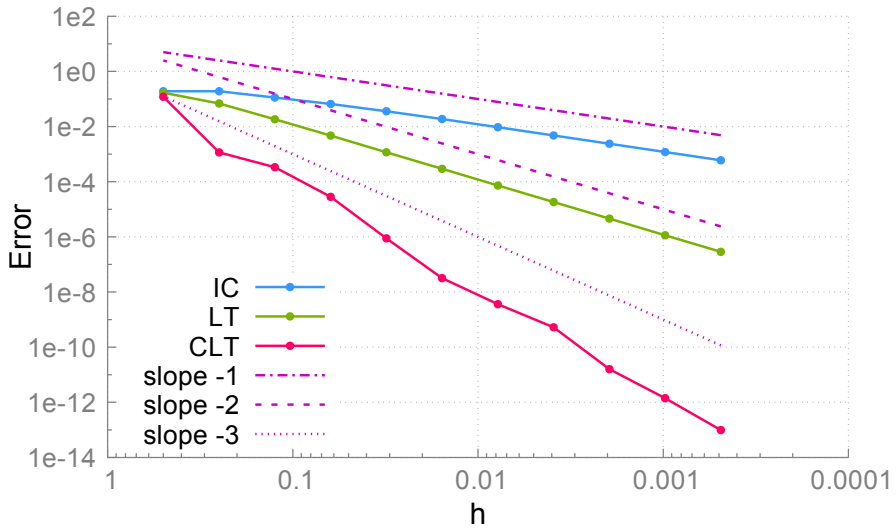


Figure 9.7.: Approximation of the volume enclosed by an ellipsoid using the LT and CLT.

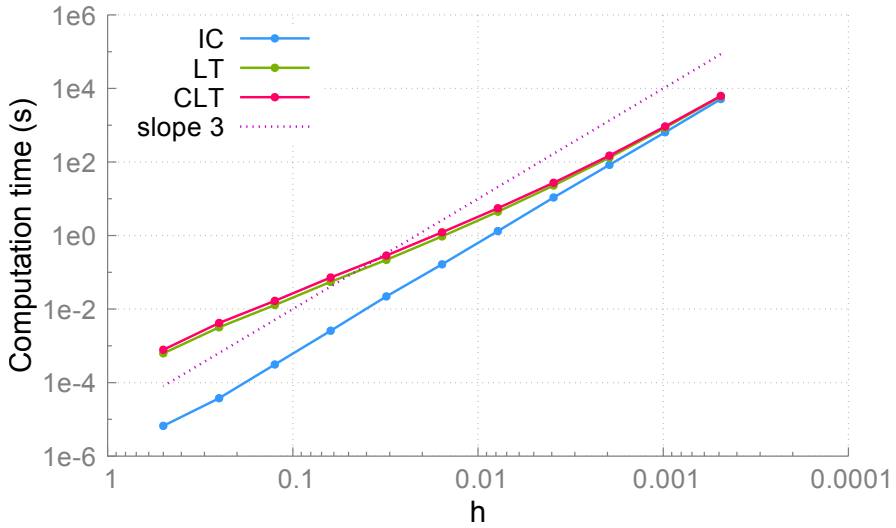


Figure 9.8.: Computation times for computing the volume enclosed by the ellipsoid

Section 9.4 showed that the computational complexity of both LT and CLT is linear with respect to the number of quadrature cells. This is the same complexity as for the usual Gauss quadrature over untrimmed cells. Figure 9.8 shows the computation times needed by our implementation of LT, CLT and IC for various values of the mesh size h. We observe asymptotically equivalent computation times of all three methods. The additional computational effort for CLT, which is required for the advanced treatment of the trimmed cells, is more than justified by the increased accuracy.

Torus example In our next example, we compute the volume of a torus given implicitly by the trimming function

$$\tau(x,y,z) = -((x-0.5)^2+(y-0.5)^2+(z-0.5)^2+R-r)^2+4R((x-0.5)^2+(y-0.5)^2).$$

Figure 9.9 shows the result of the approximation of τ by linear functions in each cell after five steps of refinement. As in the previous example, we employ a $2 \times 2 \times 2$-point Gauss rule for the trivariate quadrature in LT and a 2×2-point Gauss rule for the bivariate quadrature in the correction term for CLT. Figure 9.10 depicts the quadrature error for various values of h for LT, CLT, and IC. This example confirms that the previous observations also apply to non-convex (and hence more complicated) domains.

9.5.2. L^2 fitting and isogeometric analysis

We apply our method to the problems of L^2-projection onto the trimmed B-Spline basis and Galerkin projection of the Poisson problem on a trimmed domain $\Omega_\tau \subset \mathbb{R}^3$, where the trimming surface is given implicitly by a function $\tau : \Omega \to \mathbb{R}$.

In addition to the numerical quadrature, another important challenge in the numerical treatment of trimmed domains is the stability of the basis. If we used the trimmed B-Spline basis

$$\tilde{B}_i(x,y,z) = \begin{cases} \hat{B}_i(x,y,z) & \text{if } \tau(x,y,z) > 0 \\ 0 & \text{else} \end{cases} \tag{9.11}$$

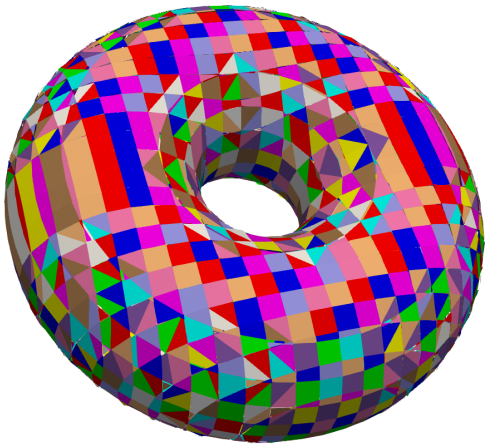


Figure 9.9.: Linear approximation of the torus after some steps of refinement

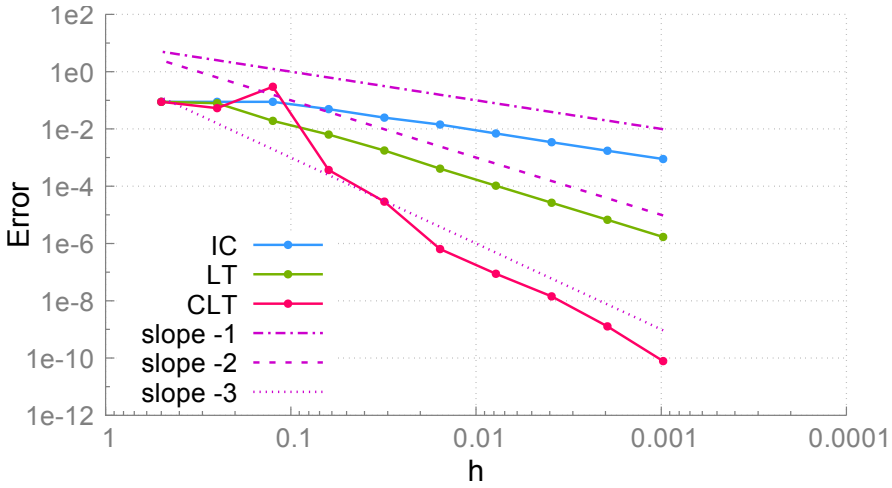


Figure 9.10.: Approximation of the volume of the torus using the LT and CLT.

for isogeometric analysis or for L^2 projection, then the supports $\text{supp}\tilde{B}_i \cap \Omega_\tau$ might be arbitrarily small, depending on how the trimming surface cuts each element of the discretization. In order to address the loss of stability caused by this fact, we use the extended B-Spline basis [42].

Basis functions whose support is smaller than a given threshold (e.g. less than an entire element) are eliminated and added to $(p+1)^d$ stable basis functions with appropriate weights, thereby maintaining the property of polynomial reproduction. This is usually implemented as a post-processing step of the full system matrix. More precisely, The stable system matrix $\hat{A}$ and right-hand-side $\hat{b}$ are given by

$$\hat{A} = EAE^T, \qquad \hat{b} = Eb,$$

where the rectangular matrix E contains the extension weights (for details see [42]) and A and b are the system matrix and right-hand-side related to the trimmed B-splines (9.11).

Example: L^2-projection We apply our quadrature method to the assembly of the mass matrix in order to perform an L^2-projection of the function

$$u(x, y, z) = \sin(\pi xyz) + \cos(2\pi(y + z)) \tag{9.12}$$

into the space of trimmed splines on the unit cube trimmed with a ball of radius $r = 0.23$ around one of the vertices. Its trimming function is given by

$$\tau(x, y, z) = x^2 + y^2 + z^2 - 0.23^2.$$

The solution is shown in Figure 9.11.

We use extended B-splines in order to arrive at a stable discretization. Figure 9.12 shows the h-dependence of the L^2-error for several polynomial degrees p. We achieve the optimal rates of convergence.

However, it should be noted that one obtains similar results when using LT or even IC. Indeed, the resulting L^2 approximation is not very sensitive with respect to perturbations of the domain boundary.

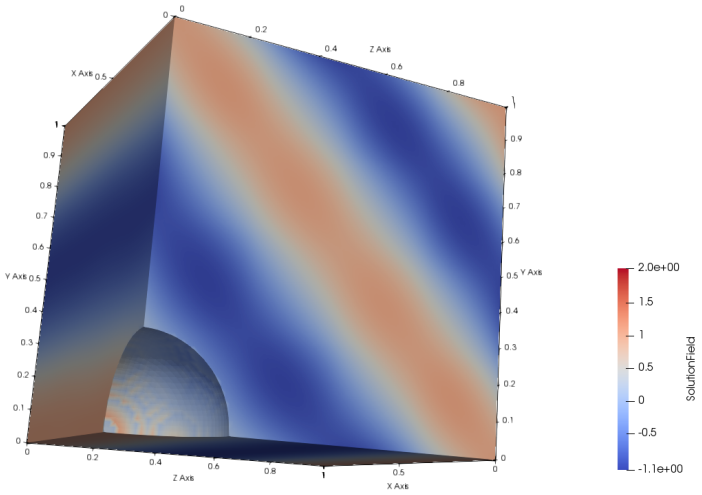


Figure 9.11.: Exact solution (9.12) on the trimmed cube.

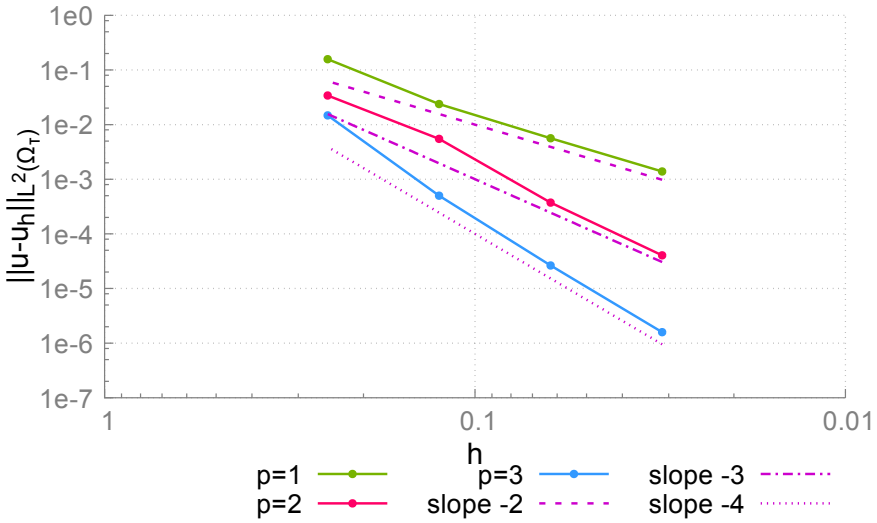


Figure 9.12.: L^2-projection, stabilized using extended B-splines

Example: Poisson equation on cube with spherical cavity Next, we use our quadrature rules to assemble the stiffness matrix in order to solve the Poisson equation

$$\begin{cases} \Delta u &= f \text{ in } \Omega_\tau, \\ \frac{\partial u}{\partial \nu} &= h \text{ on } \partial\Omega_\tau^{\text{Neumann}}, \\ u &= g \text{ on } \partial\Omega_\tau^{\text{Dirichlet}} \end{cases} \tag{9.13}$$

using Galerkin projection, based on extended B-splines.

We use Gauss quadrature with $(p+1)^3$ nodes for the element-wise quadrature on the untrimmed elements and the linear approximations to the trimmed elements. We also use $(p+1)^2$ Gauss nodes for the bivariate integral in the error correction term for CLT.

The domain is the unit cube, which has been trimmed by a ball of radius $r = 0.23$ around the center, thus defining the trimming function

$$\tau(x, y, z) = (x - 0.5)^2 + (y - 0.5)^2 + (z - 0.5)^2 - 0.23^2.$$

We impose Dirichlet boundary conditions on the boundary of the cube and homogeneous Neumann boundary conditions on the spherical trimming surface. As a manufactured solution fulfilling these boundary conditions we choose

$$\begin{aligned} u(x, y, z) = &\frac{10(x - \frac{1}{2})}{\sqrt{(x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 + (z - \frac{1}{2})^2}} \\ &+ (\tau(x, y, z))^2(8\cos(2\pi(y + z)) + 5\sin(2\pi xy)). \end{aligned} \tag{9.14}$$

The first term of u is constant on the lines through the center of the domain, thus fulfilling homogeneous Neumann boundary conditions on every sphere. The second term as well as its normal derivative vanishes on the trimming surface, which is given as the level set $\tau = 0$. Figure 9.13 shows the manufactured solution on the domain $\Omega\tau$ which was clipped by a plane through its center for the visualization.

First we consider the error measured in the H^1-seminorm, see Figure 9.14, which can be analyzed with the help of Strang's lemma [88]. It is bounded by the sum of

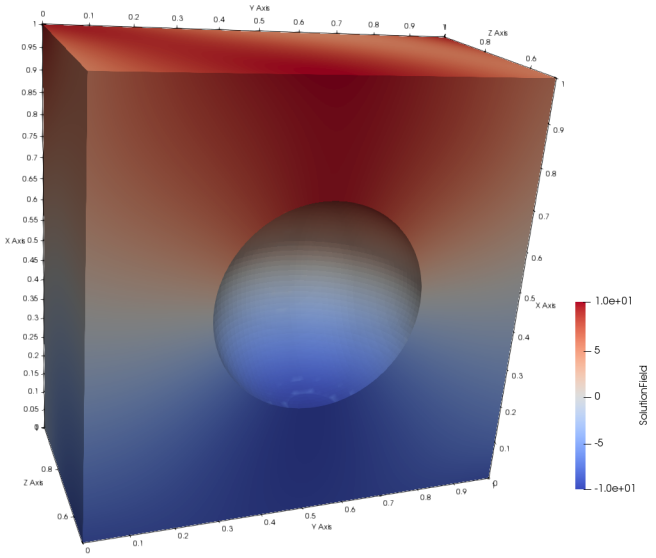


Figure 9.13.: The example solution (9.14) on the trimmed domain clipped with a plane through the origin (only for the visualization).

the approximation error (caused by the discretization) and the consistency error (due to the numerical integration).

For quadratic spline discretizations ($p = 2$), the approximation error (with respect to the H^1-seminorm) converges with order two, and hence it suffices to use a quadrature method that provides this level of accuracy. Nevertheless, the CLT provides a small improvement, compared to LT.

For cubic spline discretizations ($p = 3$), the approximation error (with respect to the H^1-seminorm) converges with order three, and hence one needs to employ quadrature method that provides the same accuracy. The plot clearly demonstrates that CLT maintains the overall accuracy, while LT does not.

Second we consider the error measured in the L^2-norm, see Figure 9.15. For quadratic spline discretizations ($p = 2$), we obtain the full rate of convergence (order 3) when using CLT. However, the use of LT does not provide the full rate, even though it did suffice for optimal (quadratic) H^1 seminorm convergence.

The situation is different for cubic splines ($p = 3$): Here, CLT gives only a (sub-optimal) cubic convergence rate, and an even lower accuracy is obtained when using LT. We conclude that the accuracy of the quadrature does not suffice to extend the optimal rate of convergence with respect to the H^1 seminorm (which

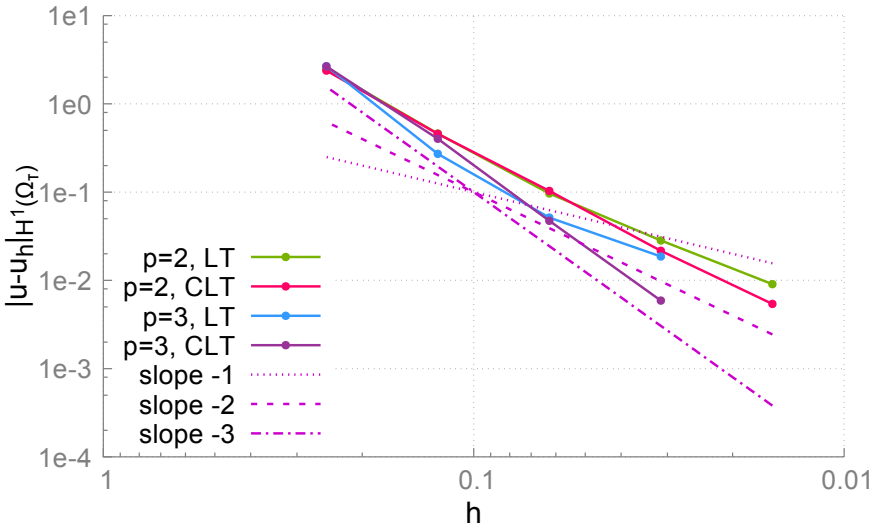


Figure 9.14.: h-dependence of the error of the Galerkin approximation in the H^1-seminorm.

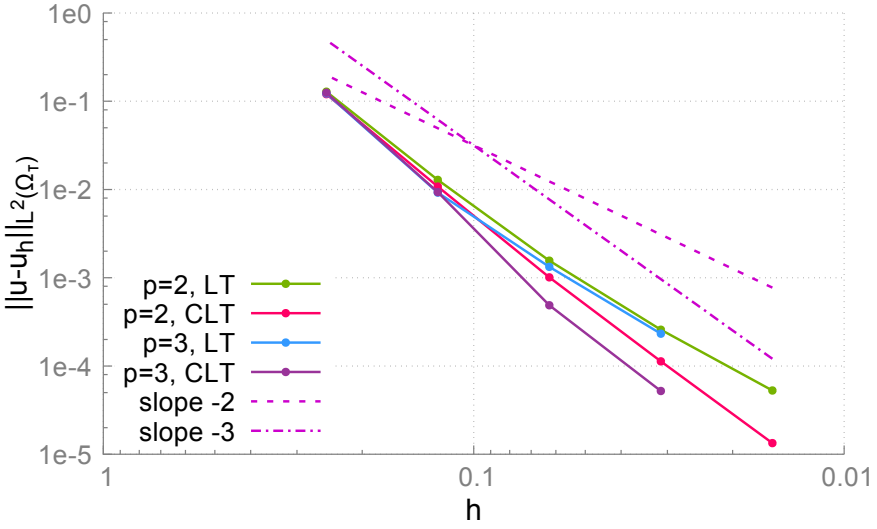


Figure 9.15.: h-dependence of the error of the Galerkin approximation in the L^2-norm.

was guaranteed by Strang's lemma and demonstrated in Figure 9.14) to the L^2 norm. A similar phenomenon was observed in [62], where a higher accuracy of the quadrature was required for even degree splines in order to achieve the optimal rate of convergence with respect to the L^2 norm.

We summarize the experimentally observed rates of convergence with respect to the H^1 seminorm and the L^2 norm (separated by /) in the following table:

degree	LT	CLT
$p = 2$	$\mathbf{2}/2$	$\mathbf{2}/\mathbf{3}$
$p = 3$	$2/2$	$\mathbf{3}/3$

Optimal rates are shown in bold. We conclude that CLT is ideally suited for quadratic discretizations, and beneficial for cubic ones.

In Figure 9.16, we see that, as is expected, the error appears predominantly on the trimming surface, which is where most of the consistency error occurs. However, for linear and quadratic spline discretizations, the error is more evenly distributed than for the cubic discretization. Again, this indicates that the accuracy of the trimmed quadrature – which is used near the trimming surface only – is still insufficient to obtain the optimal rate of convergence for cubic spline discretizations.

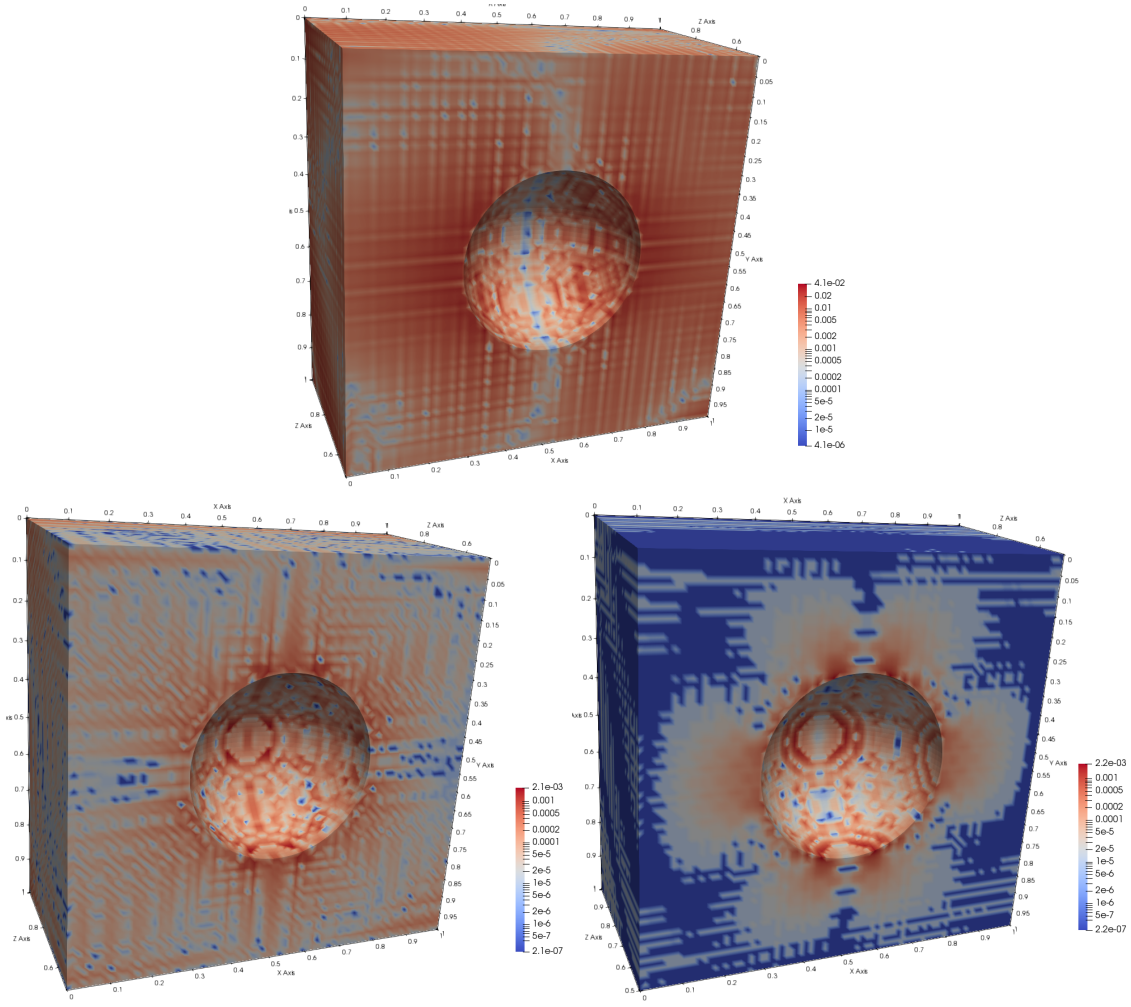


Figure 9.16.: The absolute error $|u(x, y, z) - u_h(x, y, z)|$ for $p = 1$ (top), $p = 2$ (left) and $p = 3$ (right) after five steps of refinement using CLT for the assembly. Note that the scales are different.

10. Future work

An obvious question for future work related to this thesis is how to combine the content of the Parts A and B of this thesis. Since the presence of trimming in a model that is to be used in isogeometric analysis destroys the tensor-product structure of the discretization, the tensor decomposition method presented in Chapter 4 cannot immediately used to speed up the matrix assembly in this case. However, in a space-time problem like the one solved in Section 5.2, trimming is typically only present in the spatial dimensions and not in the time direction. Thus, the two methods can be easily combined in order to arrive at an efficient method for matrix assembly for this class of problems.

A similar limitation of tensor decomposition methods in isogeometric analysis stems from the need for local refinement. Adaptive spline constructions, such as THB-Splines [29], LR B-Splines [44] or T-Splines [10] do not possess a global tensor-product structure, even in one direction. Here, further research is necessary in order to improve the efficiency of the matrix assembly step.

Yet another challenging task is to extend the treatment of space-time methods by partial tensor decomposition that was presented in Section 5.2 to problems with discontinuous diffusion coefficients. Depending of the shape of the discontinuities in space-time, it might be impossible to approximate the coefficients in a tensor-product space with sufficient accuracy to ensure the right convergence order of the consistency error. For the treatment of more general discontinuities, the space-time cylinder could be described as a multi-patch space-time domain (see [57]) that is compatible with the discontinuities of the diffusion coefficient. The low-rank approximation would be applied patch-wise and discontinuous Galerkin techniques would be needed for coupling the local patch-wise problems.

The method for numerical quadrature on trimmed domains presented in Chapter 7 can be generalized in several ways. One open question is how to apply the method

to the integration on level-set surfaces. Dirichlet boundary conditions on trimming surfaces are usually imposed weakly, for example using Nitsche's method. To this end, an integral term over the boundary is added to the bilinear form. Moreover, boundary integrals appear naturally when imposing non-homogeneous Neumann boundary conditions. This makes it necessary to perform numerical quadrature on the trimming surface.

Another direction for future research is the application of the method to higher-order spline discretizations. In order to do this, it is natural to add further terms to the Taylor expansion (9.7). Our conjecture is that each term results in an additional order of convergence of the quadrature error.

Bibliography

[1] C. Adam, T. Hughes, S. Bouabdallah, M. Zarroug, and H. Maitournam. Selective and reduced numerical integrations for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:732–761, 2015.

[2] P. Antolin, A. Buffa, F. Calabrò, M. Martinelli, and G. Sangalli. Efficient matrix computation for tensor-product isogeometric analysis: The use of sum factorization. *Computer Methods in Applied Mechanics and Engineering*, 285:817–828, 2015.

[3] P. Antolin, A. Buffa, and M. Martinelli. Isogeometric Analysis on V-reps: first results, 2019. arXiv:1903.03362.

[4] F. Auricchio, L. Beirão da Veiga, T. J. R. Hughes, A. Reali, and G. Sangalli. Isogeometric collocation methods. *Mathematical Models and Methods in Applied Sciences*, 20(11):2075–2107, 2010.

[5] F. Auricchio, F. Calabrò, T. Hughes, A. Reali, and G. Sangalli. A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 249-252:15–27, 2012.

[6] M. Bachmayr and W. Dahmen. Adaptive low-rank methods: Problems on sobolev spaces. *SIAM Journal on Numerical Analysis*, 54(2):744–796, 2016.

[7] K. Bandara, T. Rüberg, and F. Cirak. Shape optimisation with multiresolution subdivision surfaces and immersed finite elements. *Computer Methods in Applied Mechanics and Engineering*, 300(Supplement C):510 – 539, 2016.

[8] M. Bartoň and V. M. Calo. Optimal quadrature rules for odd-degree spline spaces and their application to tensor-product-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 305:217 – 240, 2016.

[9] M. Bartoň and V. M. Calo. Gauss–Galerkin quadrature rules for quadratic and cubic spline spaces and their application to isogeometric analysis. *Computer-Aided Design*, 82:57 – 67, 2017.

[10] Y. Bazilevs, V. Calo, J. Cottrell, J. Evans, T. Hughes, S. Lipton, M. Scott, and T. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5):229 – 263, 2010.

[11] G. Beer, B. Marussig, and J. Zechner. A simple approach to the numerical simulation with trimmed CAD surfaces. *Computer Methods in Applied Mechanics and Engineering*, 285:776–790, 2015.

[12] M. Bercovier and I. Soloveichik. Overlapping non matching meshes domain decomposition method in isogeometric analysis, 2015.

[13] S. P. A. Bordas, E. N. Burman, M. G. Larson, and M. A. Olshanskii, editors. *Geometrically Unfitted Finite Element Methods and Applications*. Lecture Notes in Computational Science and Engineering. Springer International Publishing, 2017.

[14] A. Buffa and G. Sangalli, editors. *IsoGeometric Analysis: A New Paradigm in the Numerical Approximation of PDEs*. Springer International Publishing, 2016.

[15] E. Burman. Ghost penalty. *Comptes Rendus Mathematique*, 348(21):1217 – 1220, 2010.

[16] E. Burman, P. Hansbo, and M. G. Larson. A cut finite element method with boundary value correction. *Mathematics of Computation*, 87:633–657, 2018.

[17] F. Calabrò, G. Sangalli, and M. Tani. Fast formation of isogeometric Galerkin matrices by weighted quadrature. *Computer Methods in Applied Mechanics and Engineering*, 316:606 – 622, 2017.

[18] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, and V. Calo. The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers. *Computer Methods in Applied Mechanics and Engineering*, 213-216:353 – 361, 2012.

[19] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, Chichester, England, 2009.

[20] D. A. Cox, T. W. Sederberg, and F. Chen. The moving line ideal basis of planar rational curves. *Computer Aided Geometric Design*, 15(8):803 – 827, 1998.

[21] L. Di Gaspero. Quadprog++. https://github.com/liuq/QuadProgpp.

[22] S. Dolgov and B. Khoromskij. Simultaneous state-time approximation of the chemical master equation using tensor product formats. *Numerical Linear Algebra with Applications*, 22:197–219, 2015.

[23] S. V. Dolgov, B. N. Khoromskij, and I. V. Oseledets. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the fokker–planck equation. *SIAM Journal on Scientific Computing*, 34(6):A3016–A3038, 2012.

[24] D. Elfverson, M. G. Larson, and K. Larsson. CutIGA with basis function removal. *Advanced Modeling and Simulation in Engineering Sciences*, 5(1):6, Mar 2018.

[25] B. Erik, C. Susanne, H. Peter, L. M. G., and M. André. CutFEM: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 104(7):472–501, 2014.

[26] M. H. et al. An overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[27] W. Gander and W. Gautschi. Adaptive quadrature—revisited. *BIT Numerical Mathematics*, 40(1):84–101, Mar 2000.

[28] D. Garcia, M. Bartoň, and D. Pardo. Optimally refined isogeometric analysis. *Procedia Computer Science*, 108:808 – 817, 2017.

[29] C. Giannelli, B. Jüttler, and H. Speleers. THB–splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29:485–498, 2012.

[30] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27:1–33, 1983.

[31] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[32] H. Gomez and L. D. Lorenzis. The variational collocation method. *Computer Methods in Applied Mechanics and Engineering*, 309:152 – 181, 2016.

[33] M. Griebel and H. Harbrecht. Approximation of bi-variate functions: singular value decomposition versus sparse grids. *IMA Journal of Numerical Analysis*, 34(1):28–54, 2014.

[34] W. Hackbusch. *Tensor spaces and numerical tensor calculus*. Springer, Berlin, 2012.

[35] R. R. Hiemstra, F. Calabrò, D. Schillinger, and T. J. Hughes. Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:966–1004, 2017.

[36] M. Hillman, J. Chen, and Y. Bazilevs. Variationally consistent domain integration for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:521–540, 2015.

[37] C. Hofer, U. Langer, M. Neumüller, and I.Toulopoulos. Time-multipatch discontinuous Galerkin space-time isogeometric analysis of parabolic evolution problems. *RICAM Report*, (26), 2017.

[38] C. Hofer, U. Langer, and I. Toulopoulos. Discontinuous Galerkin isogeometric analysis of elliptic diffusion problems on segmentations with gaps. *SIAM Journal on Scientific Computing*, 38(6):A3430–A3460, 2016.

[39] C. Hofer and I. Toulopoulos. Discontinuous Galerkin isogeometric analysis for parametrizations with overlapping regions. *RICAM Report*, 17, 2017.

[40] C.-Y. Hu, T. Maekawa, N. Patrikalakis, and X. Ye. Robust interval algorithm for surface intersections. *CAD Computer Aided Design*, 29(9):617–627, 1997.

[41] T. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 199(5 – 8):301–313, 2010.

[42] K. Höllig, U. Reif, and J. Wipper. Weighted extended b-spline approximation of dirichlet problems. *SIAM Journal on Numerical Analysis*, 39(2):442–462, 2001.

[43] C.-K. Im and S.-K. Youn. The generation of 3D trimmed elements for NURBS-based isogeometric analysis. *International Journal of Computational Methods*, 15(7), 2018.

[44] K. A. Johannessen, T. Kvamsdal, and T. Dokken. Isogeometric analysis using LR B-splines. *Computer Methods in Applied Mechanics and Engineering*, 269:471 – 514, 2014.

[45] B. Jüttler, U. Langer, A. Mantzaflaris, S. E. Moore, and W. Zulehner. Geometry + Simulation Modules: Implementing Isogeometric Analysis. *PAMM*, 14(1):961–962, 2014.

[46] B. Jüttler, A. Mantzaflaris, R. Perl, and M. Rumpf. On numerical integration in isogeometric subdivision methods for PDEs on surfaces. *Computer Methods in Applied Mechanics and Engineering*, 302:131 – 146, 2016.

[47] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. Hughes. An immersogeometric variational framework

for fluid-structure interaction: Application to bioprosthetic heart valves. *Computer Methods in Applied Mechanics and Engineering*, 284:1005 – 1053, 2015. Isogeometric Analysis Special Issue.

[48] S. Kargaran, B. Jüttler, S. K. Kleiss, A. Mantzaflaris, and T. Takacs. Overlapping multi-patch structures in isogeometric analysis. *NFN Technical Report*, 82:1–34, 2019.

[49] V. Khoromskaia and B. N. Khoromskij. Tensor numerical methods in quantum chemistry: From Hartree-Fock to excitation energies. *Physical Chemistry Chemical Physics*, 17:31491–31509, 2015.

[50] E. Kieri, C. Lubich, and H. Walach. Discretized dynamical low-rank approximation in the presence of small singular values. *SIAM Journal on Numerical Analysis*, 54(2):1020–1038, 2016.

[51] H. Kim, Y. Seo, and S. . Youn. Isogeometric analysis for trimmed CAD surfaces. *Computer Methods in Applied Mechanics and Engineering*, 198(37-40):2982–2995, 2009.

[52] S. Kleiss, C. Pechstein, B. Jüttler, and S. Tomar. IETI - Isogeometric Tearing and Interconnecting. *Computer Methods in Applied Mechanics and Engineering*, 247-248:201 – 215, 2012.

[53] L. Kudela, N. Zander, S. Kollmannsberger, and E. Rank. Smart octrees: Accurately integrating discontinuous functions in 3d. *Computer Methods in Applied Mechanics and Engineering*, 306:406 – 426, 2016.

[54] S. Kumar and D. Manocha. Efficient rendering of trimmed NURBS surfaces. *Computer-Aided Design*, 27(7):509 – 521, 1995. Display and visualisation.

[55] O. A. Ladyzhenskaya. *The Boundary Value Problems of Mathematical Physics*. Springe, New York, 1985.

[56] O. A. Ladyzhenskaya, V. A. Solonnikov, and N. N. Uraltseva. *Linear and Quasilinear Equations of Parabolic Type*. AMS, Providence, RI, 1968.

[57] U. Langer, A. Mantzaflaris, S. E. Moore, and I. Toulopoulos. Multipatch discontinuous Galerkin isogeometric analysis. In B. Jüttler and B. Simeon, editors, *Isogeometric Analysis and Applications IGAA 2014*, volume 107 of *Lecture Notes in Computer Science*, pages 1–32, Heidelberg, 2015. Springer.

[58] U. Langer, S. Moore, and M. Neumüller. Space-time isogeometric analysis of parabolic evolution equations. *Comput. Methods Appl. Mech. Engrg.*, 306:342–363, 2016.

[59] C. Lehrenfeld. High order unfitted finite element methods on level set domains using isoparametric mappings. *Computer Methods in Applied Mechanics and Engineering*, 300:716 – 733, 2016.

[60] C. Lubich, I. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM Journal on Numerical Analysis*, 53(2):917–941, 2015.

[61] A. Mantzaflaris and B. Jüttler. Exploring matrix generation strategies in isogeometric analysis. In M. Floater et al., editors, *Mathematical Methods for Curves and Surfaces*, volume 8177 of *Lecture Notes in Computer Science*, pages 364–382. Springer, 2014.

[62] A. Mantzaflaris and B. Jüttler. Integration by interpolation and look-up for Galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 284:373–400, 2015.

[63] A. Mantzaflaris, B. Jüttler, B. Khoromskij, and U. Langer. Matrix generation in isogeometric analysis by low rank tensor approximation. In J.-D. Boissonnat et al., editors, *Curves and Surfaces*, volume 9213 of *LNCS*, pages 321–340. Springer, 2015.

[64] A. Mantzaflaris, B. Jüttler, B. N. Khoromskij, and U. Langer. Low rank tensor methods in Galerkin-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:1062 – 1085, 2017.

[65] A. Mantzaflaris, F. Scholz, and others (see website). G+Smo (Geometry plus Simulation modules) v0.8.1. http://gs.jku.at/gismo, 2017 - 2019.

[66] A. Mantzaflaris, F. Scholz, and I. Toulopoulos. Low-rank space-time decoupled isogeometric analysis for parabolic problems with varying coefficients. *Computational Methods in Applied Mathematics*, 19.1:123–136, 2018.

[67] B. Marussig, R. Hiemstra, and T. Hughes. Improved conditioning of isogeometric analysis matrices for trimmed geometries. *Computer Methods in Applied Mechanics and Engineering*, 334:79–110, 2018.

[68] B. Marussig and T. Hughes. A review of trimming in isogeometric analysis: Challenges, data exchange and simulation aspects. *Archives of Computational Methods in Engineering*, pages 1–69, 2017.

[69] B. Marussig, J. Zechner, G. Beer, and T. Fries. Stable isogeometric analysis of trimmed geometries. *Computer Methods in Applied Mechanics and Engineering*, 316:497–521, 2017.

[70] F. Massarwi, P. Antolin, and G. Elber. Volumetric untrimming: Precise decomposition of trimmed trivariates into tensor products. *Computer Aided Geometric Design*, 71:1 – 15, 2019.

[71] A. Nagy and D. Benson. On the numerical integration of trimmed isogeometric elements. *Computer Methods in Applied Mechanics and Engineering*, 284:165–185, 2015.

[72] T. S. Newman and H. Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854 – 879, 2006.

[73] N. Patrikalakis and T. Maekawa. *Shape interrogation for computer aided design and manufacturing*. Springer, Berlin, Heidelberg, 2002.

[74] L. A. Piegl and W. Tiller. *The NURBS book*. Springer-Verlag Berlin Heidelberg, 1997.

[75] E. Rank, M. Ruess, S. Kollmannsberger, D. Schillinger, and A. Düster. Geometric modeling, isogeometric analysis and the finite cell method. *Computer Methods in Applied Mechanics and Engineering*, 249-252:104–115, 2012.

[76] T. Rüberg and F. Cirak. A fixed-grid b-spline finite element technique for fluid-structure interaction. *International Journal for Numerical Methods in Fluids*, 74(9):623–660, 2014.

[77] M. Ruess, D. Schillinger, A. Özcan, and E. Rank. Weak coupling for isogeometric analysis of non-matching and trimmed multi-patch geometries. *Computer Methods in Applied Mechanics and Engineering*, 269:46–71, 2014.

[78] R. Sanches, P. Bornemann, and F. Cirak. Immersed b-spline (i-spline) finite element method for geometrically complex domains. *Computer Methods in Applied Mechanics and Engineering*, 200(13):1432 – 1445, 2011.

[79] G. Sangalli and M. Tani. Isogeometric preconditioners based on fast solvers for the Sylvester equation. *SIAM Journal on Scientific Computing*, 38(6):A3644–A3671, 2016.

[80] D. Schillinger, J. Evans, A. Reali, M. Scott, and T. Hughes. Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations. *Computer Methods in Applied Mechanics and Engineering*, 267:170–232, 2013.

[81] R. Schmidt, R. Wüchner, and K.-U. Bletzinger. Isogeometric analysis of trimmed NURBS geometries. *Computer Methods in Applied Mechanics and Engineering*, 241-244:93–111, 2012.

[82] F. Scholz and B. Jüttler. Numerical integration on trimmed three-dimensional domains. *NFN Technical Report*, 85:1–18, 2019.

[83] F. Scholz, A. Mantzaflaris, and B. Jüttler. Partial tensor decomposition for decoupling isogeometric Galerkin discretizations. *Computer Methods in Applied Mechanics and Engineering*, 336:485 – 506, 2018.

[84] F. Scholz, A. Mantzaflaris, and B. Jüttler. First order error correction for trimmed quadrature in isogeometric analysis. In *Advanced Finite Element Methods with Applications*, Lecture Notes in Computational Science and Engineering. Springer International Publishing, 2019.

[85] L. Schumaker. *Spline Functions: Basic Theory*. Cambridge University Press, 3 edition, 2007.

[86] A. Seiler and B. Jüttler. Reparameterization and adaptive quadrature for the isogeometric discontinuous Galerkin method. In *Mathematical Methods for Curves and Surfaces*, pages 251–269, Cham, 2017. Springer International Publishing.

[87] Y.-D. Seo, H.-J. Kim, and S.-K. Youn. Isogeometric topology optimization using trimmed spline surfaces. *Computer Methods in Applied Mechanics and Engineering*, 199(49-52):3270–3296, 2010.

[88] G. Strang. Approximation in the finite element method. *Numerische Mathematik*, 19:81–98, 1972.

[89] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.

[90] Y. Wei, G. Wang, and P. Yang. Legendre-like orthogonal basis for spline space. *Computer-Aided Design*, 45(2):85 – 92, 2013. Solid and Physical Modeling 2012.

[91] M. Woźniak, M. Paszyński, D. Pardo, L. Dalcin, and V. M. Calo. Computational cost of isogeometric multi-frontal solvers on parallel distributed memory machines. *Computer Methods in Applied Mechanics and Engineering*, 284:971 – 987, 2015.

[92] S. Xia and X. Qian. Isogeometric analysis with Bézier tetrahedra. *Computer Methods in Applied Mechanics and Engineering*, 316:782 – 816, 2017.

[93] H. Zhang, R. Mo, and N. Wan. An IGA discontinuous Galerkin method on the union of overlapped patches. *Computer Methods in Applied Mechanics and Engineering*, 326:446–480, 2017.

[94] X. Zhu, Z. Ma, and P. Hu. Nonconforming isogeometric analysis for trimmed CAD geometries using finite-element tearing and interconnecting algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 231(8):1371–1389, 2017.

Eidestattliche Erklärung Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Dissertation ist mit dem elektronisch übermittelten Textdokument identisch.

Felix Scholz