



TECHNISCHE UNIVERSITÄT MÜNCHEN
TUM School of Engineering and Design

Efficient and robust quadrature for embedded solids:
Application to isogeometric analysis and shape optimization

Manuel Meßmer

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr.-Ing. Fabian Duddeck

Prof. Dr. Daniel Straub

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Kai-Uwe Bletzinger
2. Prof. Dr.-Ing. Stefan Kollmannsberger
3. Prof. Riccardo Rossi, Ph.D.

Die Dissertation wurde am 17.04.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 24.06.2024 angenommen.

Schriftenreihe des Lehrstuhls für Statik
TU München

Band 66

Manuel Meßmer

EFFICIENT AND ROBUST QUADRATURE FOR EMBEDDED SOLIDS:
APPLICATION TO ISOGOMETRIC ANALYSIS AND SHAPE OPTIMIZATION

München 2024

Veröffentlicht durch

Kai-Uwe Bletzinger
Lehrstuhl für Statik
Technische Universität München
Arcisstr. 21
80333 München

Telefon: +49(0)89 289 22422
Telefax: +49(0)89 289 22421
E-Mail: kub@tum.de
Internet: www.bgu.tum.de/st/startseite/

ISBN: 978-3-943683-77-6

© Lehrstuhl für Statik, TU München



für meine Eltern

Abstract

This cumulative dissertation presents a robust framework to generate analysis-ready Finite Element (FE) models from arbitrarily complex solid geometries in STereoLithography (STL) format. Using the concept of embedded boundary methods, these models are tailored to efficiently solve structural problems on unfitted discretizations. The open-source program [Quadrature for Embedded Solids – QuESo](#) combines all developments into a seamless interface between Computer-Aided Design (CAD) and FE software. In the proposed workflow, QuESo reads the STL file from CAD and outputs a list of integration points that represent the geometrical domain within a simple computational grid. As a result, the communication between QuESo and the FE program is essentially limited to a set of active elements with customized integration points.

Special attention is given to the construction of highly efficient quadrature rules, which extend the scope of application beyond linear and nonlinear static examples to efficient transient simulations, including explicit dynamic analyses. A point elimination strategy with non-negative moment fitting quadrature deals with emerging cut domains. This results in at most $n = (p+1)^3$ integration points per element, which all lie within the physical domain and feature strictly positive integration weights. In addition, the group-wise evaluation of interior/uncut elements yields nearly optimal quadrature rules for higher-continuous basis functions, e.g., B-Splines or NURBS, such that n decreases significantly below $(p+1)^3$. The second core development is a robust geometry kernel that efficiently processes arbitrarily complex models. This includes a mesh intersection algorithm, which drastically simplifies the assembly of the moment fitting equations, and a scheme for element classification that delivers robust results even for severely flawed geometries. All algorithms are successfully applied to nearly 5000 STLs of varying complexity. Finally, a novel sensitivity analysis paves the way for the integration of shape optimization strategies into the simulation process of embedded methods. Tracking potentially large shape updates by an Eulerian description avoids mesh distortion problems, which pose major challenges when using boundary-fitted discretizations.

In summary, the proposed developments drastically facilitate the model conversion process from CAD to FEM while guaranteeing a high-quality discretization. Moreover, all studied examples can straightforwardly be solved with higher-order and higher-continuous B-Spline bases.

Kurzfassung

Die vorliegende kumulative Dissertation stellt ein robustes Framework vor, um aus beliebig komplexen Festkörpergeometrien im STL-Format brauchbare Finite-Elemente (FE) Modelle zu erzeugen. Diese sind darauf zugeschnitten, Strukturprobleme auf nicht angepassten Diskretisierungen unter der Verwendung von eingebetteten Randmethoden effizient zu lösen. Das Open-Source-Programm [Quadrature for Embedded Solids – QuESo](#) vereint sämtliche Entwicklungen zu einer nahtlosen Schnittstelle zwischen Computer-Aided Design (CAD) und FE-Software. Im vorgeschlagenen Arbeitsablauf liest QuESo die STL-Dateien aus dem CAD ein und berechnet eine Liste von Integrationspunkten, die den geometrischen Bereich innerhalb eines einfachen Berechnungsgitters darstellen. Die Kommunikation zwischen QuESo und der FE-Software beschränkt sich daher im Wesentlichen auf eine Reihe aktiver Elemente mit angepassten Integrationspunkten.

Ein besonderes Augenmerk gilt der Konstruktion hocheffizienter Quadraturregeln, wodurch sich der Anwendungsbereich über lineare und nichtlineare statische Beispiele hinaus auf transiente Simulationen, einschließlich expliziter dynamischer Analysen, erweitert. Eine Punkteliminierungsstrategie mit nicht-negativer Momenten-Anpassungs-Quadratur befasst sich mit den entstehenden geschnittenen Bereichen. Dies führt zu höchstens $n = (p + 1)^3$ Integrationspunkten pro Element, welche allesamt innerhalb des physikalischen Bereichs liegen und positive Integrationsgewichte aufweisen. Darüber hinaus liefert die gruppenweise Auswertung der inneren (nicht geschnittenen) Elemente nahezu optimale Quadraturregeln für stetige Basisfunktionen höherer Ordnung, z.B. B-Splines oder NURBS, so dass n deutlich unter $(p + 1)^3$ abnimmt. Die zweite zentrale Entwicklung ist ein robuster Geometrie Kern, der die effiziente Verarbeitung beliebig komplexer Modelle ermöglicht. Dazu gehören ein Algorithmus zur Netzverschneidung, der die Momentenberechnung erheblich vereinfacht, sowie ein Elementklassifizierungsverfahren, das selbst bei stark fehlerbehafteten Geometrien robuste Ergebnisse liefert. Alle Algorithmen werden erfolgreich auf fast 5000 STLs unterschiedlicher Komplexität angewendet. Schließlich ebnet eine neuartige Sensitivitätsanalyse den Weg für die Integration von Formoptimierungsstrategien in den Simulationsprozess von eingebetteten Methoden. Die dadurch ermöglichte Eulersche Betrachtungsweise der potenziell großen Formaktualisierungen vermeidet Netzverzerrungsprobleme, die eine große Herausforderung bei Lagrangeschen Diskretisierungen darstellen.

Zusammenfassend lässt sich festhalten, dass die vorgestellten Entwicklungen den Prozess der Modellkonvertierung von CAD zu FEM erheblich vereinfachen und gleichzeitig eine hohe Diskretisierungsqualität gewährleisten. Zudem können alle untersuchten Beispiele problemlos mit B-Spline-Basen höherer Ordnung und höherer Stetigkeit gelöst werden.

Acknowledgements

The last years have been an exciting journey that has shaped me both professionally and personally. I would like to take this opportunity to thank everyone who has contributed to an instructive, successful, and joyful time.

Prof. Kai-Uwe Bletzinger has certainly been a key figure on this path. First and foremost, I would like to thank him for giving me the chance to conduct my research at the Chair of Structural Analysis. I greatly appreciated the supervision, academic freedom, and guidance that came with it during my entire time as a PhD candidate. My gratitude extends to Prof. Roland Wüchner, who opened the door for me at TUM and has been a valuable mentor ever since. I would like to express my appreciation for the countless professional and personal discussions that significantly contributed to the success of my dissertation. During my six months at CIMNE in Barcelona, Prof. Riccardo Rossi became an indispensable adviser to all kinds of technical questions. He contributed significantly to my programming and implementation skills, without which this work would not have been possible. I would also like to thank Prof. Stefan Kollmannsberger for the numerous scientifically valuable and always joyful exchanges. Since my time as a research assistant at the Chair for Computation in Engineering, he has constantly helped to orient myself in the academic world, for which I am truly grateful. Besides my academic advisors, special thanks go to my industry partners, Lukas Leidinger and Stefan Hartmann. Their interest in my work and enthusiasm to solve real-world problems were an invaluable source of motivation for me.

The amazing time at the Chair of Structural Analysis was predominantly defined by the many colleagues I was fortunate enough to meet and work with. Each day's highlight was the regular coffee breaks with Klaus Sautter and Martin Fußeder, where any frustration could be turned into laughter. I am happy to have shared an office with Shahrokh Shayegan, Aditya Ghantasala, Reza Najian Asl, Carlos Lázaro, and Talhah Ansari. Many thanks for the great atmosphere and the numerous discussions on technical solutions and private matters. I am sincerely grateful to all my colleagues and would particularly like to mention Veronika Singer and Tobias Teschemacher, who helped me find my way around the chair when I was still a newbie.

Above all, I would like to thank my family and my partner for always supporting me.

Manuel Meßmer
Technical University of Munich
June, 2024

This cumulative dissertation includes three peer-reviewed research papers, which are listed below.

Publication I

M. Meßmer, T. Teschemacher, L.F. Leidinger, R. Wüchner, K.-U. Bletzinger, Efficient CAD-integrated isogeometric analysis of trimmed solids, *Computer Methods in Applied Mechanics and Engineering* 400 (2022) 115584, <http://dx.doi.org/10.1016/j.cma.2022.115584>.

Publication II

M. Meßmer, S. Kollmannsberger, R. Wüchner, K.-U. Bletzinger, Robust numerical integration of embedded solids described in boundary representation, *Computer Methods in Applied Mechanics and Engineering* 419 (2024) 116670, <https://doi.org/10.1016/j.cma.2023.116670>.

Publication III

M. Meßmer, R. Najian Asl, S. Kollmannsberger, R. Wüchner, K.-U. Bletzinger, Shape optimization of embedded solids using implicit Vertex-Morphing, *Computer Methods in Applied Mechanics and Engineering* 426 (2024) 116999, <https://doi.org/10.1016/j.cma.2024.116999>.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Research aim and objectives	4
1.3	Outline	6
2	Quadrature for Embedded Solids – QuESo	9
2.1	General scope of application	9
2.2	Framework	9
2.2.1	Workflow	10
2.2.2	CAD-integration and STL models	13
2.2.3	Requirements for the finite element solver	15
2.2.4	Finite element applications	17
3	Scientific context	19
3.1	Scientific contributions	19
3.1.1	Publication I: Efficient quadrature for structural analysis	19
3.1.2	Publication II: Robust geometry processing	21
3.1.3	Publication III: Shape optimization	22
3.2	Critical discussion	23
3.2.1	Construction of the unfitted finite element model	23
3.2.2	Performance of the unfitted finite element model	26
4	Conclusion and outlook	31
4.1	Conclusion	31
4.2	Outlook	32
5	Publication I	35
6	Publication II	87
7	Publication III	115

Bibliography

137

Introduction

The following sections discuss the motivation behind this thesis, state the overarching research aim and the specific intermediate objectives, and provide a brief document outline.

1.1 Motivation

The development process for parts, assemblies, or entire structures in modern engineering departments classically comprises a design and analysis stage. This is seldom a linear procedure but requires several iterations, where the current design is optimized in an iterative loop. As a result, the geometric and structural models must be exchanged multiple times between the designer and the simulation engineer. Over the decades, the two disciplines have developed largely independently of each other and produced individual software packages with inherently different model descriptions. The probably most important concepts for the design and analysis of structures are Computer-Aided Design (CAD) and the Finite Element Method (FEM). However, while CAD aims at an accurate, flexible, and adaptable representation of the geometry, the FEM's main objective is to capture physical phenomena. Modern CAD models are usually described by B-Splines or Non-Uniform Rational B-Splines (NURBS), which enable parameterized free-form geometries. The traditional FEM, on the other hand, relies on discrete geometrical descriptions, so-called meshes. Therefore, in order to perform structural analysis using FEM, a mesh must first be derived from the NURBS-based CAD model. A study estimates that this model conversion, also called meshing, costs 80% of the time of the overall engineering process [1].

Hughes et al. [2] introduced the concept of Isogeometric Analysis (IGA) as an approach to eliminate this often tedious, error-prone, and labor-intensive meshing step from the outset. The fundamental difference between standard FEM and IGA is the use of different shape functions. While FEM traditionally uses low-order polynomials, IGA exploits the CAD's native B-Spline and NURBS, which are usually characterized by higher polynomial degrees and higher continuities. The main intention is to perform Finite Element Analysis (FEA) directly on the

NURBS-based CAD geometries, potentially bridging the gap between design and analysis. Due to these features, IGA was expected to have the following two advantages, among others.

- (A.1) The elimination of the need for model conversion from CAD to FEA or its considerable facilitation.
- (A.2) The numerical exploitation of the higher polynomial orders and continuities of NURBS and B-Splines.

Although the differences between FEM and IGA appear marginal at first glance, the realization of a complete IGA workflow is far more complex than simply substituting the relevant shape functions. It involves specific IGA element formulations [3], the treatment of multi-patches [4], the weak imposition of constraints [5], the numerical integration of trimmed domains [6], and different pre- and postprocessing routines [7]. Despite the above challenges, CAD-integrated simulation tools that utilize the concept of IGA have been successfully implemented. However, the modeling concept used in most CAD environments limits these workflows to thin-walled, shell-like structures. For them, the CAD program directly offers the necessary parameterization to explore the advantages (A.1)-(A.2). By contrast, a CAD-integrated simulation including volumetric geometries poses further difficulties. The idea of using the unmodified solid CAD model or a refined version of it as the basis for subsequent structural analyses is simply not feasible. Modern CAD software relies on the so-called Boundary-Representation (B-Rep), where the geometry is solely described through its outer skin. Fig. 1.1 illustrates this concept using the example of a simple cylinder with a hole. It becomes apparent that four faces compose the model. However, no volumetric function space of the interior is provided, which is essential for the structural analysis.

As mentioned above, the classical approach to tackle this problem is the generation of a boundary-fitted FE mesh with, e.g., hexahedral or tetrahedral elements that explicitly describe the geometry. More recent studies propose to derive trivariate IGA-type discretizations by reconstructing the original B-Rep with multiple volumetric NURBS patches, as in [8, 9]. Nevertheless, since this does not correspond to the inherent modeling concept of CAD, the necessary challenge of domain segmentation, i.e., meshing, remains. In addition, the inevitable occurrence of multiple patches creates coupling surfaces, which usually do not preserve the continuity properties of the respective splines. Consequently, the desired advantages (A.1)-(A.2) are only partially given.

Embedded/immersed boundary methods, on the other hand, approach the model conversion problem totally differently, avoiding the need for a boundary-fitted mesh altogether. Their main philosophy is to restrict the discretization to a simple computational mesh, e.g., a regular grid with hexahedral-shaped elements, which overlaps the entire geometrical domain. Due to this simplification, the solid's outer skin inevitably intersects certain elements. Therefore, one of the main challenges of embedded boundary methods is constructing quadrature rules that represent the resulting cut domains. Prominent examples of embedded boundary methods are: the Cut Finite Element Method (CutFEM) [10, 11], the Aggregated Finite Element Method (AgFEM) [12, 13], the Cartesian Grid Finite Element Method (cg-FEM) [14, 15], and the Finite Cell Method (FCM) [16,

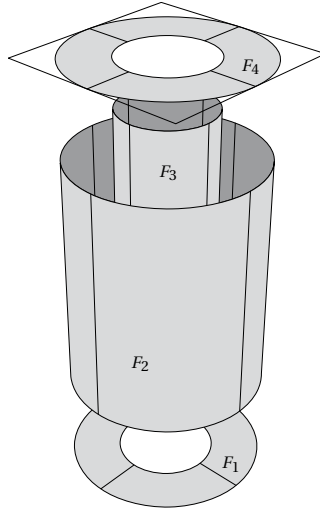


Fig. 1.1: Computer-aided design boundary representation model of a cylinder with a hole (taken from Publication 1).

17]. As demonstrated in [18], the simplicity of the underlying computational grid enables the straightforward use of higher-order and higher-continuous bases and thus fulfills (A.2). However, the extent to which embedded boundary methods can improve industrial and academic simulation workflows, as envisioned in (A.1), remains to be seen. We recall that in most development processes, the initial design provides solely the structure's outer skin, which is represented by multiple NURBS and B-Spline patches. A design-through-analysis workflow must, therefore, be able to deal with this particular model description or at least with variants of it that standard CAD programs can deliver with reasonable effort. Regarding embedded boundary methods, the main task for creating an analysis-ready finite element model is no longer the generation of a boundary-fitted mesh but the construction of suitable integration rules. These ultimately represent the potentially complex geometrical domain within the simple computational grid. However, most existing quadrature methods exhibit one or multiple of the following shortcomings that can limit the fully automated, robust, and CAD-integrated creation of efficient models.

- (S.1) Cut/trimmed elements contain a huge number of integration points.
- (S.2) Integration points are located outside the physical domain.
- (S.3) Integration weights are negative.
- (S.4) The approximation of the geometry is very coarse.
- (S.5) The scheme is restricted to implicitly defined geometries, e.g., in the form of level sets.

- (S.6) Although the scheme can generally be applied to solids explicitly described in boundary representation,
- (a) it is not robust against arbitrarily complex geometries.
 - (b) it can not handle flawed geometries.
 - (c) the generation of integration points is slow (significantly slower than solving the finite element problem).

1.2 Research aim and objectives

This work's overarching aim is to develop a complete framework for generating efficient finite element models from arbitrarily complex CAD geometries that can solve structural problems on unfitted discretizations. These developments are intended to fulfill the expectations of (A.1)-(A.2) and, simultaneously, eliminate all limitations (S.1)-(S.6) most embedded boundary methods, at least partially, suffer from. For this purpose, a robust and efficient workflow that provides unconditionally high-quality FE discretizations independent of the input model's complexity is essential. Moreover, the framework should enable a seamless interface between any state-of-the-art CAD program and FE solver with as little effort as possible. The specific research objectives required to realize the aspired workflow are listed below.

Objective I: Efficient quadrature rules

Due to the existence of arbitrarily complex integration domains, embedded boundary methods classically require significantly more integration points than boundary-fitted finite element methods. This results in an unnecessarily high number of function evaluations during the assembly of the local system matrices (implicit FEA) or the internal force vector (explicit FEA).

(a) Cut/trimmed elements

A key characteristic of the standard FEM is to mesh the geometry with only a few different types of elements. This allows reusing the same implementations and predefined integration rules, i.e., Gauss points, for any geometry. By definition, embedded boundary methods accept unrestricted intersection patterns between the elements and the geometric boundary. Therefore, suitable integration rules must evaluate functions over arbitrarily complex domains, typically leading to quadrature constructions with considerably more points than with Gaussian quadrature. This work aims at integration rules that provide sufficient accuracy to maintain optimal convergence rates in, e.g., the energy norm, with a minimum number of points. Furthermore, these rules should feature positive integration weights and point locations limited to the material domain.

(b) Patch-wise integration

As discussed above, the simple structure of the computational mesh, i.e., regular grid, enables the straightforward use of higher-order and higher-continuous basis functions, such as B-Splines and NURBS. It is well known

that these bases achieve higher continuity by reaching over multiple element/knot spans. This raises the question of whether generalized rules with fewer points, which exploit the continuity property, can substitute classical element-wise/knot span-wise Gaussian integration. Our objective is to develop a generalized patch-wise quadrature scheme that improves the efficiency of embedded boundary methods with higher-continuous bases.

Objective II: Robust algorithms for processing complex B-Reps

Aiming at a fully CAD-integrated simulation workflow requires us to deal with geometric descriptions that modern CAD programs can readily provide without using additional third-party tools. Therefore, a suitable standardized and CAD native data format is essential. The second goal of this work is a computational geometry kernel that robustly processes arbitrarily complex models provided by the CAD environment. We address the following crucial tasks necessary in embedded boundary methods.

(a) Element classification

The first step in any embedded boundary method is classifying elements into interior/uncut elements, cut elements, and exterior elements. Correct classification is paramount for an analysis-ready simulation model since a single falsely assessed element can lead to singular system matrices. However, this task is particularly challenging because standard B-Rep models are not always *watertight* but can contain small gaps or overlaps, etc., due to mathematical inaccuracies. Our objective is the development of a classification scheme that provides robust results for arbitrarily complex B-Reps, including flawed geometries.

(b) Domain reconstruction for cut elements

As discussed in Objective I (a), the numerical integration of cut domains is one of the main difficulties in embedded boundary methods and has hence been the focus of many scientific contributions. Most of their efforts are devoted to the actual construction of the quadrature rules. Each method, however, also requires a geometric parameterization to perform the necessary integrals or at least a set of geometric operations. These are rarely addressed in the literature or simply outsourced to third-party computational geometry libraries. Although many existing libraries offer state-of-the-art algorithms, they are not designed for our specific problems, resulting in inefficient and error-prone workflows. We aim at a geometry kernel tailored to the needs of the methods developed in Objective I (a), potentially accelerating runtimes and increasing robustness. This includes an algorithm to reconstruct the domains of each cut element, which provides the necessary parameterization for appearing bulk and surface integrals. The respective developments should facilitate the highly accurate calculation of properties such as the volume of a cut element and enable efficient geometrical operations, e.g., inside-outside tests.

Objective III: Sensitivity analysis

Besides developing a framework to facilitate the integration of embedded boundary methods into industrial and academic workflows, this thesis also attempts to extend the toolbox with shape optimization strategies. Gradient-based optimizers require sensitivities with respect to the parameters that govern the structure's shape. However, the implicit geometry description of embedded methods impedes the use of classical approaches developed for boundary-fitted discretizations. Our goal is a sensitivity analysis that can serve as a natural interface between embedded boundary methods and shape optimization algorithms.

Objective IV: Applications

The final objective of this work is to demonstrate the potential of the envisioned framework for various structural applications, which are listed and briefly discussed below.

(a) CAD-integrated static and transient analysis using B-Spline bases

Static structural analysis is arguably the most popular field of application for embedded methods. We aim to show that our framework provides a seamless interface for a CAD-integrated simulation workflow that can exploit the advantages of B-Spline basis functions. In addition, the quadrature schemes pursued in Objective I are expected to pave the way towards efficient transient analysis using embedded boundary methods. Our particular focus is on explicit time integration schemes, where the prevailing computational cost scales linearly with the total number of integration points. In this context, the performance of the developed quadrature rules is to be assessed. Finally, we want to investigate the influence of cut elements on the critical explicit time step.

(b) Application to arbitrarily complex and flawed B-Reps

A substantial part of this work is dedicated to geometrical operations that appear in the workflow of embedded boundary methods, as discussed in Objective II. Extensive tests with complex geometries will demonstrate the applicability of the developed algorithms for industrial and scientific workflows. In addition, the robustness against flawed, *non-watertight* models is to be investigated.

(c) Shape optimization

A common issue of using boundary-fitted discretizations in shape optimization methods is mesh distortion due to large shape updates. The sensitivity analysis followed in Objective III will be used to optimize the shape of immersed solid geometries. Embedded methods are expected to enable an Eulerian description of the shape updates, thus avoiding mesh distortion problems.

1.3 Outline

This cumulative dissertation features three peer-reviewed research papers, hereafter referred to as Publications I-III. Before they are included in Chapters 5-7, a general context is established. Chapter 2 introduces the global framework,

explains its general scope of application, and highlights the individual components developed. The presentation of the overall workflow also guides the reader by providing direct references to the respective publications. Chapter 3 discusses their scientific contributions, emphasizes the literature most relevant for understanding this work, and closes with a critical discussion. Chapter 4 concludes the present work and provides a concise outlook.

Quadrature for Embedded Solids – QuESo

This chapter briefly describes the general scope of application and introduces the overall framework developed.

2.1 General scope of application

The present work deals with structural finite element problems solved on unfitted discretizations, which are clearly distinguished from classical boundary-fitted meshes. In particular, we consider embedded/immersed boundary methods that fulfill the following criteria.

1. The computational mesh is a simple regular grid (not necessarily uniform) that is composed of hexahedral integration domains, i.e., finite elements or knot spans.
2. The geometric boundary is embedded into the above-specified mesh and allowed to intersect elements.
3. The cut domains are exclusively captured by modified quadrature rules, whereas the function space remains unchanged.

All methods meeting the above requirements are referred to as embedded boundary methods or just embedded methods in the following. Furthermore, we denote the computational grid as background mesh to avoid confusion with other meshes appearing in the workflow. Analogously, the respective elements may also be referred to as background elements. If B-Splines or NURBS serve as basis functions, their knot spans are regarded as elements.


2.2 Framework



The presented framework provides a *plug-and-play* solution for the integration of embedded boundary methods into existing scientific and industrial sim-

ulation workflows. This is an attempt to make the advantages of these technologies accessible to a broader audience. Therefore, all methodologies and interfaces are implemented in an open-source library named QuESo, which is publicly available at:

Github repository: QuESo

URL: <https://github.com/manuelmessmer/QuESo>

Language:  C++

Communication:  Python interface or  file exchange

The following sections explain how QuESo can be used to realize a design-through-analysis workflow and emphasize the adaptations necessary in the CAD and FE environment. This should explain the scope of this work but also serve as a guide for potential users. The above repository includes an interface to the open-source FE framework Kratos Multiphysics [19, 20, 21].

2.2.1 Workflow

QuESo is designed as an interface/preprocessor that seamlessly connects any CAD software and FE solver with minimal effort. The main idea is to read the input geometry provided by CAD and output an analysis-ready embedded simulation model that standard FE solvers can handle. Fig. 2.1 illustrates this concept schematically. The references integrated in Fig. 2.1, e.g., P. I, which is short for Publication I, should help to understand the scope of the papers included in Chapters 5-7.

In order to guarantee an easy integration into existing workflows, generic data formats and interfaces are of central importance. Regarding the geometric information, our workflow exploits the standardized StereoLithography (STL) file format, which provides a direct link to all modern CAD programs. An STL is a simplified B-Rep model, representing the structure's skin by an oriented triangle mesh. For a more detailed discussion about STLs and the integration of QuESo into a CAD environment, the reader is kindly referred to Section 2.2.2. A schematic input file illustrating the necessary settings, processed by the QuESo input interface in Fig. 2.1, is given below.

File 1: QuESo settings

```
"geometry_settings" : {
    "input_filename" : "geometry.stl",
    ...
},
"background_mesh_settings" : {
    "element_size" : [hx, hy, hz],
```

```

    "polynomial_degree" : [px, py, pz],
    ...
  },
  "quadrature_rule_settings" : {
    ...
  },
  "boundary_conditions" : [{
    "condition_id" : 1,
    "input_filename" : "bc_geometry_1.stl",
    ...
  }, {
    "condition_id" : 2,
    "input_filename" : "bc_geometry_2.stl",
    ...
  }],
  ...
]

```

This input deck allows the user to define the STL file representing the structure's geometry and to specify the sizes and polynomial degrees associated with the finite elements to be utilized. Based on the given information, QuESo constructs a grid-like background mesh that encloses the entire geometric domain, as depicted in Fig. 2.2. Subsequently, the framework robustly identifies all active elements and constructs integration points to efficiently assemble their local system matrices. Since the underlying algorithms are designed to operate exclusively on the geometric boundary, an STL, as depicted in Fig. 2.2, provides sufficient information for the given operations. In the context of embedded methods, active elements refer to elements that are fully contained within the geometrical domain (interior elements) or intersected by its boundary (cut elements). All other elements are inactive (exterior elements) and do not contribute to the global system matrices. Publication 1 discusses different quadrature rules, which can be specified in File 1. Due to the implicit geometry description in embedded methods, the imposition of boundary conditions requires weak formulations, which usually take the form of boundary integrals. Besides the bulk integration rules discussed above, QuESo also provides quadrature points tailored for integrals along the respective surfaces. Fig. 2.3 illustrates the complete input and output data of the proposed interface. Note that Fig. 2.3 is a detailed view of the QuESo module in Fig. 2.1. In analogy to the global geometric model, the sections of the boundary that serve as supports or are subjected to an external load are also defined as STLs, see File 1. The surface integration points are extracted from the triangular mesh. However, since finite elements are defined as piece-wise polynomials, the corresponding integrals must be evaluated in an element-wise decomposition. Thus, QuESo derives surface integration points from triangles that conform to the element boundaries in the background mesh, as highlighted in Fig. 2.3.

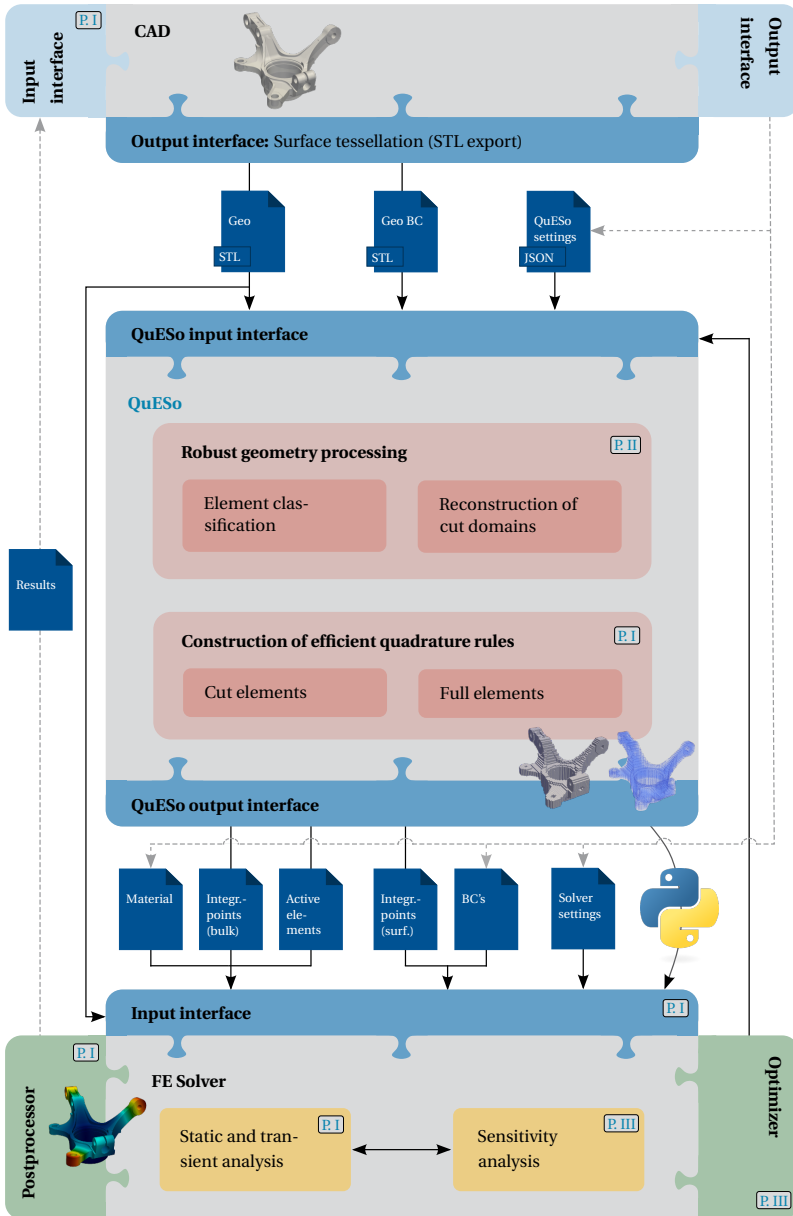


Fig. 2.1: Overall QuESO workflow with interfaces to CAD and FE solver. This figure uses the following abbreviations: Boundary Condition - BC and Geometry - Geo.

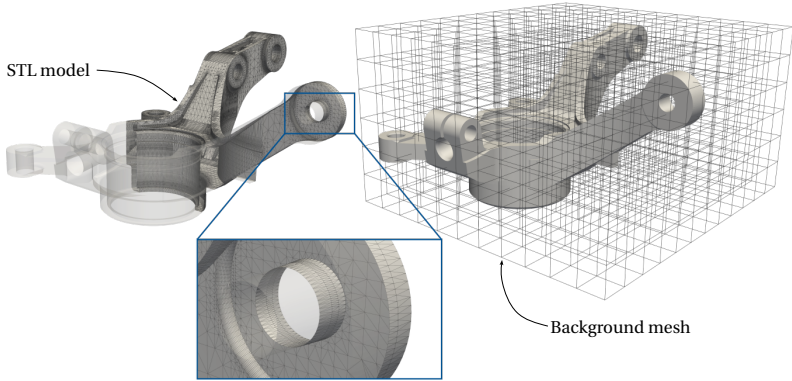


Fig. 2.2: Embedded STL model.

In summary, QuESo reads the information given in File 1, processes the geometries, and outputs an analysis-ready unfitted finite element model, as depicted in Fig. 2.3. This model contains a list of bulk integration points associated with the set of active elements, as well as surface integration points for the boundary conditions. As shown in Fig. 2.1, Publications I-II present the key components necessary for the respective model conversion. This includes the construction of efficient quadrature rules and the robust processing of complex geometries.

For a subsequent finite element analysis, the only data that must be communicated to the FE program by a customized interface are the active elements and the bulk and surface integration points. QuESo supports two different options. The corresponding data transfer can be realized either through file exchange or QuESo's Python interface. All other information, such as solver settings, material properties, and boundary condition values, e.g., external forces or prescribed displacements, are part of the standard input deck of any FE program. The necessary and optional adaptations within the FE solver to realize the overall workflow depicted in Fig. 2.1 are discussed in Section 2.2.3. Note that the dashed connections in Fig. 2.1 are not essential and are only required to drive the FE analysis directly from CAD, which is outlined in the next section.

2.2.2 CAD-integration and STL models

Standard CAD systems represent solid structures solely by their delimiting surfaces, i.e., their outer skin. These surfaces are, in turn, described by several trimmed NURBS or B-Spline patches, which leads to complex data structures that differ in each CAD program. The workflow presented in Section 2.2.1 operates on oriented triangle meshes derived from the original NURBS-based B-Rep models. Several file formats, e.g., STL, PLY, OFF, OBJ, etc., can provide such boundary tessellations. However, due to its broad acceptance and availability,

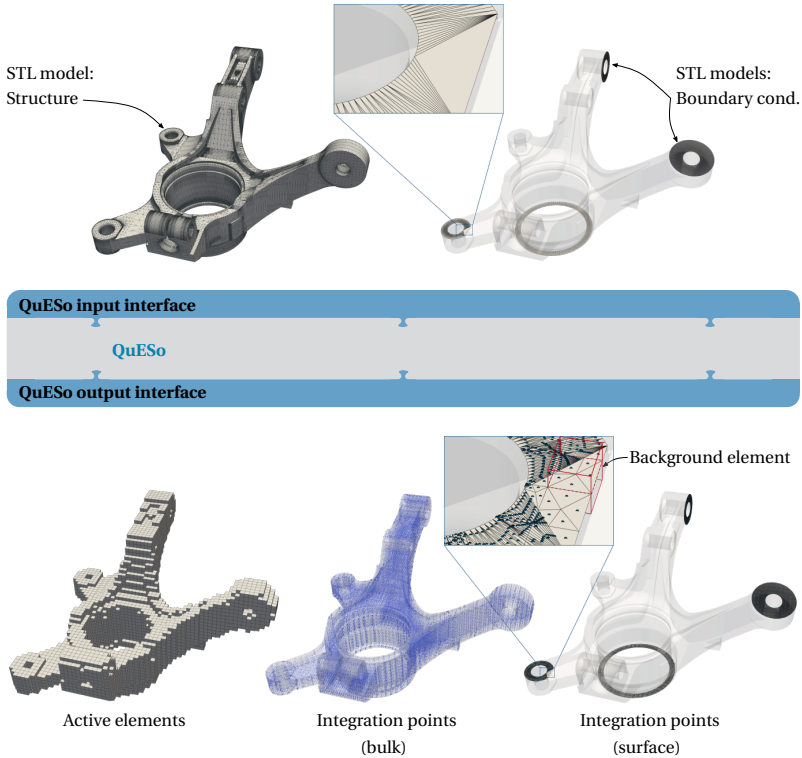


Fig. 2.3: QuESo as design-through-analysis interface: Illustration of input and output data.

the following discussion focuses on the STL format, whereby the term STL can be understood as a synonym for oriented triangle mesh.

In comparison to NURBS-based B-Rep models, STLs have two distinct advantages. Firstly, their generic file format provides a native interface to any CAD environment. In addition, the simple data structure enables a drastic acceleration of all necessary geometrical operations. For example, intersection tests can be defined as closed forms for STLs but require iterative methods, e.g., the Newton-Raphson method, when dealing with NURBS surfaces. Naturally, these benefits come at the cost of a certain loss of accuracy. By definition, STLs are discrete models that only approximate the exact geometry by simple triangles. Nevertheless, in the workflow depicted in Fig. 2.1, the STL *only* serves as a geometric representation to derive integration points. As a result, classical mesh quality criteria, such as element aspect ratios, etc., have no significance. The only objective of the respective tessellation is an accurate geometry description. This allows the utilization of entirely unstructured meshes with tiny, acute triangles in curved ar-

eas and large elements along plane sections, yielding relatively efficient boundary representations. Moreover, a fine STL model may affect QuESo's preprocessing time, i.e., the construction of quadrature rules, but it does not influence the computational cost of the finite element solver. It is also important to emphasize that unlike standard boundary-fitted meshes, where volumetric elements are unavoidable for the analysis of solid structures, the STL only represents their outer skin. QuESo features algorithms that can efficiently process large STLs. This includes models where the geometrical error is orders of magnitude smaller than the discretization error of the background mesh and thus lies below the perception level.

Since designers work with NURBS-based B-Rep models and not with STLs, an intermediate tessellation is inevitable to realize the CAD-integrated simulation workflow in Fig. 2.1. This may appear to be an additional computational overhead, which complicates the whole procedure. However, surface tessellation algorithms are a fundamental feature of every CAD program. In fact, surface meshes help to visualize and render the geometry and are, therefore, often generated or modified in real-time. This means that the STL model required for the overall geometry and the boundary conditions, as depicted in Fig. 2.3, can be retrieved without significant computing effort. In conclusion, QuESo offers a direct interface to any CAD software that does not require any customization.

Moreover, these developments also pave the way for a fully CAD-integrated simulation workflow, where the user does not have to leave the design environment. Such a workflow is implemented as a plugin for the open-source CAD program FreeCAD [22], which involves the following features that essentially draw the dashed lines in Fig. 2.1.

- A graphical user interface to define the structure's geometry, the boundary conditions, the material properties, and the solver settings.
- A customized output interface that automatically exports the STL models, provides the respective QuESo settings in File 1, and writes the input deck for the FE program, e.g., solver settings.
- A controller that invokes QuESo and the utilized FE solver.
- A customized input interface to read the results from the FE solver.
- A process to visualize the structural responses.

2.2.3 Requirements for the finite element solver

The fundamental difference for the FE solver in the proposed workflow is that elements and conditions contain customized integration points. In contrast to standard boundary-fitted meshes, the classically used Gauss points are substituted by the quadrature rules provided by QuESo. This allows the utilization of the same routines, element formulations, and solution strategies initially designed for boundary-fitted meshes. QuESo views the finite elements in the background mesh only as integration domains with a specific function space. Thus, the communication with the FE solver remains the same, regardless of whether bilinear hexahedral elements, higher-order elements, or a trivariate B-Spline patch spans the background mesh. The key prerequisite is that the actual

finite elements or knot spans correspond to QuESo's integration domains. This includes their spatial dimensions, i.e., element sizes, and the polynomial degree of the associated basis functions, which are both specified in File 1. In order to apply essential boundary conditions, the surface integration points depicted in Fig. 2.3 must be processed along with a set of properties that defines the condition's type and, for example, the force value or the prescribed displacement, see Fig. 2.1. As discussed previously, Dirichlet conditions require weak formulations. The simplest strategy is the Penalty method [23], where an artificial stiffness counteracts the violation of the constraint. Other possible candidates are Lagrange multipliers or Nitsche-type methods [5].

A general challenge of embedded methods is ill-conditioned system matrices due to little support of basis functions within small cut elements. Therefore, direct solvers are recommended for solving the linear system of equations. The use of iterative solvers is not impossible but requires specific stabilization techniques or preconditioners for them to converge. A comprehensive overview of solution strategies for systems of equations derived from unfitted discretizations is provided in [24].

As in any finite element framework, the workflow in Fig. 2.1 needs a postprocessor to visualize structural responses such as displacements, stresses, etc. In embedded boundary methods, the degrees of freedom are located at the nodes of the background mesh. Since these can lie outside the material domain, they do not necessarily represent physical values. Therefore, in order to recover physically meaningful responses, the respective fields must be evaluated/interpolated at the immersed geometry, e.g., the vertices of the STL. This information then can be transferred to third-party visualization engines, e.g., ParaView, or fed back into the CAD environment, completing the design-through-analysis workflow, as outlined in Section 2.2.2.

An additional component of this work is to extend the scope of embedded methods to gradient-based shape optimization of complex solid structures. Although this is not a necessary feature for standard structural analysis workflows, its requirements for the FE solver are worth mentioning. As indicated in Fig. 2.1, this predominantly includes a sensitivity analysis suitable for embedded boundaries and an optimizer that computes the respective shape updates. Both methods can be integrated into the FE solver or be implemented as stand-alone applications with suitable interfaces. Note that the sensitivity analysis requires direct access to the response functions of the FE problem. For this reason, it is part of the FE solver in Fig. 2.1. The communication between the sensitivity analysis and the optimizer is limited to gradients stored at the vertices of the embedded geometry, i.e., STL. Simple strategies like the method of steepest descent can drive the optimization process. Gradient projection may enrich the algorithm in order to consider one or multiple constraints [25, 26]. Note that these are standard optimization approaches that are also employed for boundary-fitted meshes. The sensitivity analysis, on the other hand, is more involved and must be tailored to embedded boundary methods. A novel strategy is developed in Publication III.

All features discussed above are realized in the open-source finite element framework Kratos Multiphysics.

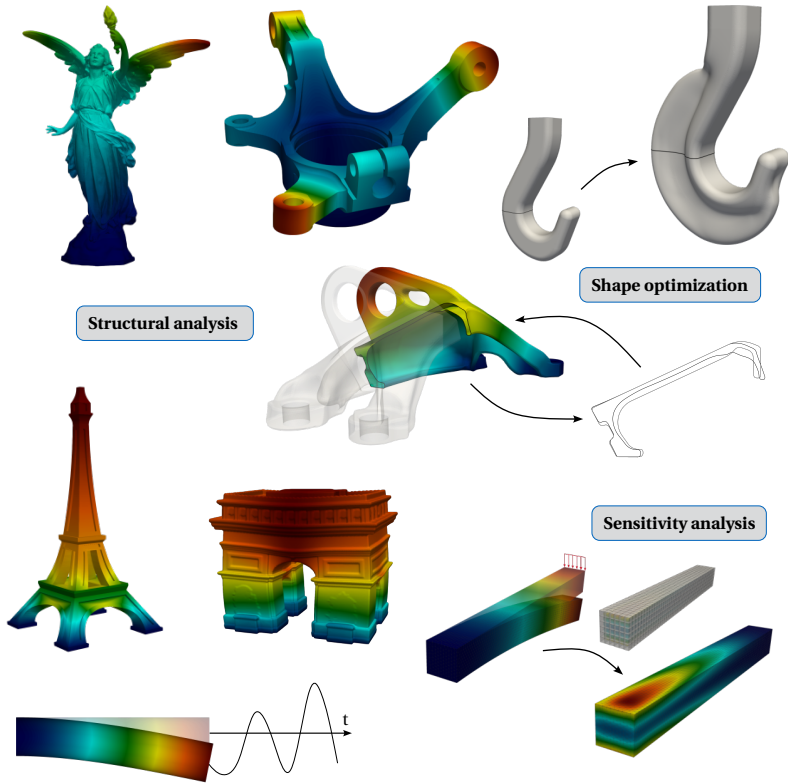


Fig. 2.4: Illustration of finite element problems solved with the proposed workflow.

2.2.4 Finite element applications

QuESo is designed to generate efficient finite element models that can be applied to various structural problems. In summary, the workflow in Fig. 2.1 applies to linear and nonlinear static analyses, as well as transient simulations, including implicit and explicit time integration methods. Additionally, our developments extend the scope of application to shape optimizations, in which common FE reactions of linear static analyses, such as strain energy, are minimized. Regardless of the given problem, standard C^0 continuous finite elements or higher-continuous B-Spline bases can be employed in the background mesh. This work demonstrates the potential of the proposed framework in all scenarios mentioned above, exploiting the distinguishing features of QuESo: The robust, efficient, and fully automatized generation of high-quality finite element

discretizations for geometries of arbitrary complexity. Fig. 2.4 shows an illustrative collection of applications tackled with the presented workflow. All simulations in this work are performed with Kratos Multiphysics. In [27], QuESo is used to solve structural problems with embedded domain methods in LS-DYNA [28].

Scientific context

This section discusses the scientific context of the framework introduced in Chapter 2. Three peer-reviewed publications, included in Chapters 5-7, present the underlying developments. The following section gives a brief overview of their individual contributions with references to the research objectives stated in Section 1.2. Please note that the respective summaries list only the key papers on which our developments are directly based. A detailed literature review can be found in the introductions of Publications I-III. Section 3.2 concludes with a critical discussion.

3.1 Scientific contributions

All contributions discussed in this section are developed by the doctoral candidate in a leading role. Contributions by other authors that go beyond supervision, paper revision and editing, or conceptual discussions with the doctoral candidate are explicitly marked as such.

3.1.1 Publication I: Efficient quadrature for structural analysis

M. Meßmer, T. Teschemacher, L.F. Leidinger, R. Wüchner, K.-U. Bletzinger, Efficient CAD-integrated isogeometric analysis of trimmed solids, *Computer Methods in Applied Mechanics and Engineering* 400 (2022) 115584, <http://dx.doi.org/10.1016/j.cma.2022.115584>.

Included in Chapter 5.

Publication I addresses the research Objectives I and IV (a). The respective contributions are discussed below.

Objective I: Efficient quadrature rules

A fundamental component of embedded boundary methods is the treatment of cut elements, which inevitably appear, as shown in Fig. 2.2. Space trees and spatial tessellations are popular methods to derive quadrature rules for such irregular domains. However, they often entail numerous integration points, leading to an unnecessarily high number of function evaluations. Moment fitting has proven to be a promising strategy towards more efficient quadrature rules [29, 30, 31]. The main idea is to optimize a set of integration points with respect to certain reference integrals, which typically takes the form of a least squares problem. To solve the moment fitting equations efficiently, Publication I combines components of the point-elimination algorithm proposed in [31, 32] with a Non-Negative Least Squares (NNLS) solver suggested in [33]. These developments guarantee $n \leq (p + 1)^3$ integration points per cut element, regardless of its geometrical complexity, where p is the polynomial degree of the basis functions. Furthermore, all integration points lie within the physical domain and have positive weights.

Remark. Please note that a very similar moment fitting scheme for three-dimensional elements was independently developed in [34]. Both papers were submitted in parallel to the respective journals.

As discussed previously, the simple structure of the background mesh enables the straightforward use of higher-continuous basis functions. When B-Splines are employed, the knot spans classically represent individual integration domains. This interpretation allows the application of standard Gaussian quadrature for all full/uncut knot spans. However, [35] showed that such knot span-wise constructions are no longer optimal for higher-continuous bases in the sense that exact integration rules with fewer points exist. In [36], patch-wise schemes are constructed by tensor products from optimal one-dimensional rules, which are referred to as Generalized Gaussian Quadrature (GGQ) rules. Publication I extends this concept to embedded boundary methods. A novel algorithm decomposes the background mesh into several tensor product domains, which are each integrated as a macro element, i.e., as a group of multiple elements, using GGQ. Both optimal and reduced quadrature constructions are derived and critically assessed. Initial studies suggest that up to $\approx 90\%$ of the integration points can be saved compared to standard element-wise Gaussian quadrature while attaining full-order accuracy in the energy norm. Note that these integration rules apply only to full/uncut knot spans. Cut knot spans are still evaluated using moment fitting, as discussed above.

Objective IV (a): Application to CAD-integrated static and transient analysis using B-Spline bases

Publication I realizes a complete CAD-integrated analysis workflow, including all interfaces marked in Fig. 2.1. The developed integration schemes are successfully applied to static and transient problems, where they demonstrably contribute to more efficient simulations. A common problem of embedded methods is that small cut elements can lead to arbitrarily small critical explicit time steps. Inspired by [37], Publication I shows that the continuity of the B-Splines can help limit the time step to feasible values. *The study on the critical explicit time step*

was initiated by Lukas Leidinger.

3.1.2 Publication II: Robust geometry processing

M. Meßmer, S. Kollmannsberger, R. Wüchner, K.-U. Bletzinger, Robust numerical integration of embedded solids described in boundary representation, *Computer Methods in Applied Mechanics and Engineering* 419 (2024) 116670, <https://doi.org/10.1016/j.cma.2023.116670>.

Included in Chapter 6.

Publication II addresses the research Objectives II and IV (b). The respective contributions are discussed below.

Objective II: Robust algorithms for processing complex B-Reps

Publication I provides a strategy to solve the moment fitting equations for highly efficient integration points suitable for arbitrarily cut domains. This assumes, however, that accurate reference integral solutions, also known as moments, are available for the underlying least squares problem. Naturally, the accuracy of the final quadrature rule is limited by the quality of the calculated moments. Publication I adopts the concept proposed in [38, 39] and converts the arising volume integrals to surface integrals over the respective boundary. In practical terms, this requires a closed surface parameterization of each cut domain, which entails complex Boolean operations. Similar to [40], Publication I outsources this task to a third-party computational geometry library. Our preliminary implementation based on CGAL* showed that the bottleneck of the whole workflow is the necessary geometrical operations, especially for large models. This is certainly not because of inefficient algorithms but due to the focus on fundamentally different applications. Programs like CGAL are designed to perform Boolean operations between two complex-shaped domains. In our particular case, however, the intersections between a potentially complicated solid structure and thousands or millions of elements are of interest. Furthermore, computational geometry libraries classically expect *watertight* geometries that are not always available.

The overarching goal of Publication II is the development of an entire geometry kernel tailored to the needs of embedded boundary methods. Firstly, this includes a robust algorithm for the intersection of the input STL with all elements in the background mesh. The result is a closed surface mesh for each cut domain that facilitates the accurate and efficient computation of the moments. Since the respective parameterizations are exclusively used for integration purposes, standard mesh quality criteria, such as aspect ratios, hanging nodes, etc., are not significant. By avoiding related redundant operations, the algorithms in Publication II achieve extremely fast runtimes.

Another fundamental task of embedded boundary methods is the classification of elements into interior, exterior, and cut domains. Standard methods rely on ray tracing techniques, which can fail in the case of *non-watertight* geometries. As a remedy, [42] suggests categorizing elements as a group rather than

* The Computational Geometry Algorithms Library (CGAL) [41].

individually, whereby the clustering is based on a flood fill algorithm. Publication II generalizes this concept for the application of geometries with arbitrary topology.

Objective IV (b): Application to arbitrarily complex and flawed B-Reps

All algorithms developed in Publication II are intensively tested on nearly 5000 STLs from the Thingi10K database [43], ranging from simple shapes to highly complex models with detailed features. These studies demonstrate their remarkable robustness and low computational cost. Finally, the developed geometry kernel is integrated into the simulation workflow presented in Fig. 2.1, allowing finite element analyses of highly complex models and even flawed geometries.

3.1.3 Publication III: Shape optimization

M. Meßmer, R. Najian Asl, S. Kollmannsberger, R. Wüchner, K.-U. Bletzinger, Shape optimization of embedded solids using implicit Vertex-Morphing, *Computer Methods in Applied Mechanics and Engineering* 426 (2024) 116999, <https://doi.org/10.1016/j.cma.2024.116999>.

Included in Chapter 7.

Publication III addresses the research Objectives III and IV (c). The respective contributions are discussed below.

Objective III: Sensitivity analysis

Lagrangian representations, i.e., boundary-fitted meshes, usually suffer from mesh distortions when exposed to large shape updates during an optimization process. The main goal of Publication III is to integrate gradient-based shape optimization strategies into the workflow in Fig. 2.1, with the aim of eliminating mesh distortion problems. This requires the calculation of shape sensitivities at the vertices of the immersed boundary, i.e., the STL model. However, the implicit geometry description in embedded methods prevents the direct adoption of classical sensitivity analyses, which are employed for boundary-fitted meshes.

Publication III presents a novel strategy to compute sensitivities of standard response functions, such as strain energy, with respect to the shape of embedded solids. The underlying scheme comprises three main steps. First, semi-analytical methods [44] calculate the gradients at the nodes of the background mesh. In the second step, Publication III adopts the sensitivity weighting proposed in [45]. This makes it possible to recover a continuous gradient field, which, in the last step, can be consistently evaluated at any point on the structure's skin. The resulting sensitivities are shown to be independent of the background mesh and the discretization of the STL.

Objective IV (c): Application to shape optimization

As soon as the shape sensitivities are available at the embedded boundary, standard strategies that have proven successful in optimizing shells [46, 47] can be transferred to solid models. Similar to [48], Publication III follows a node-based approach that manipulates each boundary vertex individually. In our

implementation, Rosen's gradient projection algorithm [25, 26] drives the constrained optimization process. Additionally, an implicit Helmholtz/Sobolev-based filter [49] is applied to the sensitivities and shape updates, allowing full control over the feature size in the final design. *The discussed filter was developed and implemented by Reza Najian Asl. He also provided the optimizer used in Publication III.*

Furthermore, Publication III presents an effective technique to enforce geometrical constraints, such as minimum wall thicknesses and design space limitations. Finally, the developed workflow is used to optimize the shape of solid structures with industrial complexity.

3.2 Critical discussion

The following sections critically reflect on the overall developments presented in Publication I-III. This detailed discussion is split into the construction and the performance of the unfitted finite element model. For a concise summary, the reader is referred to Chapter 4.

3.2.1 Construction of the unfitted finite element model

A robust generation of finite element models for arbitrarily complex geometries that can be solved efficiently on unfitted discretizations is the predominant goal of the framework presented in Section 2.2. The most important contributions to its realization are the moment fitting scheme proposed in Publication I and the intersection algorithm developed in Publication II. They are particularly effective in combination, which will be discussed below.

As shown in Publication II, the closed boundary meshes, constructed for each cut element, enable the calculation of the moments with near-machine precision. Compared to an octree, a prevalent method to tackle this task [34, 40], our approach delivers orders of magnitude higher accuracies. Due to the voxel-like approximation of the boundary, octrees can require thousands of integration points per finite element. For each point, an inside/outside test, e.g., using ray tracing, must be performed, resulting in an enormous computational burden. Moreover, dependent on the cut element's shape, even a million integration points do not necessarily attain more than a rough approximation of the respective integrals, such that the relative error in the L2 norm of the moments is often still $r > 10^{-3}$. By contrast, the boundary integrals achieve residuals of $r < 10^{-10}$ with much fewer function evaluations and no single inside/outside test, see Section 6.1 in Publication II. Certainly, the necessary construction of the boundary meshes must also be considered when comparing the two methods. However, the time required to calculate the mesh intersections is nearly negligible, as it is considerably less than for the assembly and the solution of the moment fitting equations, see Section 6.3 in Publication II.

As an ideal complement, Publication I finds a set with no more than $n = (p + 1)^3$ integration points that satisfy the moments calculated above in the least squares sense with residuals in the order of $< 10^{-8}$. Note that $n = (p + 1)^3$ represents the same number used by standard Gaussian quadrature for hexahedral

finite elements. It may seem obvious that a reasonable solution with $n = (p + 1)^3$ points also exists for cut elements. However, this is more complex since cut domains do not represent a tensor product space. In addition, orthogonal polynomials are not necessarily available, which is the fundamental requirement of Gaussian quadrature. Section 3.2.1 in Publication I discusses this topic in detail. Consequently, a rule with $n = (p + 1)^3$ points can not be expected to provide exact results. Nevertheless, Publication I shows that $n \leq (p + 1)^3$ integration points are sufficient to maintain optimal convergence in the energy norm.

In order to linearize the moment fitting equation, most approaches define the locations of the integration points a priori and only solve for their weights. As quadrature rules react very sensitively to the point's locations, this decision is accompanied by a degradation in accuracy. Thus, the point elimination step is crucial for the quadrature scheme presented in Publication I. Its main idea is to start with more points than moments, leading to an initially under-determined system of equations. This gives the algorithm the flexibility to select the most suitable point locations from a discrete set while preserving a linear system of equations. The integrated Non-Negative Least Squares (NNLS) solver not only guarantees positive integration weights but also contributes to extremely fast convergence of the point elimination algorithm. In all studied examples, the first iteration yields at most $n = (p + 1)^3$ points with weights unequal to zero. Since points with a zero weight have no significance for the final quadrature rule, they can be immediately discarded. Depending on the targeted residual of the least squares problem, a few more iterations may eliminate additional points. As a result, the average number of iterations is in the order of < 5 . This is a drastic improvement over schemes where points are eliminated individually from the initial set.

Remark. While Publication I uses a simple scheme that distributes the initial points for the elimination algorithm uniformly within the physical domain, the current implementation in QuESo relies on an adaptive octree with Gauss-Legendre points in each leaf node, as in [34].

An important decision when dealing with moment fitting is the definition of the underlying set of basis functions. Most approaches employ simple polynomials that provide fast function evaluations and a straightforward definition of the anti-derivatives, which are required for the application of the divergence theorem, see Section 3.2.2 in Publication I and Section 3.2 in Publication II. Since no true Gaussian quadrature can be obtained for cut domains due to the reasons mentioned above, a substitution of the moment fitting bases can lead to different integration points. As a result, the respective residuals of the least squares are associated with a particular set of functions. In other words, the obtained residuals are only an estimate of how well the computed quadrature rules perform when applied to the actual finite element shape functions. Therefore, it seems beneficial to prefer *distinct* moment fitting bases, which are challenging to evaluate. This provides residuals that enable a more conservative estimate of the quadrature rule's quality. For example, simple monomials often lead to fewer integration points during point elimination than, e.g., orthogonal Legendre polynomials. These results do not imply that monomials generally provide more efficient quadrature rules but rather that they tend to cause poorly conditioned moment fitting matrices, as reported in [31]. For this reason, all examples studied in the

present work employ orthogonal Legendre polynomials as moment fitting bases. Note that the term orthogonality here refers to the tensor-product space of the uncut background element or an axis-aligned bounding box containing the cut domain. They are not orthogonal on the cut domain itself.

Up to this point, the discussion on the treatment of cut elements mainly focused on measures like the accuracy and the efficiency of the quadrature rules. Another important aspect, however, is the robustness of the overall workflow, which is primarily dictated by the intersection algorithm. We can assume that if the reconstruction of a closed boundary mesh is successful for a given cut domain, a quadrature rule can be derived from it using the proposed methods. Therefore, Publication II gives special attention to the robustness of the developed intersection algorithm. An essential feature in this context is the approach suggested in Section 5.5 in Publication II for estimating the quality of the mesh intersections. It can, for instance, determine whether the calculated parameterization is closed or not. An open surface mesh can imply two different scenarios. Either the intersection algorithm failed due to fluctuations in the floating point operations, or the initial geometric model, i.e., the input STL, is flawed, e.g., contains gaps, overlaps, etc. In order to rule out the first possibility, the intersection algorithm is tried again on a slightly perturbed element. If the boundary mesh is still non-closed after several attempts (<5), we can assume that the STL model contains flaws that intersect the given background element. Section 6.2.1 in Publication II successfully applies the intersection algorithm to nearly 5000 valid/non-flawed STLs from the Thingi10K database [43], where each of them is embedded into a background mesh with more than 10^3 elements. The term successful here means that a closed boundary mesh is constructed for each cut domain. This demonstrates the extraordinary robustness of the algorithm and hence offers a very high level of confidence that a resulting non-closed mesh intersection is due to defects in the initial model. Consequently, the proposed quality measure enables the identification and localization of geometric defects in the input STL. Since the mesh intersections are only used for integration purposes, the moment fitting equation can still be assembled and solved even if the parameterization is not perfectly closed. Naturally, this leads to a certain integration error. However, the developed method can estimate its magnitude and, therefore, provide a reasonable basis for deciding whether or not to include the associated background element in the simulation. In summary, the intersection algorithm delivers precise results for all valid/non-flawed geometries. If the geometry contains defects, the affected background elements can be detected and, if necessary, specially treated. It is important to note that the respective error is, in any case, limited to the vicinity of the geometric defect and does not pollute the overall FE solution.

In fact, geometric flaws are much more critical during the classification of the background mesh into interior, exterior, and cut elements. Imagine that the algorithm falsely categorizes an element that is far away from the geometrical domain as inside. This single misclassification will lead to singular system matrices and hence produce an unsolvable finite element model. The flood fill-based classification scheme proposed in Publication II produces reliable results even for severely flawed STLs. This is again demonstrated using nearly 5000 models from the Thingi10K database. It should be noted that standard ray tracing meth-

ods, where one or several rays determine the classification of a specific element, are sufficient for most valid (non-flawed) geometries. In those cases, the suggested classification scheme is not necessarily superior in terms of robustness. However, the underlying group-based majority voting reduces the number of total ray tracing tests required, significantly decreasing the computational cost. In summary, the developed algorithm provides robust results for flawed geometries and is more efficient than the traditional methods.

Publication 1 offers the additional feature of using so-called Generalized Gaussian Quadrature (GGQ) rules for non-cut domains. These rules are manufactured to exploit the continuity properties of B-Splines or NURBS. Thus, if such higher-continuous bases are employed, the GGQ rules can significantly reduce the number of integration points within interior/full knot spans. Compared to traditional element-wise Gaussian quadrature schemes, savings in the order of 50-90% are achieved in Section 7 in Publication 1, depending on whether optimal or reduced quadrature constructions are employed. However, by definition, the GGQ rules only affect interior/full knot spans. Their practical influence, therefore, depends on the ratio between the number of cut and full elements in the background mesh, which is determined by the shape of the geometrical domain. In the above example, the total reduction in the number of integration points is approximately 20-40%, with 7847 full and 10243 cut knot spans. Note that other examples can have comparatively far more full elements, leading to a greater improvement. Generally, GGQ rules are not fundamentally necessary for the workflow in Fig. 2.1. Nevertheless, they can decrease simulation times in specific applications, as will be discussed in the next section. For a more detailed discussion on GGQ rules, including optimal and reduced quadrature constructions, the reader is referred to Section 3.1 in Publication 1.

Regarding the complete process from STL to analysis-ready finite element model, the total computing time is primarily determined by the cost of assembling and solving the moment fitting equations. All geometrical operations, including element classification and mesh intersections, are significantly less expensive in practical applications. This is in opposition to workflows that rely on third-party geometry libraries, which are not optimized for the needs of embedded boundary methods. In our preliminary implementation based on CGAL [41], both the element classification and mesh intersections were orders of magnitude more expensive than moment fitting, restricting the application to problems of moderate size. Therefore, the presented geometry kernel is a major step towards the efficient generation of unfitted finite element models. With these developments, model creation is now considerably faster than solving a simple linear static finite element problem.

3.2.2 Performance of the unfitted finite element model

The proposed framework enables structural finite element analysis of arbitrarily complex solids, including flawed geometries. An important aspect of the model formulation is that the underlying integration points only need to be constructed once, regardless of the application. For example, in problems where the linear system matrices are solved multiple times, the calculated quadrature rules can be reused in each iteration. While the computational cost for model gen-

eration is already lower than for solving a linear finite element problem, as discussed above, it tends to be negligible in nonlinear or dynamic applications. The presented moment fitting scheme guarantees positive integration weights and locations restricted to the material domain. These are beneficial properties for finite element problems that include geometrical or material nonlinearities. In fact, negative weights can lead to a divergence of the Newton-Raphson method [34]. Moreover, material variables, such as plasticity or damage values, are difficult to interpret when associated with a point outside the physical domain or a negative weight. In conclusion, our quadrature methods bring all fundamental properties to tackle the aforementioned problems.

Furthermore, a particular focus of this work is on minimizing the number of integration points. In the scope of implicit finite element analysis, the proposed quadrature rules can drastically accelerate the assembly of the system matrices by reducing the number of function evaluations. The potential speed-up, therefore, increases with the number of required matrix formations. With these developments, Publication I aims to pave the way for efficient transient simulations using embedded boundary methods. Compared to implicit procedures, the potential for accelerating explicit time integration schemes is even greater. It turns out that in explicit dynamics, where the computational cost is directly proportional to the number of integration points, conventional quadrature schemes such as octrees strongly limit the scope of application. By contrast, quadrature rules with $n \leq (p+1)^3$ points per cut element are a significant step toward competitive explicit finite element analysis. Moreover, the proposed GGQ rules can reduce the average number of integration points per element to values significantly below $n = (p+1)^3$, further decreasing the overall simulation time.

However, a huge number of integration points is not the only issue that often prevents the use of explicit time integration schemes in embedded methods. Another common problem is that small cut elements cause critical time steps with infeasible values. Publication I demonstrates that B-Splines can help mitigate this phenomenon, provided that two essential conditions are met. The mass matrix must be diagonal/lumped, and the B-Splines employed have to be at least C^1 continuous. Note that lumped mass matrices are traditionally used by default in explicit dynamics to decouple the linear system of equations. The necessary continuity is also easy to achieve by employing B-Splines with a quadratic or higher polynomial degree. Although satisfying both requirements is, therefore, trivial, the implication of higher-order bases in combination with mass lumping is worthy of discussion. As a matter of fact, mass lumping restricts the accuracy in the frequency domain to second-order regardless of the polynomial degree [50]. Thus, additional h-refinement is inevitable to capture the dynamic behavior. Publication I discusses the described phenomenon using several examples. An additional important aspect in this context is that embedded boundary methods allow uniform meshes with equal element sizes. In contrast to boundary-fitted discretizations, the elements do not have to be refined in order to capture the geometry. Since the critical time step is generally determined by the smallest element in the computational mesh, embedded methods have a clear advantage. Thus, even if the background mesh requires the same global (average) refinement level as linear boundary-fitted tetrahedral or hexahedral elements, the critical time step will still be much larger in the embedded approach. Moreover, Publica-

tion I adopts a predictor multi-corrector scheme [51, 52]. This change preserves the computational architecture of explicit algorithms but can drastically improve the accuracy of the lumped mass matrix for higher-order bases. The main idea is to keep the lumped mass matrix on the *left-hand-side* for the decoupling of the system matrices while additionally introducing the consistent mass matrix for the calculation of the residual vector. In [52], this method is shown to behave like an implicit Newmark scheme using the consistent mass matrix if a sufficient number of corrector iterations is conducted. The associated improvement in accuracy naturally comes at a certain cost, which increases proportionally with the number of iterations. However, our examples in Publication I support the observation in [52] that 2-3 corrector passes are sufficient for most practical applications. As a result, a moderate computational overhead allows much better utilization of higher-order basis functions and, consequently, an increase of the global element size in embedded methods. Our preliminary studies indicate that the benefit of an associated higher critical time step outweighs the additional cost of the predictor multi-corrector scheme. The example in Section 7 in Publication I shows that a uniform background mesh with quadratic B-Splines can potentially increase the time step by a factor up to ≈ 45 compared to standard linear tetrahedral elements. Note that both models achieve the same relative error in strain energy in a linear static analysis. Since the polynomial degree of the B-Splines is still moderate ($p = 2$), the loss in accuracy due to mass lumping is not expected to yield a severe deterioration of the dynamic behavior in an explicit simulation. Moreover, even if a predictor multi-corrector scheme with 3 corrector passes is employed to increase the accuracy of the lumped mass matrix in the embedded model, a significant speed-up in the overall simulation time remains. Note that this estimation is based on a fully automatically generated boundary-fitted finite element mesh, in which no active attempt is made to avoid very small elements. A comparison to state-of-the-art models used in industry is yet to be conducted and may yield different results.

An additional objective of this work is the integration of shape optimization strategies into the workflow in Fig. 2.1. The concept of embedded boundary methods enables an Eulerian description of the respective shape updates, which completely eliminates mesh distortion problems that pose a major challenge for boundary-fitted discretizations. The key component in this context is the development of a novel sensitivity analysis presented in Publication III. The proposed approach computes shape gradients with respect to the immersed boundary, which are entirely independent of the background discretization and the surface mesh, i.e., the STL. Generally, once the sensitivities are available at the vertices of the STL, the same algorithms can be used as for the optimization of shells. Thus, the sensitivity analysis acts as a direct interface between the FE response computed on the background mesh and the optimizer, which operates on the immersed boundary. Publication III applies an implicit filter to control the feature size and to enforce non-design surfaces. Nevertheless, explicit filter variants are also valid options. During the optimization process, the repetitively updated geometric boundary can move freely through a fixed background mesh. In order to account for the geometric changes, new quadrature rules are constructed in each iteration. Therefore, the robust model generation discussed in Section 3.2.1 is an absolutely essential feature for a successful termination of the optimization.

Furthermore, Publication III proposes an approach to enforce geometrical constraints such as minimum wall thicknesses and design space limitations. Note that the former condition also automatically avoids infeasible, self-intersecting geometries. In summary, combining the developments discussed leads to an optimization process that can tackle arbitrarily complex geometries.

All problems in this work are solved with B-Spline bases spanning the background mesh, taking advantage of their higher-order and higher-continuity properties. Note that due to the simplicity of the background mesh, NURBS have no distinct advantage over B-Splines. We generally observe that the unfitted finite element models perform remarkably well for solid geometries that contain small features or thin-walled sections. Regarding the example in Section 7.3 in Publication III, quadratic splines with a uniform knot span size of 1.5 mm achieve practically converged solutions in strain energy for a part with a minimal wall thickness of 1 mm. In contrast, a boundary-fitted discretization with tetrahedral or hexahedral elements would require several element rows along the cross-section. This already yields considerable advantages for linear static analyses, as shown in Publication III. However, if we recapitulate the above discussion on the critical explicit time step, drastic improvements seem possible with the embedded approach.

Generally, it should be noted that the advantages of higher-order and higher-continuous basis functions come at the price of additional computational costs for certain operations. During model generation, higher orders primarily increase the size of the moment fitting equations. Remember that the point elimination scheme uses an under-determined system of equations. Thus, the initial number of points is significantly larger than $(p + 1)^3$ and scales cubically with the polynomial degree. Moreover, elevated orders and, especially, higher continuities lead to a more expensive assembly of the finite element system matrices. In addition, the associated greater bandwidth increases the burden on the linear solver. While these are general challenges in, e.g., the scope of isogeometric shell analysis, they become even more pronounced for volumetric discretizations. The studies performed in this work suggest the use of quadratic or, at most, cubic B-Splines. They drastically increase the discretization quality compared to meshes utilizing linear elements with moderate extra computational effort.

Another general phenomenon relates to the difficulty of embedded boundary methods in representing discontinuities. The main problem is that the basis functions can span over geometric gaps, which leads to an unphysical coupling of the two opposite sides of, e.g., a thin hole. A formal definition of this issue, referred to as *cross-talk* [53], is given in [54, 55]. Generally, the region influenced by *cross-talk* depends on the element size and the continuity of the basis functions. This problem is, therefore, exacerbated when using B-splines instead of C^0 continuous bases and provides a further argument to keep the spline's polynomial degree moderate. For potential solutions to mitigate *cross-talk*, the reader is kindly referred to the outlook in Section 4.2.

Conclusion and outlook

The following sections contain a concluding summary and an outlook on the framework's possible further developments.

4.1 Conclusion

This cumulative dissertation presents a robust framework for CAD-integrated structural analyses using the concept of embedded boundary methods. At its core is an open-source library – Quadrature for Embedded Solids, a.k.a. [QuESo](#) – that provides analysis-ready unfitted finite element models for arbitrarily complex solid geometries in STereoLithographie (STL) format. The publicly accessible source code is intended to encourage colleagues to integrate embedded boundary methods into their workflows. QuESo's main idea is to act as a seamless interface between any CAD environment and FE program. For this purpose, a particular focus is on generic data formats that enable the realization of the overall workflow with minimal effort. An important component is the STL model, which can be retrieved from any CAD program in near real-time. QuESo converts the respective STL file into a list of integration points containing all the necessary information to perform an FE analysis on a specified unfitted background discretization, i.e., regular grid. As a result, the communication between QuESo and the FE solver is restricted to a set of active elements with customized quadrature rules. Necessary and optional requirements for the FE framework are discussed in detail.

The constructed quadrature rules fulfill all necessary requirements for application to static linear and nonlinear finite element analyses as well as transient simulations, including implicit and explicit time integration schemes. Their crucial characteristics are listed below.

- Cut elements contain at most the same number of integration points as standard Gaussian quadrature rules, such that $n \leq (p+1)^3$, where p is the polynomial degree of the basis functions spanning the background mesh.

- If B-Splines or NURBS are employed, generalized Gaussian quadrature schemes can significantly reduce the number of integration points within interior knot spans. Savings up to $\approx 90\%$ are possible compared to classical element-wise Gaussian quadrature constructions.
- All integration points are located in the physical domain.
- The integration weights are strictly positive.

All examples studied in this thesis exploit the higher-order and higher-continuity properties of B-Splines bases. The respective finite element models maintain optimal convergence in the energy norm while being significantly more efficient than conventional methods that rely, for instance, on octree-based quadrature constructions. Particular benefits arise in explicit dynamics, where the number of integration points predominantly drives the computational cost. Moreover, B-Splines are shown to prevent infeasible critical explicit time steps that usually occur in embedded methods due to small cut elements.

Besides efficient quadrature rules, the second main contribution of this work is a geometry kernel tailored to the needs of embedded boundary methods. This includes a mesh intersection algorithm, which drastically facilitates the assembly of the moment fitting equations, and a flood fill-based element classification scheme. The underlying developments not only allow extremely fast runtimes but also provide a remarkably robust workflow that enables the processing of arbitrarily complex STLs, including flawed models. All algorithms are rigorously tested on nearly 5000 STLs from the Thingi10K database. Overall, the developments of this work fulfill the core properties sought by the isogeometric community, which are listed in (A.1)-(A.2). Note that the CAD native boundary representation inherently prevents the direct execution of finite element analyses on solid CAD models. However, the proposed framework drastically alleviates the often tedious model conversion from CAD to FEA through a fully automatized and robust process. The complete model creation is computationally less expensive than the actual solution of a simple linear static finite element analysis. Moreover, the presented developments eliminate all the shortcomings listed in (S.1)-(S.6), which at least partially limit existing embedded boundary methods.

Finally, a novel sensitivity analysis paves the way for a straightforward application of shape optimization strategies to embedded solids. In comparison to boundary-fitted meshes, embedded methods offer an Eulerian description of the shape updates, which eliminates mesh distortion problems from the outset. The proposed methodology is shown to attain the initial discretization quality throughout the entire optimization process. An effective approach to enforcing geometrical constraints, such as minimum wall thicknesses and design space limitations, complements a robust workflow that can tackle examples of industrial relevance.

4.2 Outlook

As discussed in Section 3.2.2, the accurate representation of geometric discontinuities remains a challenging task in embedded methods, especially for higher-continuous bases. The core problem is an unphysical coupling due to

shape functions that extend across geometric gaps, e.g., holes. This phenomenon is also referred to as *cross-talk* [53]. As a remedy, [56, 57] propose a local refinement scheme using hierarchical spline formulations. The idea is to refine the elements along small gaps or holes below the critical limit, where the respective degrees of freedom are fully decoupled. In addition, locally refined splines offer two further advantages. They can improve the accuracy in the vicinity of boundaries that are subjected to weak constraints and enable a more efficient approximation of local deformations [56, 57]. However, in order to prevent *cross-talk* in practical applications, potentially problematic regions must first be detected. A recent study provides a comprehensive classification of different types of *cross-talk* and proposes a systematic approach for their detection [55]. Moreover, a promising technique to mitigate *cross-talk* based on control point duplication is presented. These developments enable the decoupling of the respective degrees of freedom without the need for local mesh refinement. Therefore, this concept may also cover very small features that are below a minimum element size. Particular advantages are observed in the scope of explicit crash simulations, where relatively coarse elements are inevitable in order to achieve the necessary efficiency [55].

Since locally refined splines come with additional benefits, as discussed above, they may be prioritized in future work and subsequently combined with control point duplication. In general, the results of this work suggest that explicit dynamic simulations, such as crash analysis, can potentially greatly benefit from the proposed developments. For practical application, however, further investigations on plasticity modeling are required. While the studies in [34] yield promising results for several simple elastoplastic examples, the behavior in a component-wise or a full car crash simulation is still unknown. Future studies also need to address the extension to damage and failure mechanisms. In addition, contact modeling is a fundamental part of crash simulation that has not yet been extensively studied in the context of embedded boundary methods and, hence, requires further research. This includes the detection of colliding parts, the estimation of contact stiffnesses, and the introduction of contact forces.

Publication I

Comput. Methods Appl. Mech. Engrg. 400 (2022) 115584

Efficient CAD-integrated isogeometric analysis of trimmed solids

Manuel Meßmer, Tobias Teschemacher, Lukas F. Leidingner, Roland Wüchner, Kai-Uwe Bletzinger

Highlights

- CAD-integrated analysis workflow for solid B-Rep models based on trimmed B-Splines.
- Highly efficient numerical integration schemes for trimmed and non-trimmed domains.
- Modified point elimination algorithm based on non-negative moment fitting quadrature.
- Extension of generalized Gaussian quadrature rules to non-tensor product domains.
- Feasible critical explicit time steps despite arbitrarily trimmed knot spans.

Doi: <http://dx.doi.org/10.1016/j.cma.2022.115584>

The included publication is an open access article published by Elsevier B.V. under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).



Efficient CAD-integrated isogeometric analysis of trimmed solids

Manuel Meßmer^{a,*}, Tobias Teschemacher^a, Lukas F. Leidinger^b, Roland Wüchner^c,
Kai-Uwe Bletzinger^a

^a Chair of Structural Analysis, Technical University of Munich, Arcisstr. 21, 80333 München, Germany

^b DYNAmore GmbH, Industriestr. 2, 70565 Stuttgart, Germany

^c Institute of Structural Analysis, Technische Universität Braunschweig, Beethovenstr. 51, 38106 Braunschweig, Germany

Received 19 May 2022; received in revised form 25 July 2022; accepted 19 August 2022

Available online xxxx

Abstract

This publication presents a robust and efficient approach for fully CAD-integrated analyses of solids, which aims to reduce the current modeling effort for static and transient problems, including implicit and explicit dynamic simulations. Generating high-quality finite element meshes of solid structures is still a time- and labor-intensive process. Since embedded methods do not require sophisticated boundary-fitted meshes, they have gained popularity in recent years. However, most approaches tend to be computationally expensive due to numerous integration points, especially within trimmed elements. Moreover, their practical applicability in explicit dynamics is often limited because the classically used C^0 continuous discretization field combined with trimming leads to infeasible time steps. In the following, we present methodologies addressing both of these shortcomings.

The basic idea is to embed a three-dimensional object into a uniform C^{p-1} continuous B-Spline cuboid, where the solid boundary representation provided by CAD is used as trimming surfaces to distinguish between material and void domain. Our primary focus is on constructing highly efficient quadrature rules for both trimmed and full knot spans, which accelerates required matrix formations and, in particular, drastically reduces the simulation times of explicit transient analyses. To fully exploit the potential of the B-Spline bases employed, first- and second-order reduced integration schemes are investigated in addition to optimal quadrature constructions. Despite the appearance of arbitrarily shaped domains, trimmed knot spans are evaluated at most with the same number of integration points as required for full Gaussian quadrature while maintaining optimal convergence in the energy norm. For full knot spans, savings in the number of quadrature points beyond 90% with respect to full Gaussian quadrature are achieved without observing any degradation in accuracy.

The proposed methodologies are critically assessed based on scientific benchmarks of increasing complexity and a detailed industrial example, completing the design-through-analysis workflow by performing postprocessing operations directly on the deformed solid CAD model.

© 2022 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Keywords: Isogeometric B-Rep Analysis (IBRA); Trimmed trivariate B-Splines; Moment fitting equation; Point elimination algorithm; Generalized Gaussian quadrature; Implicit and explicit finite element analysis

* Corresponding author.

E-mail address: manuel.messmer@tum.de (M. Meßmer).

1. Introduction

The continuous innovations in Computer-Aided Engineering (CAE) have significantly increased the scope and complexity of models in modern development processes. Inevitably, this also increased the demand for facilitated modeling to avoid labor-intensive interfaces between design and analysis. Accordingly, simulation integrated with Computer-Aided Design (CAD) has become a vital field of research. Conventional procedures discretize the CAD geometry with lower-order entities to create a model suitable for analysis. This process is called meshing and is unavoidable in the probably most prominent approach in CAE — the Finite Element Method (FEM). However, meshing can be a time-consuming and labor-intensive process that often requires manual intervention, is error-prone, and results in an approximate model.

To overcome this problem, Hughes et al. introduced the Isogeometric Analysis (IGA), which allows the Non-Uniform Rational B-Spline (NURBS) shape functions provided by CAD to be used directly within an analysis framework [1,2]. The associated transformation from traditional FEM to IGA has the potential to increase the solution quality by avoiding any remodeling and by preserving the exact geometry for simulation. IGA inherently provides the ability to analyze single, untrimmed NURBS curves, surfaces, and solids. However, most CAD programs rely on the geometric boundary representation (B-Rep) [3], which shapes objects solely by their geometric delineations. This enables a fast model generation but adds further complexity to numerical simulations due to trimming and the occurrence of multiple patches.

The automatic construction of untrimmed spline-based models suitable for analysis seems an effective solution to circumvent the aforementioned difficulties. Isogeometric analysis on T-Splines promises to be an essential step toward this goal [4]. T-Splines are a generalization of NURBS, allow mesh refinement/coarsening, and enable the representation of holes without trimming [5]. In [6], an algorithm for constructing surface and volume T-Splines from unstructured quadrilateral and hexahedral meshes is proposed. This concept is further developed in [7] to construct volumetric T-splines from standard boundary triangulations efficiently. However, the proposed algorithm is limited to genus-zero topologies and does not guarantee positive Jacobians for all Bézier elements. Inspired by the concept of Constructive Solid Geometry (CSG), the same task is solved in [8] based on Boolean operations. Optimal convergence rates are achieved in [9] using the construction of blended B-Splines over unstructured quadrilateral and hexahedral meshes. Similar to the previously mentioned strategies, these approaches do not apply to arbitrarily complex geometries and are limited to C^0 continuous splines at irregular subdomains containing extraordinary vertices. Other algorithms are devoted to the even more challenging process of designing tensor product splines [10,11], which fits nicely into the isogeometric paradigm, but inevitably leads to multiple spline patches for complex geometries. In [12,13], the concept of the scaled boundary finite element method is applied to parameterize the volumetric physical domain of the solid CAD model by scaling the boundary based on a predefined radial scaling center. This method has been shown to produce accurate results, but a suitable physical domain decomposition algorithm has yet to be developed for its application to complex real-world problems. Despite existing promising approaches, the automatized generation of boundary-fitted analysis-suitable surface splines and, in particular, volumetric splines from the NURBS-based boundary representation is still an open research question.

Another important role in this context is played by the family of immersed and embedded boundary methods, which are characterized by the fact that no sophisticated boundary-fitted meshes are required. The eXtended Finite Element Method (XFEM) [14,15] was initially developed to simulate discontinuities such as cracks by enriching the corresponding shape functions at the points of interest. In the scope of the Cut Finite Element Method (CutFEM) [16], the same concept is applied to embedded boundaries or interfaces. Instead of modifying the basis functions of individual elements, the Finite Cell Method (FCM) [17–19] introduces an indicator function into the variational form to represent the material discontinuity at the geometric boundary. The classical FCM combines the fictitious domain technique [20,21] with the high-order finite element approach. Its application to dynamic problems in the time domain is presented in [22]. The authors in [23] formulate a conservative approach to accurately evaluate boundary fluxes in embedded domain methods. In addition to the widely spread and studied p-version of the FCM, its concept has also been extended to B-Spline bases [24]. The FCM solves the problem of capturing complex geometries not explicitly through a boundary-fitted mesh but by an accurate integration of discontinuous functions. Similar challenges arise in the Isogeometric B-Rep Analysis (IBRA) [25,26], which can be seen as an extension to IGA, including trimmed patches, weak enforcement of constraints, and coupling of multi-patches based on NURBS. Since IBRA strictly avoids any remodeling, it features complete data consistency between design and analysis. The development of IBRA has enabled a wide range of models for structural analyses but has only been able

to handle surface- and curve-based formulations. This limitation is mainly because most CAD programs rely on B-Rep modeling and describe three-dimensional objects solely by their delimiting surfaces, i.e., they provide neither a physical nor a geometrical description of the interior.

If the classical IBRA concept is directly extended to three dimensions by representing the physical domain through a trimmed B-Spline discretization, the boundaries to embedded methods such as FCM become blurred. Therefore, the CAD-integrated analysis workflow presented in this work adopts features from both the IGA and FCM communities. For a continuous mathematical description of the entire physical domain, the B-Rep provided by CAD is defined as the trimming surfaces of a uniform trivariate B-Spline cuboid. Following the IBRA paradigm, the trimming surfaces are incorporated into the parametric space of the solid, which guarantees a consistent boundary, e.g., for the imposition of boundary conditions, throughout the entire simulation. In addition, our approach is distinguished from others by highly efficient quadrature rules for both trimmed and non-trimmed domains, based on integration points with strictly positive weights and locations limited to the material domain. The associated drastic reduction of quadrature points accelerates required matrix formations and paves the way for efficient explicit dynamic simulations. Note that due to the computational architecture of explicit algorithms, their predominant cost is determined by the number of quadrature points. Despite the presence of arbitrarily trimmed knot spans, practically feasible explicit time steps are guaranteed by using C^{p-1} continuous basis functions. Moreover, a predictor multi-corrector scheme is adopted to improve the accuracy of the lumped mass matrix for higher-order bases. An overview of this work shall be given in the following.

- Section 2 discusses preliminaries and describes the basic steps from a standard B-Rep model to a model suitable for analysis, leading to a consistent extension of the IBRA concept to three dimensions.
- Section 3 presents the core of this publication. Based on the original idea proposed in [27], a Generalized Gaussian Quadrature (GGQ) scheme is developed to construct nearly optimal and highly efficient reduced integration rules for all full knot spans. A novel algorithm is presented to decompose arbitrary domains into suitable tensor products to enable the application of GGQ to trimmed trivariate B-Spline patches. For an efficient numerical integration of trimmed knot spans, the point elimination algorithm from [28] is combined with the recent developments presented in [29] to inherently achieve positive integration weights and points that are all within the material domain.
- Section 4 discusses solutions to prevent numerical stability issues arising from small trimmed knot spans.
- Section 5 gives an overview of the presented workflow and covers pre- and postprocessing within CAD.
- Section 6 demonstrates the method's effectiveness based on static and transient benchmark problems.
- Section 7 presents the simulation results of a detailed solid CAD model with an industrial level of complexity.
- Section 8 concludes and discusses open research questions.

2. Isogeometric analysis of trimmed solids: Preliminaries and concept

IGA aims at the interchangeable use of design and analysis models. In order to improve industrial workflows, various shell and membrane element formulations have been derived from the isogeometric concept and successfully applied to NURBS-based CAD models. The following section discusses the geometric representation of solid CAD models and the resulting challenges for IGA. Subsequently, the solution approach pursued in this work will be presented.

2.1. Solid CAD models

For the mathematical description of solid geometries, a distinction is made between implicit and explicit representations. Implicit models express the geometry by the level set of a function $f(x, y, z)$ such that $f < 0$ if point $P(x, y, z)$ is inside the boundary, and $f > 0$ if $P(x, y, z)$ is outside the boundary, whereas a zero level set $f = 0$ indicates that $P(x, y, z)$ lies on the boundary. The CSG enables the representation of complex objects by combining implicitly defined primitives through Boolean operations. For direct analysis based on CSG models, the interested reader is referred to [30].

On the other hand, explicit models are represented through their bounding surfaces, which form the B-Rep. Unlike implicit techniques, explicit B-Rep models allow direct and efficient visualization, which made them the predominant geometry description in modern CAD systems. Generally, a B-Rep encloses a volume if and only

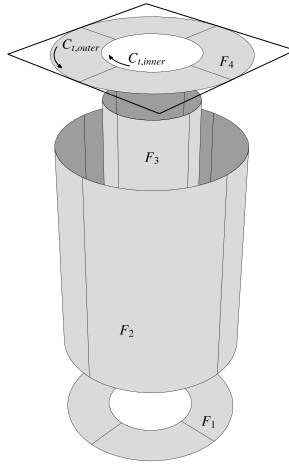


Fig. 1. CAD-based B-Rep model of a cylinder with a hole.

if its surface is an orientable two-manifold without a boundary [31]. Fig. 1 illustrates the B-Rep concept using a simple cylindrical object defined solely by the faces F_{1-4} . The faces F_1 and F_4 are delimited by an inner $C_{t,inner}$ and outer trimming curve $C_{t,outer}$. Both surfaces and curves are mathematically described by B-Splines or NURBS, enabling a direct analysis on one- and two-dimensional topologies [1,2,26]. However, due to the lack of a volumetric function space, the isogeometric paradigm is not readily applicable to solid B-Rep models but requires additional parameterization of the interior. Throughout this work, the computational domain $\Omega \subset \mathbb{R}^3$ is defined by its closed boundary $\Gamma \subset \mathbb{R}^3$ with

$$\Gamma = \bigcup_a F_a. \tag{1}$$

Note that a B-Rep may also contain internal boundaries, which are not considered in Eq. (1) for brevity.

2.2. B-Spline shape functions for solids

To fill the inner cavity of the B-Rep model, as exemplified in Fig. 1, the geometry is embedded into a trivariate B-Spline discretization. The basic principles of the underlying basis functions and the construction of B-Spline solids are discussed below.

B-Spline basis functions $N_{i,p}$ are defined by their polynomial degree p and a sorted set of coordinates in parametric space, denoted as knot vector $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$, with $\xi_i \in \mathbb{R}$ being the i th knot and n denoting the number of basis functions. They can be constructed using the Cox-de Boor recursion formula [32,33]

$$\begin{aligned} \text{if } p = 0: \quad N_{i,0}(\xi) &= \begin{cases} 1, & \xi_i \leq \xi < \xi_{i+1}; \\ 0, & \text{otherwise} \end{cases}; \\ \text{else:} \quad N_{i,p}(\xi) &= \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \end{aligned} \tag{2}$$

The computed functions are $C^{p-\bar{k}}$ continuous, where \bar{k} is the knot multiplicity. Throughout this work, we employ $\bar{k} = 1$ for all inner knots except otherwise specified and use $\bar{k} = p + 1$ at the ends of the knot interval, resulting in an open knot vector. A trivariate parameterization for solids is obtained from a tensor product of the shape functions in each spatial direction. The mapping from a point $\xi = (\xi, \eta, \zeta)$ in parametric space defined by the knot vectors

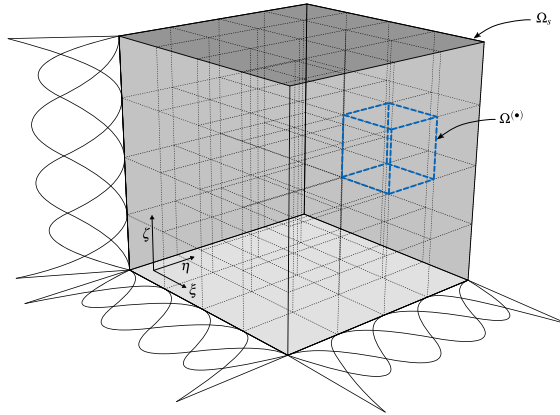


Fig. 2. Embedded solid domain Ω_s spanned by quadratic B-Spline shape functions with $4 \times 4 \times 4$ knot spans and uniform open knot vectors. One individual knot span domain is indicated by $\Omega^{(\bullet)}$.

$\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p_\xi+1}\}$, $\varkappa = \{\eta_1, \eta_2, \dots, \eta_{m+p_\eta+1}\}$, and $\vartheta = \{\zeta_1, \zeta_2, \dots, \zeta_{l+p_\zeta+1}\}$ to its corresponding physical point is computed as

$$S(\xi) = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^l N_{i,p_\xi}(\xi) N_{j,p_\eta}(\eta) N_{k,p_\zeta}(\zeta) \mathbf{P}_{i,j,k}, \tag{3}$$

with the control points $\mathbf{P}_{i,j,k} \in \mathbb{R}^3$ and n, m , and l being the respective number of basis functions. For the sake of conciseness, we may write Eq. (3) as $S = \mathbf{N}\mathbf{P}_{i,j,k}$. In the following, S is defined over the embedded solid domain Ω_s , where $\Omega^{(\bullet)}$ is the subdomain spanned by one knot interval $[\xi_i, \xi_{i+1}] \times [\eta_j, \eta_{j+1}] \times [\zeta_k, \zeta_{k+1}]$, such that

$$\Omega_s = \bigcup_a \Omega_a^{(\bullet)}. \tag{4}$$

Note that the polynomial degree may be chosen differently for $N_{i,p_\xi}(\xi)$, $N_{j,p_\eta}(\eta)$, and $N_{k,p_\zeta}(\zeta)$, but we fix $p = p_\xi = p_\eta = p_\zeta$ for clarity. Fig. 2 illustrates a B-Spline domain with $4 \times 4 \times 4$ knot spans and quadratic basis functions. This publication focuses on B-Splines with a polynomial degree of $p = 2$ to $p = 4$. The linear case ($p = 1$) is not considered to ensure C^1 continuity or higher over the entire domain.

2.3. Trimmed solid

To avoid a computationally expensive meshing process, the initial non-trimmed domain of the B-Spline solid is restricted to a cuboid shape defined by uniform knot vectors. With this simplification, any B-Spline discretization can be efficiently constructed from algorithms for order elevation and knot refinement [34]. The required inputs are:

- Physical dimension and orientation of the B-Spline solid.
- Polynomial degree p .
- Knot span size.

Remark. NURBS, as a generalization of B-Splines, would also be a suitable parameterization. However, as the embedded domain Ω_s is created without considering the actual computational domain Ω , NURBS provide no distinct benefit over B-Splines but introduce additional complexity.

Due to the above established restrictions on the B-Spline mesh, the boundary Γ of the computational domain $\Omega \subset \Omega_s$ will not coincide with the boundaries $\Gamma^{(\bullet)}$ of the individual knot span subdomains $\Omega^{(\bullet)}$. Therefore, a distinction is

made between the interior (untrimmed) knot span domain Ω^i , the exterior (empty) knot span domain Ω^e , and the trimmed knot span domain divided into its inner Ω^i and outer part $\Omega^{i,e}$, such that the physical domain of the solid is defined as

$$\Omega = \left(\bigcup_a \Omega_a^i \right) \cup \left(\bigcup_b \Omega_b^{i,e} \right). \tag{5}$$

2.4. Variational formulation

For the following discussion, the boundary Γ of the computational domain Ω is partitioned into a Neumann boundary Γ_N and a Dirichlet boundary Γ_D , where $\overline{\Gamma_D \cup \Gamma_N} = \Gamma$ and $\Gamma_D \cap \Gamma_N = \emptyset$. Moreover, $\mathbf{x} = (x, y, z)$ denotes an arbitrary point in physical space at time t .

2.4.1. Strong form

Given the symmetric Cauchy stress tensor $\boldsymbol{\sigma}$, the body force \mathbf{b} , and the material density ρ , the initial boundary value problem reads

$$\begin{aligned} \operatorname{div} \boldsymbol{\sigma} + \mathbf{b} &= \rho \ddot{\mathbf{u}} && \text{in } \Omega, \\ \mathbf{u} &= \bar{\mathbf{u}} && \text{on } \Gamma_D, \\ \mathbf{t} = \boldsymbol{\sigma} \mathbf{n} &= \bar{\mathbf{t}} && \text{on } \Gamma_N, \end{aligned} \tag{6}$$

where $\mathbf{n} \in \mathbb{R}^3$ is the outward pointing unit normal on Γ . The prescribed displacement $\bar{\mathbf{u}} \in \mathbb{R}^3$ is enforced on the Dirichlet boundary Γ_D , while the traction $\bar{\mathbf{t}} \in \mathbb{R}^3$ acts on the Neumann boundary Γ_N . Due to the time dependence of Eq. (6), the initial conditions $\mathbf{u}(\mathbf{x}, t)|_{t=0} = \mathbf{u}_0(\mathbf{x})$ and $\dot{\mathbf{u}}(\mathbf{x}, t)|_{t=0} = \dot{\mathbf{u}}_0(\mathbf{x})$ are additionally introduced [35]. In the scope of this work, the initial values $\mathbf{u}_0(\mathbf{x})$ and $\dot{\mathbf{u}}_0(\mathbf{x})$ are assumed to be zero and only homogeneous Dirichlet conditions are considered.

2.4.2. Weak form

We define the trial space $\mathcal{U}(\Omega) = \{\mathbf{u}(\mathbf{x}, t) \mid \mathbf{u}(\mathbf{x}, t) \in H^1(\Omega), \mathbf{u}|_{\Gamma_D} = \bar{\mathbf{u}}\}$ and the weighting space $\mathcal{V}(\Omega) = \{\mathbf{v}(\mathbf{x}) \mid \mathbf{v}(\mathbf{x}) \in H^1(\Omega), \mathbf{v}|_{\Gamma_D} = 0\}$, which satisfies the homogeneous Dirichlet boundary conditions on Γ_D . Multiplying Eq. (6) by an arbitrary test function $\mathbf{v} \in \mathcal{V}$ and integration by parts leads to the variational form of the initial boundary value problem, which reads: find $\mathbf{u} \in \mathcal{U}$ such that

$$(\rho \ddot{\mathbf{u}}, \mathbf{v}) + a(\mathbf{u}, \mathbf{v}) = L(\mathbf{v}), \quad \forall \mathbf{v} \in \mathcal{V}, \tag{7}$$

with

$$(\rho \ddot{\mathbf{u}}, \mathbf{v}) + a(\mathbf{u}, \mathbf{v}) = \int_{\Omega} (\rho \ddot{\mathbf{u}}) \cdot \mathbf{v} \, d\Omega + \int_{\Omega} \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{v}) \, d\Omega, \tag{8}$$

and

$$L(\mathbf{v}) = \int_{\Omega} \mathbf{b} \cdot \mathbf{v} \, d\Omega + \int_{\Gamma_N} \bar{\mathbf{t}} \cdot \mathbf{v} \, d\Gamma. \tag{9}$$

Thereby, H^1 denotes the first-order Sobolev space [36], and $\boldsymbol{\epsilon}$ represents the symmetric gradient of the displacement field.

2.4.3. Boundary conditions

Neumann boundary conditions on Γ_N appear naturally in the variational form, see Eq. (9). To enforce Dirichlet boundary conditions, a penalty term [37] is introduced to Eq. (7)

$$b(\mathbf{u}, \mathbf{v}) = \beta \int_{\Gamma_D} (\mathbf{u} - \bar{\mathbf{u}}) \cdot \mathbf{v} \, d\Gamma, \tag{10}$$

with the penalty factor β . It should be mentioned that Lagrange multiplier and Nitsche-type methods are also possible candidates to prescribe essential boundary conditions. A comprehensive comparison of these formulations in the context of isogeometric analysis can be found in [38].

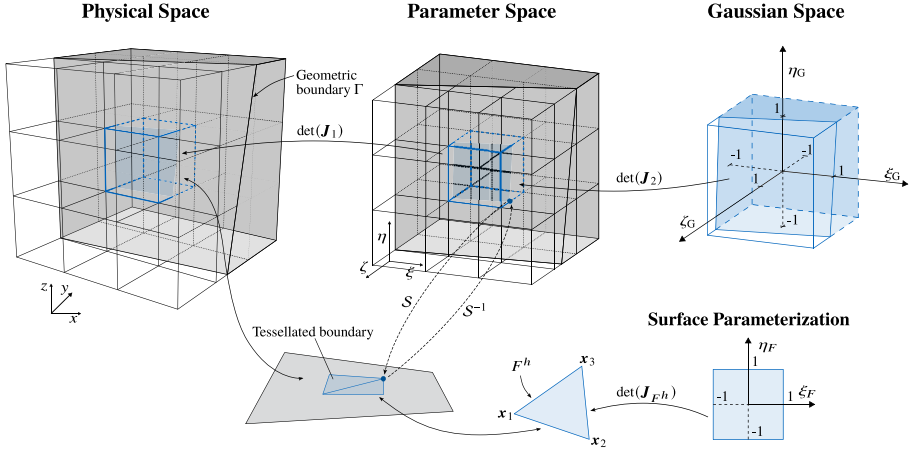


Fig. 3. Mapping between spaces. The illustration shows an excerpt of the solid B-Spline domain Ω_S that is trimmed by the geometric boundary Γ .

2.4.4. Discretization in space and time

According to the Bubnov–Galerkin approach [39], both the trial and the weighting function are discretized with the same Ansatz functions, namely the trivariate B-Spline bases given in Eq. (3), which reads

$$\mathbf{u} = \mathbf{N}\mathbf{U}, \quad \mathbf{v} = \mathbf{N}\mathbf{V}. \tag{11}$$

The spatial discretizations provided in Eq. (11) are substituted into Eq. (7) to arrive at the semi-discrete formulation

$$\mathbf{M}\ddot{\mathbf{U}} + \mathbf{K}\mathbf{U} = \mathbf{F}, \tag{12}$$

with \mathbf{M} , \mathbf{K} , and \mathbf{F} denoting the mass matrix, stiffness matrix, and global load vector. The vector \mathbf{U} represents the displacements at the control points $\mathbf{P}_{i,j,k}$ of the B-Spline solid \mathcal{S} . In Section 6.4, Eq. (12) is solved using the implicit Newmark and the explicit central difference method [39]. For the latter time integration scheme, the system of equations is classically decoupled by a diagonal mass matrix \mathbf{M}_L . However, [40] shows that \mathbf{M}_L obtained, e.g., by row summing, yields only second-order accurate natural frequencies, regardless of the polynomial degree of $N_{i,p}$. To improve the accuracy for higher-order bases, we employ an explicit predictor multi-corrector algorithm [39,41]. It also exploits the diagonal matrix \mathbf{M}_L for the simplification of Eq. (12) but additionally uses the consistent mass matrix \mathbf{M} to compute the residual vector. In Section 6.4, the performance of the predictor multi-corrector algorithm is compared to the explicit central difference scheme. For all static problems solved in Section 6, the dynamic term in Eq. (12) is neglected.

2.5. Mapping between spaces

Fig. 3 reveals that different parametric spaces can be identified as part of the presented workflow. Integration by numerical means requires an appropriate mapping among them and the physical space, which will be discussed in this section. For a detailed description of the employed numerical integration schemes, the reader is referred to Section 3.

Following the notation introduced earlier, let $\xi = (\xi, \eta, \zeta)$ denote a point in the parameter space spanned by the knot vectors Ξ , \mathbf{x} , and ϑ . The direct mapping from ξ to its conjugate coordinate $\mathbf{x} = (x, y, z)$ located in the physical domain Ω_S is defined by the B-Spline solid $\mathcal{S} : \xi \mapsto \mathbf{x}$, as stated in Eq. (3). Its inverse, which generally requires the solution of a nonlinear system of equations [34], is denoted as $\mathcal{S}^{-1} : \mathbf{x} \mapsto \xi$.

Remark. If Ξ , κ and ϑ are uniform knot vectors, and the solid is in the undeformed configuration (cuboid shape), S^{-1} decomposes into a linear function. Since all necessary conditions can be met naturally during preprocessing, exploiting this property accelerates required mapping operations drastically.

The determinant of the Jacobian matrix $\det(\mathbf{J}_1)$ (see Fig. 3) accounts for the volumetric change of the computational domain in physical space Ω and parametric space $\hat{\Omega}$, with

$$\mathbf{J}_1 = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}. \tag{13}$$

Each individual knot span domain $\hat{\Omega}^{(s)}$ is additionally mapped to a Gaussian space $\hat{\Omega}_G^{(s)}$ in order to perform numerical integration. The determinant of the corresponding Jacobian matrix reads

$$\det(\mathbf{J}_2) = \frac{\partial \xi}{\partial \xi_G} \frac{\partial \eta}{\partial \eta_G} \frac{\partial \zeta}{\partial \zeta_G}. \tag{14}$$

Since the computational domain Ω contains untrimmed Ω^i and trimmed domains Ω^t , the relevant integral can be decomposed to

$$\int_{\Omega} d\Omega = \sum_a \int_{\Omega_a^i} d\Omega_a^i + \sum_b \int_{\Omega_b^t} d\Omega_b^t, \tag{15}$$

$$= \sum_a \int_{\hat{\Omega}_a^i} \det(\mathbf{J}_1) d\hat{\Omega}_a^i + \sum_b \int_{\hat{\Omega}_b^t} \det(\mathbf{J}_1) d\hat{\Omega}_b^t, \tag{16}$$

$$= \sum_a \int_{\hat{\Omega}_{a,G}^i} \det(\mathbf{J}_1)\det(\mathbf{J}_2) d\hat{\Omega}_{a,G}^i + \sum_b \int_{\hat{\Omega}_{b,G}^t} \det(\mathbf{J}_1)\det(\mathbf{J}_2) d\hat{\Omega}_{b,G}^t. \tag{17}$$

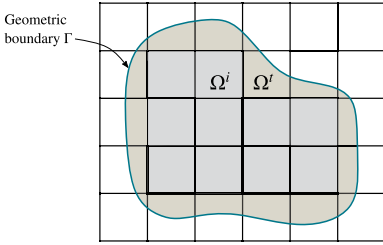
The integration over Γ may be directly performed using the corresponding surfaces F of the NURBS-based B-Rep model. However, the parameterization provided by CAD usually contains trimmed patches, as depicted in Fig. 1, which requires sophisticated integration schemes (see Section 3.2). To facilitate the embedding process, the essential integrals are performed over a triangulated boundary, as suggested in [18]. The respective integral reads

$$\int_{\Gamma} d\Gamma = \sum_a \int_{\Gamma_{F_a^h}} d\Gamma_{F_a^h} = \sum_a \int_{\hat{\Gamma}_{F_a^h}} \det(\mathbf{J}_{F_a^h}) d\hat{\Gamma}_{F_a^h}, \tag{18}$$

where $\det(\mathbf{J}_{F^h})$ represents the mapping between physical (Γ_{F^h}) and parametric space ($\hat{\Gamma}_{F^h}$) of each boundary triangle F^h . During preprocessing, each triangle is mapped and stored in the parametric space of the B-Spline solid (see Fig. 3), ensuring a consistent boundary parameterization throughout the entire simulation, even as the geometry deforms. A more detailed discussion on the evaluation of boundary integrals and the treatment of non-matching triangles at trimmed knot span boundaries is provided in Section 3.2.2.

3. Numerical integration of trimmed solids

This section is devoted to the construction of efficient quadrature rules for both trimmed Ω^t and full knot span domains Ω^i . Exterior knot span domains Ω^e , on the other hand, are empty and do not have any contribution to the weak form in Eq. (7). For brevity, we use the term knot span as a synonym for knot span domain in the following. As indicated in the previous sections, this approach relies on an a priori classification of knot spans with respect to the geometry boundary. Intersected Ω^i , interior Ω^i , and exterior Ω^e knot spans may be categorized using multiple inside/outside tests, e.g., at each knot. Additional intermediate points can be considered to improve robustness and accuracy. Ray tracing techniques allow such point membership classifications to be performed directly on the NURBS-based B-Rep model [42]. However, classical ray tracing on conventional CAD models is computationally expensive and might fail due to non-watertight geometries. The challenge of point membership classification on flawed CAD models is addressed in [43], along with a strategy for direct mechanical analysis of these models using the FCM.



- Section 3.1 presents the numerical integration of Ω^i .
- Section 3.2 presents the numerical integration of Ω^j .

Fig. 4. Structure of Section 3 based on the distinction of different integration domains.

In the following, we use a robust and efficient alternative to ray tracing on NURBS-based B-Rep models and classify knot spans based on an intermediate tessellation. Note that the same discretization is reused to evaluate the boundary integrals in Eq. (18). The necessary information is recovered from an STL (STereoLithography or Standard Tessellation Language) model, which by definition contains only triangular elements. The STL is a common file format to exchange geometric information between CAD and Computer-Aided Manufacturing (CAM) processes such as rapid prototyping and 3D printing. Due to the wide range of possible applications, the efficient generation of STL representations from NURBS-based B-Rep models is a common task for CAD programs. Appendix A briefly discusses the influence of different parameter settings for the tessellation algorithm on the accuracy of STL meshes. In the context of the classification problem, the STL can be utilized to speed up the process as it allows the use of well-established geometric algorithms for polygonal meshes [44]. An efficient and robust two-step classification scheme using the STL format is applied in this work. In the first step, intersected knot spans are identified according to [45]. Implementations of the corresponding algorithms are available in the open-source project CGAL [46,47]. In the second step, all untrimmed knot spans are categorized as interior or exterior according to the location of their center relative to the geometric boundary, similar to the point membership classification presented in [48] for the application of the FCM based on oriented point clouds.

The following discussion on the numerical integration of trimmed and non-trimmed domains is organized as depicted in Fig. 4.

3.1. Numerical integration of full knot spans

Full knot spans are typically evaluated by the tensor product of Gaussian quadrature rules, achieving exact integration. However, the number of integration points can be significantly decreased while maintaining full accuracy by leveraging the continuity property of B-Splines. The conceptual idea for optimal and reduced integration rules for tensor product splines with C^{p-1} continuity will be discussed in the following. Subsequently, this concept is extended to arbitrary arrangements of full knot spans within a trimmed patch.

3.1.1. Construction of optimal and reduced quadrature rules

In [27], Hughes et al. initiated the discussion of optimal quadrature or Generalized Gaussian Quadrature (GGQ) rules for NURBS and B-Splines to improve the efficiency of IGA in general. These rules are referred to as optimal since no other exact construction with fewer integration points exists [27,49]. The key concept is to construct integration schemes for a macro element rather than for individual knot spans. In this context, a macro element can be defined as multiple consecutive knot spans or an entire patch. For the following discussion, we introduce the function space

$$\mathcal{L}_s^q \quad \text{with} \quad \begin{matrix} q : \text{polynomial degree,} \\ s : \text{continuity,} \end{matrix} \tag{19}$$

where q and s are associated with the specified macro element. Within a one-dimensional B-Spline domain, each knot span is influenced by $n_{cp} = q + 1$ control points. In conjunction with the fact that Gaussian quadrature gives

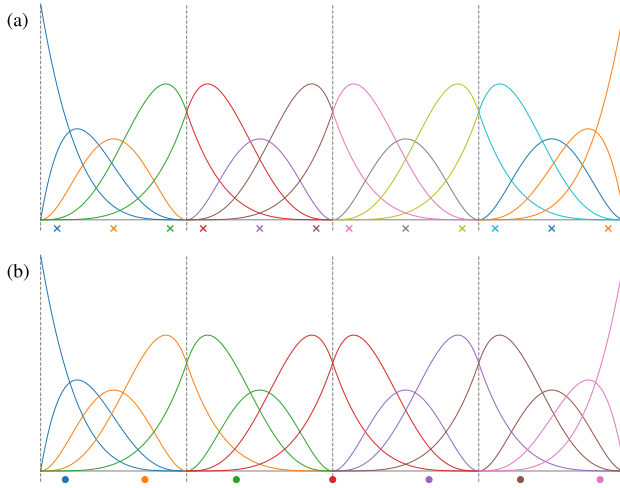


Fig. 5. Distribution of integration points: B-Spline basis functions over four knot spans with a uniform open knot vector, $q = 4$ and $s = 1$. (a) Knot span-wise Gaussian quadrature. (b) Generalized Gaussian quadrature. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

exact results for $n_q \geq (q + 1)/2$ quadrature points, we state

$$n_q \geq \frac{n_{cp}}{2}. \tag{20}$$

This suggests that, on average, each quadrature point can be assigned to two control points and, more importantly, is able to evaluate the associated basis functions. Using standard knot span-wise Gaussian integration, the condition in Eq. (20) is satisfied independently for each knot span. However, due to the smoothness of B-Splines it is beneficial to fulfill Eq. (20) over an enlarged domain. This is exemplified in Fig. 5 for B-Spline bases of order $q = 4$ and regularity $s = 1$. The colors indicate a fictitious affiliation between basis functions and integration points. They are neither meant to represent a strict connection nor an independent integration area but rather to illustrate the conceptual idea.

In IGA, as in traditional FEM, the polynomial degree and the continuity of the occurring integrands are determined by the weak form. According to Eqs. (7) and (11), the integrals for the computation of the mass and stiffness matrices take the following well-known structure

$$\int_{\Omega} N_i(\xi)N_j(\xi)\phi(\xi) d\Omega, \tag{21}$$

$$\int_{\Omega} \nabla N_i(\xi)\nabla N_j(\xi)\phi(\xi) d\Omega. \tag{22}$$

where $N_i(\xi)$, and $N_j(\xi)$ denote the tensor product B-Spline basis functions, and $\phi(\xi)$ represents the geometrical mapping between physical space and parameter space. It is a common approach to chose appropriate quadrature rules under the assumption that $\phi(\xi)$ is constant [27,50]

$$\int_{\Omega} N_i(\xi)N_j(\xi) d\Omega, \tag{23}$$

$$\int_{\Omega} \nabla N_i(\xi)\nabla N_j(\xi) d\Omega. \tag{24}$$

Consequently, the integrands in Eqs. (23) and (24) are contained in \mathfrak{L}_r^{2p} and \mathfrak{L}_{r-1}^{2p} , which aggregate to a total space of \mathfrak{L}_{r-1}^{2p} , with p and r denoting the polynomial degree and regularity of the basis functions $N_i(\xi)$, respectively. Similar to standard Gaussian quadrature, optimal rules for the integration of functions in \mathfrak{L}_{r-1}^{2p} can be obtained from the univariate case and applied to multiple dimensions using the tensor product [27]. However, these rules are unique for one particular knot vector, i.e., if the position of a single knot is slightly shifted, the rule must be adopted. Moreover, their construction requires the solution of a system of equations that depends linearly on the integration weights but strongly nonlinearly on the positions of the integration points. Since this is a challenging task, especially for large meshes, different algorithms solve the nonlinear equations locally [50,51], leading to more efficient but still suboptimal quadrature rules. For some combinations of polynomial degree and continuity, explicit recursion methods [52–54] are used to find optimal quadrature rules for B-Spline bases. Furthermore, [55,56] employ Gauss–Greville rules to avoid solving the nonlinear equations and successfully apply the concept to isogeometric shell analysis.

However, since the proposed trivariate B-Spline discretization relies on uniform knot vectors, it opens the door for a more efficient methodology. In [49], Hiemstra et al. propose to assemble optimal integration points for uniform knot vectors from precomputed quadrature rules. Generally, these constructions still depend on the number of knot spans, here denoted as n_{ks} . Nonetheless, observations show that integration points with sufficient distance from the boundary follow a periodic pattern. Therefore, optimal quadrature rules can be constructed for any number of knot spans from a few known solutions. Considering an exact integration of cubic basis functions associated with the target space \mathfrak{L}_0^3 , only the nearest 15 points are affected by each boundary. Due to symmetry constraints, an additional center rule must be considered, which varies depending on whether n_q is even or odd. Optimal quadrature rules are explicitly provided in [49] to evaluate quadratic (\mathfrak{L}_0^4), cubic (\mathfrak{L}_1^6), and quartic (\mathfrak{L}_2^8) B-Spline bases with uniform knot vectors. To further decrease the number of integration points, reduced quadrature rules may be constructed by decreasing the polynomial degree of the target space: $\mathfrak{L}_{r-1}^{2p-1}$. In fact, [49] shows that optimal convergence rates can be achieved with \mathfrak{L}_0^3 , \mathfrak{L}_1^5 , and \mathfrak{L}_2^7 for $p = 2$, $p = 3$, and $p = 4$. To consequently continue the investigations in [49], we additionally consider the second-order reduced spaces ($\mathfrak{L}_{r-1}^{2p-2}$) in the examples conducted in Sections 6.3 and 6.4. The positions and weights of the corresponding integration points are computed with a relative error of $< 10^{-15}$ and documented in Appendix B for the first 10 knot spans and $p = 2$, $p = 3$, and $p = 4$. Fig. 6 compares the number of required integration points for \mathfrak{L}_{r-1}^{2p} , $\mathfrak{L}_{r-1}^{2p-1}$, and $\mathfrak{L}_{r-1}^{2p-2}$ to element-wise full ($n_q/n_{ks} = (p+1)^3$) and reduced ($n_q/n_{ks} = (p)^3$) Gaussian quadrature. The gain in efficiency clearly depends on the mesh size, but even for small and moderate numbers of knot spans, the reduction is shown to be significant. In the limit case, optimal quadrature rules (\mathfrak{L}_{r-1}^{2p}) for cubic splines can save up to 75,5% of the necessary points. When reduced integration schemes are applied this number is increased to 87,5% for $\mathfrak{L}_{r-1}^{2p-1}$, and 94,7% for $\mathfrak{L}_{r-1}^{2p-2}$, respectively.

3.1.2. Generalized Gaussian quadrature for non-tensor product spaces

Following the concept introduced in the previous section, optimal and reduced one-dimensional quadrature rules can be found for any number of knot spans at minimal cost. In [49], these quadrature rules are derived from uniform open knot vectors. We want to emphasize that the respective quadrature constructions depend on the number of knot spans, the continuity, and the polynomial degree of the given integrand. However, they are not restricted to open knot vectors. The same quadrature rules can also be applied to non-open knot vectors or any set of consecutive intermediate knot spans as long as the knot vectors are uniform. To illustrate this phenomenon, we construct the GGQ rule for the same B-Spline target space as in Fig. 5(b), but with trimmed ends, see Fig. 7. Since the number of active knot spans is identical, we obtain the same quadrature rule as without trimmed ends. This is because the basis functions of open and closed knot vectors differ only in the polynomial coefficients, but not in the polynomial degree, the continuity in the interior of the domain, or the number of active basis functions. As the coefficients are constants, they do not affect the quadrature rule. Consequently, the precomputed integration points can be applied to any subset of consecutive full knot spans within the trimmed B-Spline domain Ω .

However, the construction of global multi-dimensional rules using the tensor product is not straightforward in the context of the present work. It is only applicable if all full knot spans form a perfect cuboid, representing only one particular corner case, as the neighbor relations of untrimmed knot spans can be arbitrary. In order to ensure an efficient integration for any knot span arrangement in multiple dimensions, a novel decomposition algorithm is presented. Thereby, adjacent knot spans are grouped into tensor product domains that have the shape of a cuboid

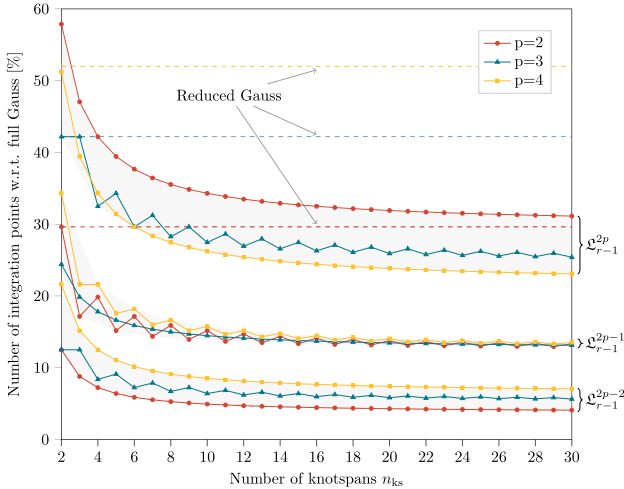


Fig. 6. Number of integration points for generalized Gaussian quadrature rules in comparison to knot span-wise Gaussian quadrature for a B-Spline unit cube with C^{p-1} continuity and n_{ks} knot spans in each spatial direction.

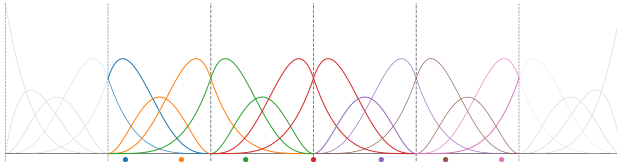


Fig. 7. Distribution of integration points: generalized Gaussian quadrature on trimmed knot vector with $n_{ks} = 4$ active knot spans, $q = 4$ and $s = 1$.

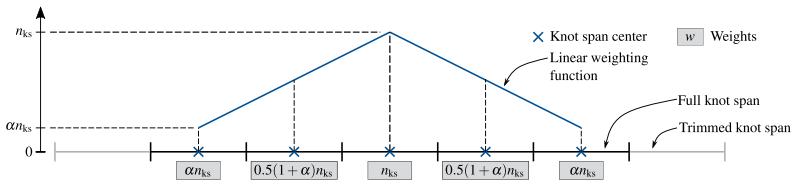


Fig. 8. Weighting of knot spans. The x -axis represents a trimmed knot vector with $n_{ks} = 5$ consecutive full knot spans.

and thus enable the use of GGQ rules. Generally, the larger each subdomain, the greater the savings in the number of required integration points (see Fig. 6).

In a first step, consecutive interior knot spans are collected and weighted according to the size of their group, more precisely, the number of knot spans n_{ks} that are included. Based on a parameter $0 < \alpha < 1$, a linear weighting function is introduced to give additional importance to center knot spans, as depicted in Fig. 8. This process is performed in each spatial direction. Subsequently, the directional weights w^x , w^y , and w^z are multiplied to form one global weight w per knot span. The complete algorithm is schematically depicted in Fig. 9, where the color map visualizes the global weights based on $\alpha = 0.1$. For illustration purposes, the respective representation is

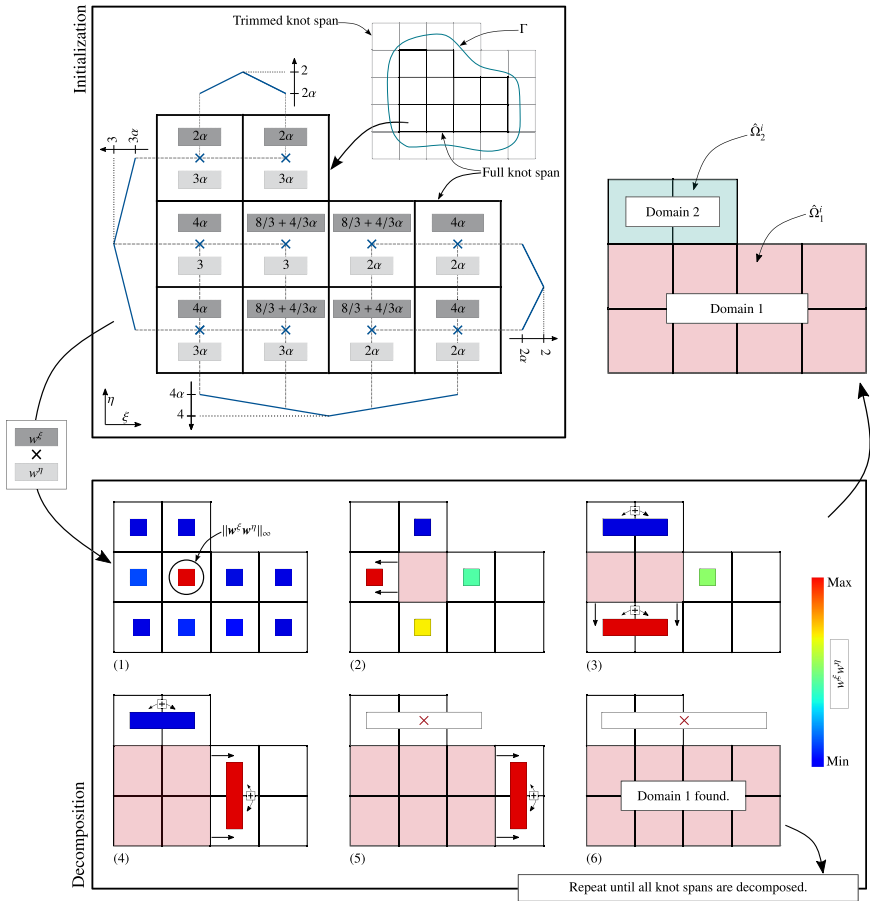


Fig. 9. Tensor-product decomposition algorithm. The color map visualizes the global knot span weights $w = w^\xi w^\eta$ that are initialized according to Fig. 8. The already found tensor product domain is indicated by fully filled knot spans. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

limited to two dimensions, but the application to three dimensions is straightforward. After the initialization phase, the algorithm enters the decomposition loop, where the knot span with the largest weight ($w = \|w^\xi w^\eta\|_\infty$) is defined as the starting point for the first tensor product domain. The domain is successively expanded in the direction of its neighbor with maximal weight while maintaining a rectangular/cuboid shape in two/three dimensions. If the already found domain is adjacent to more than one knot span per direction, their weights are summed up. The current decomposition loop ends if no direction allows to make a valid move, and a new domain is started until all knot spans are decomposed. Once the decomposition is complete, each domain is defined as one macro element $\hat{\Omega}^i$, allowing the construction of a domain-wise tensor product quadrature rule according to Eqs. (16) and (17).

Remark. In principle, it is reasonable to define α with any value between $0 < \alpha < 1$. For values close to $\alpha = 0$, knot spans at the active boundaries are weighted significantly less compared to interior ones, while $\alpha = 1$ results in an equal weighting for all knot spans within one group (see Fig. 8). Generally, it is advantageous to put more importance on interior domains. In addition, however, boundary knot spans should not be neglected entirely. Thus, we propose to initialize all weights with $\alpha = 0.1$. Note that the presented algorithm does not necessarily provide the best possible decomposition regarding the final number of quadrature points in all scenarios. This would require the solution of a complex optimization problem. Nevertheless, since the savings in integration points are already close to the maximum for a relatively small number of knot spans (see Fig. 6), the additional effort would not bring a significant improvement.

3.2. Integration of trimmed knot spans

Trimmed or cut elements pose a difficult challenge in most embedded boundary methods since their underlying integration scheme must guarantee sufficient accuracy for arbitrarily shaped domains. An octree refinement, as used in the original version of the FCM [17], can tackle such problems and is known to work robustly on various shapes. However, already a moderate octree depth can lead to a vast number of integration points. A similar overhead must be expected if the integration is performed on a lower-order tessellation based on tetrahedrons, as per [57]. In [58], the so-called smart octree is developed, which incorporates a node-relocation algorithm. Due to the smart-octree’s higher flexibility, better convergence rates are achieved, which entails a reduced number of integration points. The authors in [59] compare the smart octree to the moment fitting approach and show that solving the moment fitting equation can further decrease the number of integration points while attaining the same accuracy.

In the following, we propose a combined approach that solves the moment fitting equation during the execution of an iterative point elimination algorithm. A novel solution strategy is presented, which establishes an upper bound for the final number of integration points. As a result, the constructed quadrature rules require equal or fewer function evaluations than full Gaussian quadrature for arbitrarily shaped domains.

3.2.1. Moment fitting equation

Given a polynomial function $h(\xi)$ of degree q over a continuous parametric domain, its integral can be approximated by n_q quadrature points

$$\int h(\xi) \, d\xi \approx \sum_{i=1}^{n_q} h(\xi^i) w^i, \tag{25}$$

where ξ^i and w^i are the distinct point positions and integration weights. We may rewrite the polynomial $h(\xi)$ as a sum of $m = q + 1$ independent basis functions $f_j(\xi)$ and constant coefficients β_j

$$h(\xi) = \sum_{j=1}^m \beta_j f_j(\xi). \tag{26}$$

The moment fitting approach suggests to seek for n_q integration points to approximately evaluate each basis among the given set

$$\begin{aligned} \int h(\xi) \, d\xi &= \sum_{j=1}^m \beta_j \int f_j(\xi) \, d\xi, \\ &\approx \sum_{j=1}^m \beta_j \sum_{i=1}^{n_q} f_j(\xi^i) w^i. \end{aligned} \tag{27}$$

Finally, the terms on the right-hand side in Eq. (27) are rearranged, and the formulation is adapted for the integration of the three-dimensional trimmed domain $\hat{\Omega}^t$

$$\begin{bmatrix} f_1(\xi^1) & \cdots & f_1(\xi^{n_q}) \\ \vdots & \ddots & \vdots \\ f_m(\xi^1) & \cdots & f_m(\xi^{n_q}) \end{bmatrix} \begin{bmatrix} w^1 \\ \vdots \\ w^{n_q} \end{bmatrix} = \begin{bmatrix} \int_{\hat{\Omega}^t} f_1(\xi) \, d\hat{\Omega}^t \\ \vdots \\ \int_{\hat{\Omega}^t} f_m(\xi) \, d\hat{\Omega}^t \end{bmatrix}, \tag{28}$$

with $\xi^i = (\xi^i, \eta^i, \zeta^i)$. The moment fitting approach can be interpreted as a procedure to optimize an a priori defined set of integration points toward a known reference solution $f_c = [\int_{\Omega'} f_1(\xi) d\Omega', \dots]^T$. These are commonly referred to as moments or the constant terms of the moment fitting equation.

The fact that Gaussian quadrature yields exact results for $q \leq 2n_q - 1$ implies that the choice of an appropriate quadrature rule depends solely on the maximum polynomial degree q of the given integrand. Eq. (28) illustrates this proposition, as all coefficients β_i introduced in Eq. (26) have disappeared. Thus, in the univariate case, we may choose any set of linearly independent functions \mathfrak{F}_{1D} capable of representing the integrand $h(\xi)$ of order q as moment fitting bases f_j

$$\mathfrak{F}_{1D} = \{f_j(\xi) = \tilde{L}_r(\xi); r = 0, 1, 2, \dots, q\}. \tag{29}$$

However, to provide orthogonal bases on each knot span $[\xi_i, \xi_{i+1}]$, \tilde{L}_r is defined as

$$\tilde{L}_r = L_r \left(\frac{2\xi - \xi_i - \xi_{i+1}}{\xi_{i+1} - \xi_i} \right), \tag{30}$$

with L_r being the r th Legendre polynomial. For a tensor product space, the moment fitting equation could be solved independently in each spatial direction since, e.g., η and ζ appear as constant terms in the integral along ξ . However, if the domain is trimmed, the same integral can be bounded by an arbitrary function that depends on η and ζ and vice versa. Following the notation introduced earlier, the moment fitting bases in three dimensions may hence be defined by a tensor product, as per

$$\mathfrak{F}_{3D} = \{f_j(\xi) = \tilde{L}_r(\xi)\tilde{L}_s(\eta)\tilde{L}_t(\zeta); r, s, t = 0, 1, 2, \dots, q\}, \tag{31}$$

resulting in $m = (q + 1)^3$ functions. As the weak form in Eqs. (23) and (24) dictates an integrand of order $q = 2p$, $m = 343$ moment fitting bases would emerge if cubic shape functions are applied. For a more detailed discussion on the necessary maximum polynomial degree in \mathfrak{F}_{3D} , let us divide the univariate function $h(\xi)$ of order no greater than $2n_q - 1$ by a function $L(\xi)$ of order n_q

$$\underbrace{h(\xi)}_{q=2n_q-1} = \underbrace{q(\xi)}_{n_q-1} \underbrace{L(\xi)}_{n_q} + \underbrace{r(\xi)}_{n_q-1}. \tag{32}$$

Accordingly, the resulting quotient $q(\xi)$ and the remainder $r(\xi)$ are of order $n_q - 1$ or less. In Gaussian quadrature, L is defined as the n_q th Legendre polynomial L_{n_q} . Since L_{n_q} is orthogonal to all polynomials of order $q < n_q$ on $[-1, 1]$, the integral of $h(\xi)$ over this particular domain simplifies to the integration of the remainder $r(\xi)$

$$\int_{-1}^1 h(\xi) d\xi = \int_{-1}^1 q(\xi) L_{n_q}(\xi) d\xi + \int_{-1}^1 r(\xi) d\xi. \tag{33}$$

For the respective inner product to be zero, the Gaussian quadrature points are defined as the n_q distinct roots of L_{n_q} . The corresponding weights can be found by solving a system of equations, which takes the form of a linearized moment fitting equation with $\mathfrak{F}_{1D} = \{f_j = L_r(\xi); r = 0, 1, 2, \dots, n_q - 1\}$. Indeed, if orthogonal polynomials exist for the given integration domain, the positions of the integration points are inherently given. Moreover, it seems sufficient to restrict the moment fitting bases to $m = n_q = (q/2 + 1)^d$ functions, where d is the spatial dimension. Considering the trivariate example with cubic bases introduced above, the number of required functions drastically reduces from $m = 343$ to $m = 64$.

Unfortunately, orthogonal polynomials are difficult to construct or might not even exist for arbitrarily shaped domains in multiple dimensions. Due to this problem, the nonlinear moment fitting equation may be utilized to satisfy $\int q(\xi) L_{n_q}(\xi) d\xi \approx 0$ implicitly, without any knowledge about the form of L_{n_q} . For the univariate case, this approach is shown to be successful in Section 3.1 since the computation of the generalized Gaussian quadrature rule for multiple knot spans is deduced from a similar nonlinear system of equations. However, the computationally expensive task of solving the nonlinear problem was circumvented by using precomputed integration points. This is not applicable in the present context due to arbitrary integral boundaries. Moreover, the moment fitting equation (Eq. (28)) is defined in three dimensions, making its solution even more complicated and factually infeasible during the execution of the simulation. Thus, in Section 3.2.3, the moment fitting equation is linearized by defining the positions of the integration points a priori in order to reduce the computational effort. Additionally, a point elimination algorithm based on the original ideas from [28,60] is utilized to select the most suitable positions from

a discretized set. The proposed approach allows to restrict the moment fitting bases to $m = (q/2 + 1)^3$ functions in three dimensions while no significant degradation in accuracy is observed, see Section 6.2. Before the mentioned solution strategies are presented, we will focus on calculating the constant terms f_c in the next section.

Remark. Since the moment fitting equation in Eq. (28) is defined in the parametric space, the resulting quadrature rules are tailored to integrate Eq. (16). The additional mapping to a Gaussian space as stated in Eq. (17) is omitted.

3.2.2. Computation of constant terms

In general, every integration scheme listed in the introduction of this section, such as octree, smart-octree, and lower-order tessellation, are suitable candidates to compute the constant terms in Eq. (28). However, a different methodology enables a more seamless integration into the presented workflow due to the inherently available boundary representation. To this end, the moments in Eq. (28) are evaluated by integrating over the boundary surfaces utilizing the divergence theorem in a similar fashion as described by [61]

$$\int_{\Omega'} \mathcal{F}_j(\mathbf{x}) \, d\Omega' = \int_{\Omega'} \nabla \cdot \mathbf{g}_j(\mathbf{x}) \, d\Omega' = \int_{\Gamma'} \mathbf{g}_j(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) \, d\Gamma', \tag{34}$$

where $\mathcal{F}_j(\mathbf{x})$ represents the moment fitting bases defined on the physical domain. Eq. (34) realizes a transformation from volume to contour integrals with $\mathbf{n}(\mathbf{x})$ denoting the normal vector pointing in outwards direction of the geometry. For the anti-derivatives \mathbf{g}_j , the notation is adopted from [62]

$$\mathbf{g}_j(\mathbf{x}) = \frac{1}{3} \begin{bmatrix} \int \mathcal{F}_j(\mathbf{x}) \, dx \\ \int \mathcal{F}_j(\mathbf{x}) \, dy \\ \int \mathcal{F}_j(\mathbf{x}) \, dz \end{bmatrix} = \frac{1}{3} \begin{bmatrix} \int f_j(\xi) \frac{\partial x}{\partial \xi} \, d\xi \\ \int f_j(\xi) \frac{\partial y}{\partial \eta} \, d\eta \\ \int f_j(\xi) \frac{\partial z}{\partial \zeta} \, d\zeta \end{bmatrix}. \tag{35}$$

Gaussian quadrature seems to be predestined for an efficient evaluation of Eq. (34). However, numerical integration requires a closed surface parameterization of the trimmed domain, which may be obtained from solid-to-solid intersection algorithms. Most CAD programs, e.g., *Rhinoceros 3D*, include functionalities to conduct such Boolean operations directly on the NURBS-based B-Rep model. Nevertheless, similar to the knot span classification problem mentioned earlier, it can be advantageous to perform these operations on an intermediate discretized surface description, e.g., STL mesh. Among other software packages, *CGAL* provides an efficient and robust implementation for algorithms required in this context. Moreover, the resulting B-Rep of the intersecting domain is inherently discretized and can directly be used for Gaussian quadrature. Given a boundary parameterization with n_F triangular elements, the integral is performed as follows

$$\int_{\Omega'} \mathcal{F}_j(\mathbf{x}) \, d\Omega' = \sum_{a=1}^{n_F} \int_{\hat{\Gamma}_a^h} \mathbf{g}_j(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) \, \det(\mathbf{J}_{F_a^h}) \, d\hat{\Gamma}_a^h, \tag{36}$$

with $\det(\mathbf{J}_{F^h})$ denoting the determinant of the Jacobian matrix, which accounts for the mapping between physical (Γ^h) a parametric space ($\hat{\Gamma}^h$) of each boundary triangle F^h . Finally, the integral is retracted to the parametric space of the B-Spline solid

$$\int_{\hat{\Omega}'} f_j(\xi) \, d\hat{\Omega}' = \int_{\Omega'} \mathcal{F}_j(\mathbf{x}) \frac{1}{\det(\mathbf{J}_1)} \, d\Omega'. \tag{37}$$

Remark. In [63], Sudhakar et al. point out that

$$\int_{\Omega} d\Omega = \int_{\Gamma} xn_1 d\Gamma = \int_{\Gamma} yn_2 d\Gamma = \int_{\Gamma} zn_3 d\Gamma, \tag{38}$$

and hence suggest to restrict the evaluation in Eq. (35) to a single direction, e.g., $\mathbf{g}_j(\mathbf{x}) = [\int \mathcal{F}_j(\mathbf{x}) \, dx, 0, 0]$. Nevertheless, as Eq. (35) may seem more intuitive, we adopted the notation from [62]. In any case, the known basis functions f_j allow the calculation of the integrals in Eq. (35) by analytical means. In [64], a quadrature-free approach for evaluating polynomials over spline-based B-Reps is proposed. The divergence theorem is applied twice to transform the volume integrals first to surface integrals and then to line integrals. In the present work, we omit the second transformation because Eq. (36) agrees with the terms required to impose boundary conditions (see

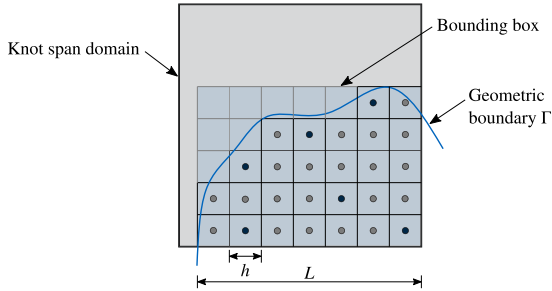


Fig. 10. Initial point distribution within a trimmed domain. Dark blue: Selected points by elimination algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Eq. (18), allowing a fast integration into traditional finite element frameworks. Note that the computed closed surface parameterizations obtained from the intersection of the knot span domains and the computational domain respect the knot span boundaries by definition. Therefore, the corresponding discretizations are also suitable for the evaluation of Eq. (18).

3.2.3. Point elimination algorithm

As highlighted in Section 3.2.1, Eq. (28) is linearly dependent on the weights w^i but strongly nonlinearly dependent on the positions ξ^i of the integration points. To reduce the computational effort, it is a common approach to linearize the moment fitting equation by defining the positions of the integration points a priori [65]. The authors in [62] propose to place the quadrature points at the locations of the standard Gauss points, which, however, does not guarantee that all quadrature points are inside the physical domain. Especially for nonlinear simulations where material variables, e.g., plasticity values, have to be stored, points outside the domain are unfavorable. Therefore, in the present work, interior points are selected from the resulting locations of the adaptive point distribution scheme developed by [61,66]. The distribution scheme defines the positions based on a regular grid, which classically covers the space spanned by one finite element/cell. In our proposed implementation, the point distribution is restricted to a bounding box within the parametric space of one knot span that delineates the geometry boundaries, as depicted in Fig. 10. Especially if only small portions of an element intersect, this approach reduces the number of requisite inside/outside tests significantly. A point distribution factor $\gamma \in \mathbb{N}$ is introduced to define the discretization length h

$$h = \frac{L}{\gamma (p + 1)}, \tag{39}$$

determining the emerging number of points. Note that h , L , and p might differ in each spatial direction, which is neglected for brevity. In general, γ is initialized large enough ($\gamma \geq 2$) such that the initial number of points n_q exceeds the number of moment fitting bases $m = (p + 1)^3$. For the solution of such an under-determined system, the moment fitting equation (Eq. (28)) is reformulated to a least squares problem reducing the L2-norm of the residual vector \mathbf{r} for the given set of points (ξ, \mathbf{w})

$$\mathbf{q}(\xi) := \mathbf{f}_c - \mathbf{f}(\xi)\mathbf{w} = \mathbf{r}. \tag{40}$$

To improve the solution, Eq. (40) is embedded into the iterative process of an elimination algorithm that detects points with more suitable locations and dismisses suboptimal ones. Our implementation builds on the first proposals of [60] and further developments by [28]. In agreement with [28], we expect the algorithm to find an approximate solution to the nonlinear problem. In addition, if the solution is sufficiently accurate, we assume that the zero condition in Eq. (33) is also approximately fulfilled. Therefore, we claim that using only $m = (p + 1)^3$ moment fitting bases is reasonable.

Remark. Note that in a worst-case scenario where the zero condition in Eq. (33) is not satisfied, the obtained integration points would represent a reduced quadrature rule, which can still evaluate integrands of order p but not necessarily $2p$ accurately. Section 6.2 compares the proposed reduced set ($m = (p + 1)^3$) to the full set of moment fitting bases ($m = (2p + 1)^3$) and illustrates the impact on the solution quality.

Moreover, the extension recently presented in [29] is incorporated to guarantee positive weights without additional feasibility constraints and to decrease the number of required elimination loops. The respective concept is discussed in the following.

During the first iteration ($k = 1$) of the elimination algorithm, the moment fitting equation is solved for the weights \mathbf{w} of the initial trial points obtained from the modified adaptive point distribution scheme. The key idea is to consecutively remove quadrature points with the least significance for evaluating the constant terms \mathbf{f}_c [28,60]. In the present work, the significance of the quadrature point is measured by its computed weight w^i . As proposed in [29], a Non-Negative Least Squares (NNLS) [67] solver is employed to ensure positiveness for all elements in \mathbf{w} without any additional feasibility constraint. Note that a standard least squares solver does not generally satisfy this condition. Since we enter the elimination algorithm with an under-determined system of equations ($n_q \gg m$), the NNLS solver converges to a solution vector \mathbf{w} that contains multiple entries equal to zero. As integration points with $w_i = 0$ have no significance for the final quadrature rule, they are all discarded after the first iteration, resulting in a considerable reduction of required elimination loops. Additionally, the one integration point with the lowest positive significance is also removed since, without its elimination, the results would remain unchanged in the second iteration. In all subsequent iterations, only the integration point that exhibits the smallest significance is removed from the list of potential solutions. Fig. 11 visualizes the main routines of the elimination algorithm in a flowchart. The algorithm terminates if the L2-norm of the current residual vector exceeds a user-defined value δ and returns the integration points from the penultimate iteration. If no subset of the initially distributed trial points can be found to satisfy Eq. (40), such that $\|\mathbf{r}\|_{L2} < \delta$, γ is automatically increased to enrich the discrete solution space. An upper bound on γ may be introduced for performance purposes in practical applications. The presented point elimination algorithm provides a robust strategy for the construction of efficient integration rules for arbitrarily trimmed domains. In three dimensions, the resulting number of quadrature points is limited to $n_q \leq (p + 1)^3$. A detailed discussion on the performance of the algorithm is given in Section 6.

4. Numerical stability

Trimming and the evaluation of trimmed domains are key features of the presented method. Due to little support of basis functions within small trimmed knot spans, linear dependencies may be introduced to the system matrices in Eq. (12) [68]. As a result, the respective condition number can assume values that prevent the use of iterative solvers or substantially reduce the accuracy of direct solvers.

To counteract ill-conditioned system matrices, one class of methods introduces quadrature points with scaled integration weights into the fictitious domain to weakly penalize the exterior part of each trimmed element [18,20,69]. Following this approach, the fictitious domain is interpreted as a soft material rather than an empty void. The authors in [70] propose to remove basis functions with supports below a certain threshold entirely. In CutFEM [16,71], a ghost penalty term is applied to add an artificial stiffness to nodes that may precipitate numerical instability. Usually, these nodes are weakly coupled to one or multiple of their stable neighbors. Depending on the magnitude of the penalization factor, the artificially introduced weight, or the ratio of neglected basis functions, a modeling error must be expected for the methods mentioned. Preconditioners are mathematically consistent and proposed to heal ill-conditioning of immersed boundary methods. In fact, an Additive-Schwarz inspired preconditioner developed in [68] has proven to be an effective and robust tool, even in the presence of higher-order basis functions. The basic concept is outlined in the following.

4.1. Additive Schwarz preconditioner for ill-conditioned system matrices

The Additive-Schwarz preconditioner \mathbf{P} is defined as a sparse block-wise approximation of the inverse of the system matrix \mathbf{A} , as per

$$\mathbf{P} = \sum_{B \in \mathcal{B}} \mathbf{R}_B^T \underbrace{(\mathbf{R}_B \mathbf{A} \mathbf{R}_B^T)^{-1}}_{\Lambda_B^{-1}} \mathbf{R}_B. \tag{41}$$

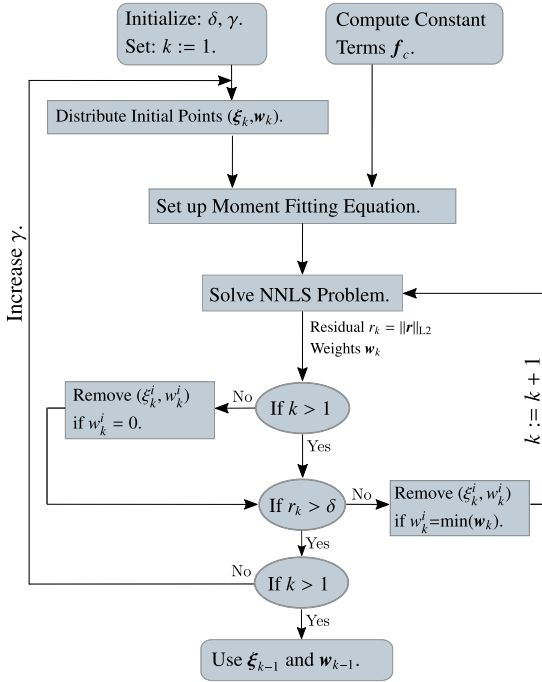


Fig. 11. Flowchart of point elimination algorithm.

Thereby, $\mathbf{R}_B \mathbf{A} \mathbf{R}_B^T$ restricts matrix \mathbf{A} to row and column indices contained in block $B \in \mathcal{B}$, where \mathcal{B} is the set of all blocks. Each sub-matrix \mathbf{A}_B is inverted, and its entries are mapped onto \mathbf{P} by the term $\mathbf{R}_B^T \mathbf{A}_B^{-1} \mathbf{R}_B$, such that their original row and column indices are preserved. This procedure is conducted for each block B . The blocks may overlap, but each basis function must be contained in at least one block to avoid \mathbf{P} being singular. In the context of this work, the degrees of freedom associated with one knot span form an individual block B . For further details on the theoretical background, the implementation of such preconditioners, and their application to embedded boundary methods, the reader is referred to [68,72]. Moreover, the authors in [73] extend the given preconditioner to multigrid approaches. In [74], Additive-Schwarz preconditioners are discussed in a general FEM setting. To demonstrate the proposed method’s scalability potential, all static and implicit dynamic examples in Sections 6 and 7 are solved with an Additive-Schwarz preconditioned iterative Biconjugate Gradient Stabilized Method (BiCSTAB). Nevertheless, direct solvers are also appropriate for problems performed as part of this work.

4.2. Light control points in explicit dynamics

Since explicit solvers do not solve the system of equations, they are not constrained by ill-conditioned matrices. However, the influence of small trimmed knot spans is still perceptible in the form of so-called light control points characterized by small mass and stiffness terms. They are not harmful per se, as their influence on the solution is negligible by definition. Nevertheless, the displacements, velocities, and accelerations at light control points tend to take on extremely large values, which can cause the solver to terminate prematurely due to overflow errors [75]. In the scope of this work, explicit simulations successfully terminated without additional stabilization. Generally, light

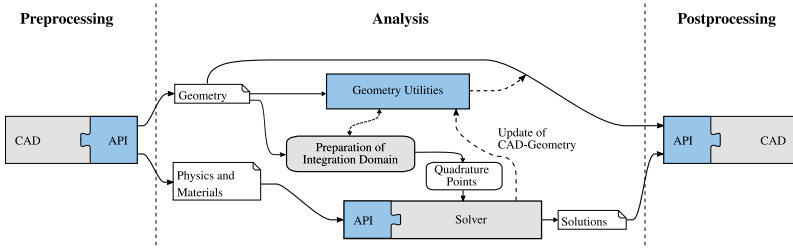


Fig. 12. Required processes and interfaces for a design-through-analysis workflow for solid CAD models. Source: Modified from [76].

control points are especially critical for highly dynamic nonlinear problems, such as impact and crash simulations, where stabilization techniques may be considered. In [75], an effective method for stabilizing light control points is presented, along with a comprehensive overview of other possible stabilization strategies.

5. CAD-integrated analysis workflow

This research aims to develop methods for seamless simulations within the CAD design environment focusing on solid models. In [76], a workflow and a set of interfaces are proposed to close the gap between CAD and classical FEM solvers in the scope of the shell-based IBRA. Essential steps are adopted and extended by some functionalities for the application to volumetric B-Reps, as depicted in Fig. 12. The entire process is fully automatized and, unlike classical finite element analyses on hexahedral meshes, requires no manual intervention to ensure high-quality discretizations of solid structures. An overview of the necessary routines and operations is provided in this section.

5.1. Preprocessing

The proposed CAD-integrated analysis workflow begins within the CAD environment, where the B-Rep model is prepared for analysis. This includes:

- Geometric modeling of the structure to be investigated.
- Definition of the solid B-Spline domain Ω_s , which can be determined automatically or by a user input.
- Definition of the boundary domains Γ_D and Γ_N for the application of boundary conditions.
- Writing the input file for the FE solver according to [76].

The geometric modeling kernel from *Rhinoceros 3D* [77] is utilized for the design of all structures investigated in Sections 6 and 7. All remaining preprocessing operations are performed in *Cocodrilo* [78,79] - an IGA preprocessor for *Rhinoceros 3D*.

5.2. Preparation of the integration domains and construction of quadrature rules

The centerpiece of the present workflow is the conversion of the CAD model into an integration domain suitable for FEM analysis, resulting in a trimmed parametric space of the B-Spline solid \mathcal{S} , see Section 3. To facilitate the integration into state-of-the-art FE solvers, a *meshless* approach that relies on the exchange of information via different sets of quadrature points is followed. Given Eqs. (8)–(10), these sets serve for the integration of:

- Mass and stiffness matrices in Ω .
- Neumann boundary conditions on Γ_N .
- Dirichlet boundary conditions on Γ_D .

Note that quadrature points associated with boundary conditions may carry additional information, e.g., prescribed displacement $\bar{\mathbf{u}}$ and traction force $\bar{\mathbf{t}}$. As a result, all necessary calculations can be performed in existing FE solvers without code duplication and methodological verification, enabling rapid integration. In addition, the above operations to prepare a model suitable for analysis can be completed at the beginning of the simulation and do not need to be repeated. Especially for transient problems with many time steps, the computational effort associated with the preparation of the integration domains becomes insignificant compared to the total simulation time. All processes and algorithms related to the automatic construction of quadrature rules presented in Section 3 are implemented by the corresponding author in *Trivariate Isogeometric B-Rep Analysis (TIBRA)* [80]. The underlying code is written in C++ and uses the *Polygon Mesh Processing* package from *CGAL* [47] to perform necessary geometrical operations. The data transfer between *TIBRA* to third-party software is realized either via file exchange or an extensive python interface. This also allows the necessary information to be easily passed on to commercial solvers.

5.3. FE solver

Given the setup discussed in the previous sections, the FE solver must perform the following fundamental operations:

- Read the sets of quadrature points.
- Pre-evaluate the B-Spline basis functions, the normal vectors, and the material properties at each quadrature point.
- Assemble and solve the system of equations.
- Write the output for the postprocessor.

In the scope of this work, all examples are solved using the open-source FE framework *Kratos Multiphysics* [81–83]. Necessary implementations required for the analysis of trimmed B-Spline solids are made by the authors. In [84], we show preliminary results of highly dynamic nonlinear simulations performed in *LS-DYNA* [85], including large deformation and contact mechanisms.

5.4. Postprocessing

For the visualization of results, such as displacements and stresses, an auxiliary mesh is defined on the NURBS-based B-Rep model. This is a standard procedure in CAD programs like *Rhinoceros 3D* to facilitate color mapping on smooth surfaces. Here, the required mesh is constructed by tessellating the integration points of the boundary surfaces F . In order to reduce the complexity to two dimensions, the tessellation is performed in the respective parametric spaces. Additional points are introduced at each surface's trimming or delimiting edges, ensuring that the mesh accurately represents the geometry. Finally, in a consecutive step, simulation resultants are either interpolated (e.g., displacements) or evaluated (e.g., stresses) at each mesh vertex to provide a smooth visualization on the NURBS-based B-Rep model. Moreover, the *Universal Deformation Technologies* from *Rhinoceros 3D* are utilized to deform the B-Rep model according to the displacements of the control points $P_{i,j,k}$ of the B-Spline solid S . Therefore, S is defined as a so-called *control cage* that dictates the deformation of its *captives objects*, namely the surfaces F contained in the B-Rep model. For more information, the reader is referred to [77]. The corresponding interface is realized in *Cocodrilo* [78].

6. Scientific benchmarks

In the following, we demonstrate the efficacy of the proposed method for the analysis of static and transient problems. All examples are conducted on a C^{p-1} B-Spline discretization constructed by standard k-refinement [1,34]. The contour parameterization of each trimmed knot span required for evaluating Eq. (36) is retrieved from the intersection of the knot span domain and an STL mesh. The corresponding tessellation is automatically performed in *Rhinoceros 3D* with a chordal tolerance of 10^{-5} (see Appendix A). Subsequently, the obtained intersection mesh is homogenized using an isotropic remeshing algorithm [47], targeting a prescribed minimum number of triangles. This lower bound is introduced to ensure a high-quality contour parameterization of the trimmed domains. Note that rather conservative thresholds are used in the following examples to avoid the results being affected by the quality of the constant terms of the moment fitting equation. The interested reader is referred to [61], where the

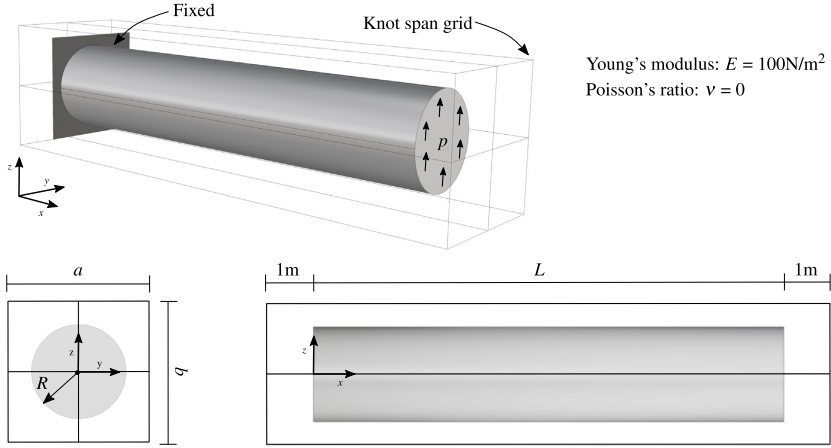


Fig. 13. Cantilever subjected to a surface load: Configuration of the beam and solid B-Spline discretization.

influence of the number of elements on the quality of the constant terms in Eq. (28) is studied. In the following, we further increase the accuracy of the respective volume integrals by using three instead of one quadrature point per boundary triangle. The allowed residual for the moment fitting equation is fixed to $\delta = 10^{-10}$ unless stated otherwise. Moreover, the point distribution factor is initialized with $\gamma = 2$. For the sake of comprehension, all depicted knot span domains are visualized in physical space.

6.1. Trimmed cantilever

In the first example, the influence of trimming on the solution quality of static problems is investigated. A trimmed cantilever with a circular cross-section is subjected to a tip load. The structure is fully embedded into the B-Spline solid S , as depicted in Fig. 13, where $a = 3$ m, $b = 3$ m, $R = 1$ m, and $L = 10$ m. Homogeneous Dirichlet boundary conditions are enforced using a penalty factor of $\beta = 10^{10}$ N/m³ at $x = 0$ m. The tip load is modeled as a surface load with $p = 0.1$ N/m² and applied over the structure's boundary at $x = L$. The discretization of the cross-section is fixed to 2×2 knot spans, whereas the number of knot spans along x is parameterized with n_{ks}^x . Consequently, all knot spans in the B-Spline solid are trimmed. Given the predefined chordal tolerance of 10^{-5} , the tessellation algorithm generates a B-Rep model with 2808 elements. Furthermore, the computed solid-to-solid intersections are parameterized with approximately 3000 boundary triangles used to evaluate Eq. (36).

According to [86], the analytical solution of such a Timoshenko beam is given as

$$w^e = \frac{px^2(3L - x)}{6EI} + \frac{px}{GA\kappa}, \tag{42}$$

with EI and G denoting the bending stiffness and shear modulus, respectively. The shear coefficient $\kappa = (6 + 12\nu + 6\nu^2)/(7 + 12\nu + 4\nu^2)$ accounts for the varying shear stress distribution across the circular cross-section with area A [87]. Fig. 14 shows the relative error in the vertical displacement with respect to the analytical solution w^e over the entire length of the beam for five different meshes. The first four simulations are conducted using a constant polynomial degree of $p = 2$, but a varying number of knot spans in x -direction: $n_{ks}^x = 3$, $n_{ks}^x = 4$, $n_{ks}^x = 5$ and $n_{ks}^x = 10$. The corresponding maximum relative errors are: 1.3%, 0.47%, 0.19%, and 0.015%. In all cases, the accuracy can be further increased by raising the polynomial degree. For example, when p is elevated to cubic order, a single trimmed knot span in x -direction yields a relative error of 0.03%.

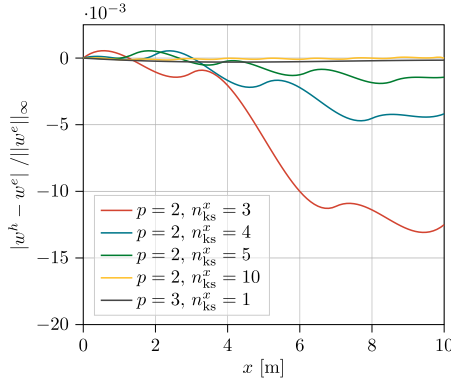


Fig. 14. Cantilever subjected to a surface load: Relative error in displacement for different meshes, indicating convergence.

Table 1

Convergence of point elimination algorithm. The current residual $\|r\|_{L_2}$ and the number of quadrature points n_q are given for one trimmed knot span. Point distribution factor $\gamma = 2$. Prescribed tolerance $\delta = 10^{-10}$.

$p = 2, n_{ks}^x = 10$			$p = 3, n_{ks}^x = 1$		
Iteration k	$\ r\ _{L_2}$	n_q	Iteration k	$\ r\ _{L_2}$	n_q
1	$<10^{-15}$	168	1	$<10^{-15}$	416
2	$<10^{-15}$	27	2	$<10^{-15}$	64
3	1.21×10^{-5}	26	3	1.74×10^{-8}	63

As discussed in Section 3.2.1, the moment fitting equation is assembled with $m = (p + 1)^3$ basis functions. Therefore, the system of equations (Eq. (28)) is determined for $n_q = (p + 1)^3$, corresponding to the exact number of integration points required for full Gaussian quadrature. Table 1 lists the number of integration points and the obtained residual $\|r\|_{L_2}$ during the execution of the point elimination algorithm for one knot span. Note that due to the simple shape of the structure, all trimmed domains are similar and thus give equivalent results. Already the predefined initial distribution factor of $\gamma = 2$ leads to a set of points that satisfies Eq. (40) with machine precision. Furthermore, the NNLS solver in conjunction with the orthogonal Legendre polynomials as moment fitting bases allows the algorithm to converge to $n_q = (p + 1)^3$ after the first iteration. This behavior is observed independent of the employed polynomial degree. The values are shown for $p = 2, n_{ks}^x = 10$ and $p = 3, n_{ks}^x = 1$ in Table 1. Due to a predefined maximum error norm of $\delta = 10^{-10}$, the algorithm terminates after the third iteration. The size of the final set of quadrature points is highlighted in gray. Fig. 15(a) depicts the Von Mises stresses on the NURBS-based B-Rep model, which exhibit a clear axisymmetric distribution. Next to it, the deformed CAD model and the B-Spline solid discretization is visualized. The postprocessing of stresses and displacements on the CAD B-Rep model is performed in *Rhinoceros 3D* as described in Section 5.4.

The obtained results show that the full potential of higher-order basis functions is exploited despite the presence of trimmed knot spans. Moreover, Table 1 reveals the clear advantage of the presented modified point elimination algorithm, which converges to $n_q = (p + 1)^3$ quadrature points after the first iteration, in contrast to classical methods that eliminate only one point per iteration.

6.2. Thick-walled cylinder subjected to internal pressure

In this section, the performance of the presented method shall be assessed by studying a thick-walled cylinder subjected to internal pressure. Fig. 16 depicts the simulation setup and the four different meshes investigated. Similar

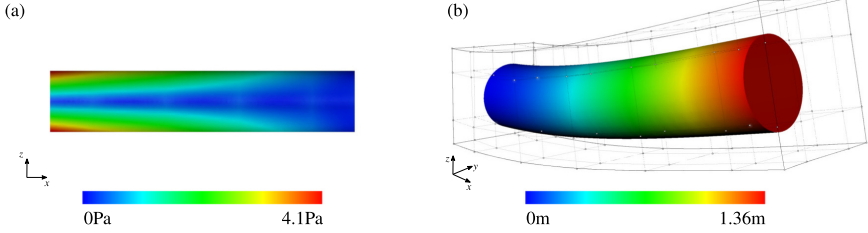


Fig. 15. Postprocessing of the cantilever beam ($p = 2, n_{ks}^x = 5$) on the NURBS-based CAD model in *Rhinoceros 3D*: (a) Von Mises stress and (b) Deformation and active control points.

to [1,12,13], which study the same example in the scope of IGA, plane strain conditions are applied by fixing the longitudinal displacement at the bottom and top surface of the cylinder. The exact solutions for the radial displacement and the stresses are taken from [88] and serve as reference

$$u_r^e(r) = \frac{1}{\bar{E}} \frac{P R_i^2}{R_o^2 - R_i^2} \left((1 - \bar{\nu})r + \frac{R_o^2(1 + \bar{\nu})}{r} \right), \tag{43}$$

$$\sigma_{rr}^e(r) = \frac{P R_i^2}{R_o^2 - R_i^2} - \frac{P R_i^2 R_o^2}{r^2(R_o^2 - R_i^2)}, \tag{44}$$

$$\sigma_{\theta\theta}^e(r) = \frac{P R_i^2}{R_o^2 - R_i^2} + \frac{P R_i^2 R_o^2}{r^2(R_o^2 - R_i^2)}, \tag{45}$$

where $\bar{E} = E/(1 - \nu^2)$ and $\bar{\nu} = \nu/(1 - \nu)$ account for the plane strain conditions. The bottom surface with an inner radius of $R_i = 1$ m and an outer radius of $R_o = 2$ m is extruded to form a cylinder with a length of $L = 5$ m. In all simulations, symmetry conditions are applied to reduce the problem's complexity. The linear elastic material properties are defined by the Young's modulus $E = 40$ N/m² and the Poisson's ratio $\nu = 0.0$. To model the internal pressure, a constant surface load of $p = 20$ N/m² is applied in normal direction of the boundary parameterization of the inner surface. The constant terms of the moment fitting equation are computed from a boundary parameterization with approximately 1500 triangular elements. According to the discussion in Section 3.2.1, the moment fitting bases are defined by a reduced set of $m = (p + 1)^3$ functions. Consequently, Eq. (28) turns into a determined system of equations for $n_q = (p + 1)^3$ quadrature points per trimmed knot span domain. Table 2 shows the convergence behavior of the point elimination algorithm for the two trimmed knot span domains Ω_1^t and Ω_2^t indicated in Fig. 16. Similar to the results obtained in Section 6.1, the algorithm finds $n_q = (p + 1)^3$ quadrature points that satisfy Eq. (40) with machine precision. The numbers of quadrature points in the finally selected sets are highlighted in gray. Given the prescribed tolerance of 10^{-10} , an additional iteration is performed in domain Ω_2^t , resulting in a quadrature rule with one integration point less. Note that Ω_2^t is the smaller of the two trimmed domains. Fig. 17(a) depicts the deformed solid CAD model in *Rhinoceros 3D*. The displacement contour is smooth and symmetric. Fig. 17(b) plots the relative error in radial displacement using quadratic basis functions. The maximum errors for Mesh 1, 2, 3, and 4 are 1.09%, 0.21%, 0.059%, and 0.02%. In all cases, the accuracy can be further increased by order elevation.

Up to this point, all full knot spans have been evaluated by classical knot span-wise Gaussian quadrature. Unlike the optimal and reduced GGQ rules discussed in Section 3.1.1, standard Gaussian quadrature schemes do not exploit the continuity across adjacent knot spans and are therefore less efficient. However, applying GGQ rules to two and three dimensions requires a tensor product structure of the respective knot spans, which is not inherently given for trimmed patches. To enable the use of GGQ rules, the decomposition algorithm presented in Section 3.1.2 is employed. Fig. 18(a) illustrates the decomposed local tensor product domains for Mesh 4, where the numbers indicate the order followed by the algorithm. Note that the order, and thus the final decomposition, can depend on the knot span numbering, see domains 3 and 7. Nevertheless, this does not affect the accuracy of the quadrature rules. Although the visualization is limited to two dimensions, the GGQ rules are also applied in axial direction. For

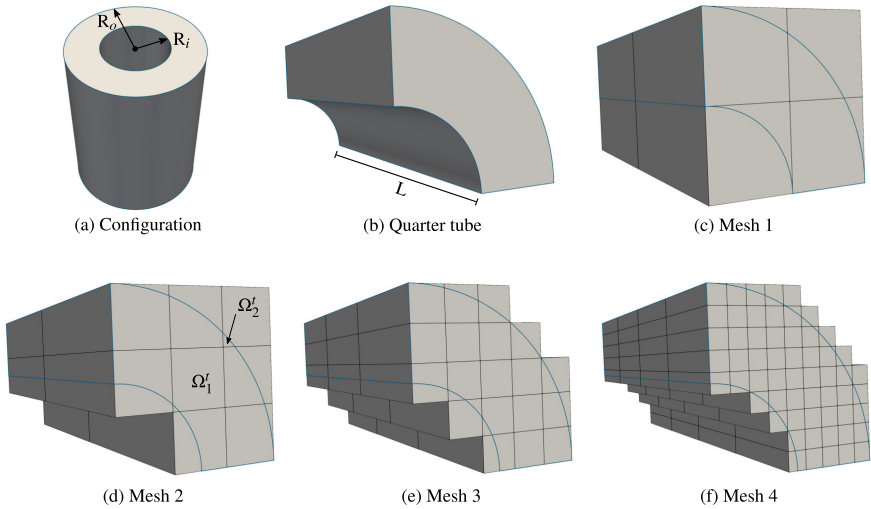


Fig. 16. Thick-walled cylinder: Model configuration and meshes.

Table 2

Convergence of point elimination algorithm. The current residual $\|r\|_{L_2}$ and the number of quadrature points n_q are given for the domains Ω_1' and Ω_2' (see Fig. 16). Point distribution factor $\gamma = 2$. Prescribed tolerance $\delta = 10^{-10}$.

Ω_1'			Ω_2'		
Iteration k	$\ r\ _{L_2}$	n_q	Iteration k	$\ r\ _{L_2}$	n_q
1	$< 10^{-15}$	156	1	$< 10^{-15}$	126
2	$< 10^{-15}$	27	2	$< 10^{-15}$	27
3	3.41×10^{-6}	26	3	8.67×10^{-11}	26
–			4	1.095×10^{-10}	25

demonstration purposes, the decomposition is additionally performed for the full cylinder, as shown in Fig. 18(b). In both cases, the linear weighting function is defined by $\alpha = 0.1$. Based on Mesh 4, Fig. 19(b) compares the performance of the three different quadrature rules associated with the B-Spline target spaces \mathcal{Q}_{r-1}^{2p} , \mathcal{Q}_{r-1}^{2p-1} , and \mathcal{Q}_{r-1}^{2p-2} introduced in Section 3.1.1. Note that the depicted graph is a detailed view of Fig. 17(b). The results highlight the exactness of quadrature rules derived from \mathcal{Q}_{r-1}^{2p} . When using first- and second-order reduced quadrature rules, minor deviations from full Gaussian quadrature are apparent. However, we observe that the discrepancy is of the same order of magnitude as the discretization error in both cases. Fig. 19(a) illustrates the point distributions of the investigated quadrature rules. Despite the modest size of this particular example, the potential savings in required points are significant. While attaining exact integration, the average number of quadrature points inside one knot span domain is reduced from $n_q = 27$ for full Gaussian quadrature to $n_q = 13.8$ for \mathcal{Q}_{r-1}^{2p} . The reduced integration rules \mathcal{Q}_{r-1}^{2p-1} and \mathcal{Q}_{r-1}^{2p-2} decrease this number further to $n_q = 5.9$ and $n_q = 1.9$, respectively. We continue the discussions on the performance of \mathcal{Q}_{r-1}^{2p} , \mathcal{Q}_{r-1}^{2p-1} , and \mathcal{Q}_{r-1}^{2p-2} in Sections 6.3.1 and 6.4.1.

In a second study, the suitability of the proposed reduced set of moment fitting bases using $m = (p+1)^3$ functions (see Section 3.2.1) shall be illustrated. For this purpose, the convergence in energy norm for quadratic, cubic, and quartic B-Spline bases is examined. To obtain a computational domain with uniform edge length, the cylinder’s length is reduced to $L = 2$ m. Initially, $n_{ks} = 2$ knot spans discretize the cylinder in each spatial direction. In an iterative process, we subsequently refine the mesh and compute the relative error in energy norm, which is defined

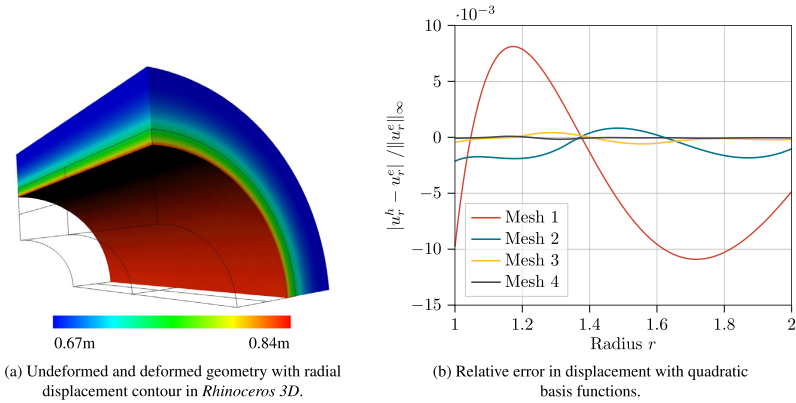


Fig. 17. Thick-walled cylinder: Deformation and convergence of displacement.

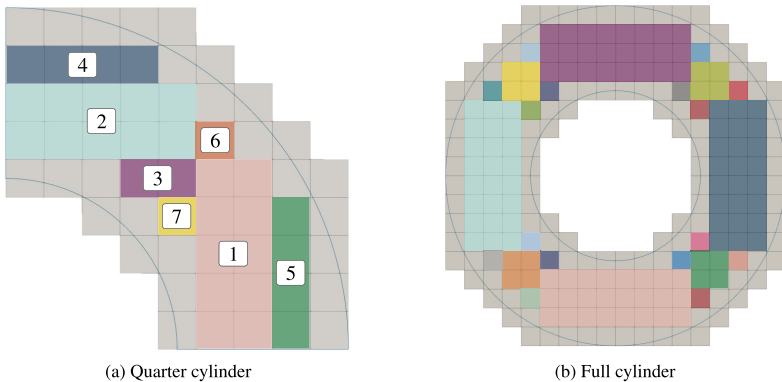


Fig. 18. Thick-walled cylinder: Tensor product decomposition of full knot spans. Colors indicate different domains. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

as follows

$$e_r = \sqrt{\frac{\int_{\Omega} (\sigma^h - \sigma^e) \mathbb{C}^{-1} (\sigma^h - \sigma^e) \, d\Omega}{\int_{\Omega} \sigma^e \mathbb{C}^{-1} \sigma^e \, d\Omega}}, \tag{46}$$

where σ^h and σ^e are the approximated and exact stress tensors, and \mathbb{C} is the constitutive tensor of linear elasticity. Fig. 20 shows the convergence of e_r with respect to the knot span length h . To avoid the results being affected by the quality of the constant terms of the moment fitting equation, we parameterize the trimmed domains with an average of 2500 triangles instead of 1500 used previously. The simulations are performed for $p = 2$, $p = 3$, and $p = 4$ using the proposed reduced set of $m = (p + 1)^3$ moment fitting bases. As a reference, the results for $m = (2p + 1)^3$ are additionally plotted, which reveal optimal quadratic, cubic, and quartic convergence rates. We observe that the reduction of the moment fitting function space is only slightly reflected in Fig. 20. In all cases, nearly optimal convergence rates are achieved. Since $n_q = m$ results in a determined system of equations, the

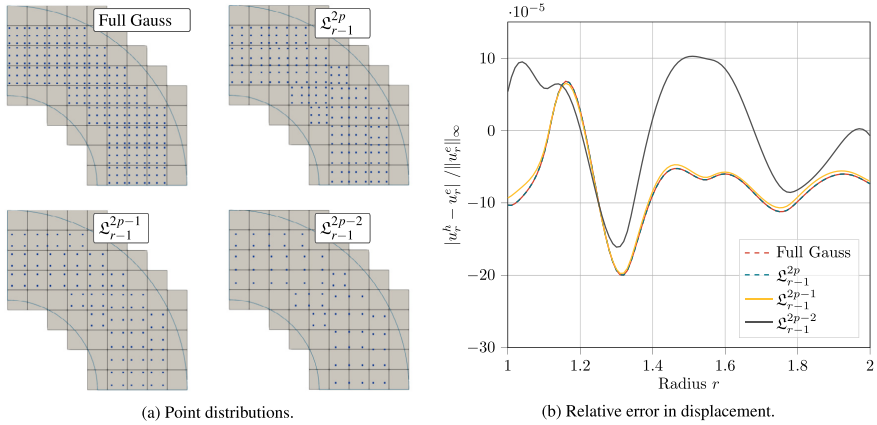


Fig. 19. Thick-walled cylinder: Exact and reduced integration for Mesh 4 and quadratic basis functions.

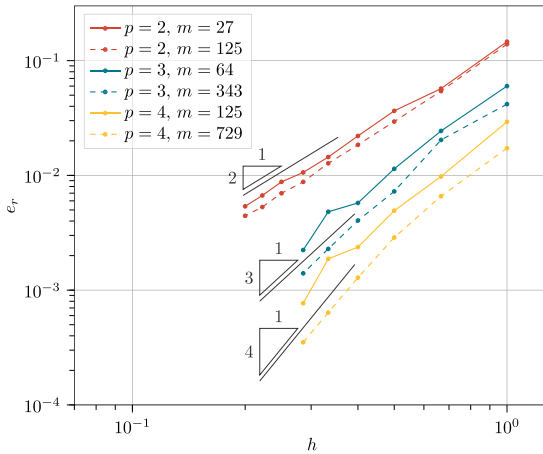


Fig. 20. Thick-walled cylinder: Relative error in energy norm e_r over knot span edge length h .

number of moment fitting bases provides a rough estimate of the required number of integration points, such that $n_q \approx m$, where m represents an upper bound, i.e., $n_q \leq m$. Thus, $m = (2p + 1)^3$ inherently yields a more complex quadrature rule, which in addition is more computationally intensive to construct. Moreover, the improvement in accuracy over the proposed approach seems irrelevant for practical applications. We conclude that the restriction to $m = (p + 1)^3$ basis functions is justifiable and therefore applied in all other simulations in this work.

6.3. Eigenfrequency analysis of an elastic cube

An elastic cube is investigated to demonstrate the potential of the presented method for efficient analyses of transient problems. First, we compare the performance of the exact and reduced quadrature rules presented in

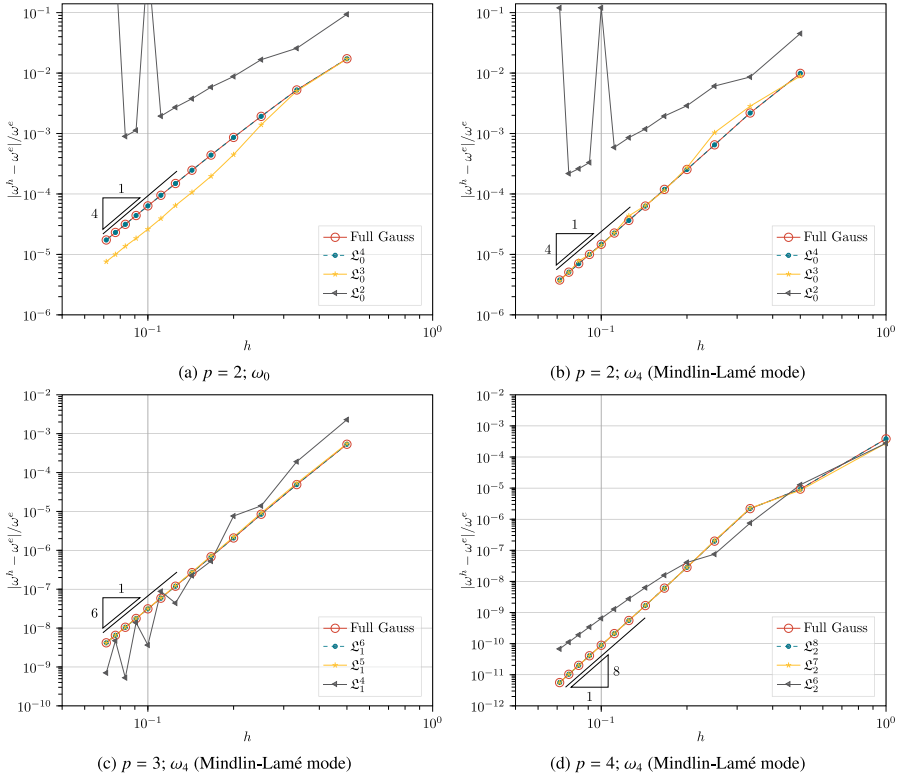


Fig. 21. Free Vibrating Cube: Relative error in eigenfrequency under global h-refinement with consistent mass matrix for quadratic, cubic, and quartic basis functions.

Section 3.1 based on the angular eigenfrequencies of the cube. Inspired by the results presented in [75], the second example is devoted to the influence of trimming on the critical time step Δt_{crit} of explicit dynamic simulations.

6.3.1. Free vibration of an unit cube

The natural frequencies of a traction-free unit cube are computed for $E = 100 \text{ N/m}^2$, $\nu = 0.3$, and $\rho = 1 \text{ kg/m}^3$ by solving the generalized eigenvalue problem

$$(\mathbf{K} - \omega^2 \mathbf{M})\Phi = 0, \tag{47}$$

where ω^2 is the vector of eigenvalues and Φ is the modal matrix. In fact, for most natural frequencies of a vibrating isotropic cube, no analytical solution is available. However, an exact solution exists for the Mindlin–Lamé modes [89], which are given as

$$\omega_i^e = \frac{\sqrt{2\pi} i}{L} \sqrt{\frac{G}{\rho}}, \tag{48}$$

where G is the shear modulus, L is the cube’s edge length, and i is a positive integer. In our example, the first Mindlin–Lamé mode corresponds to the fourth natural frequency of the free vibrating cube. A reference solution

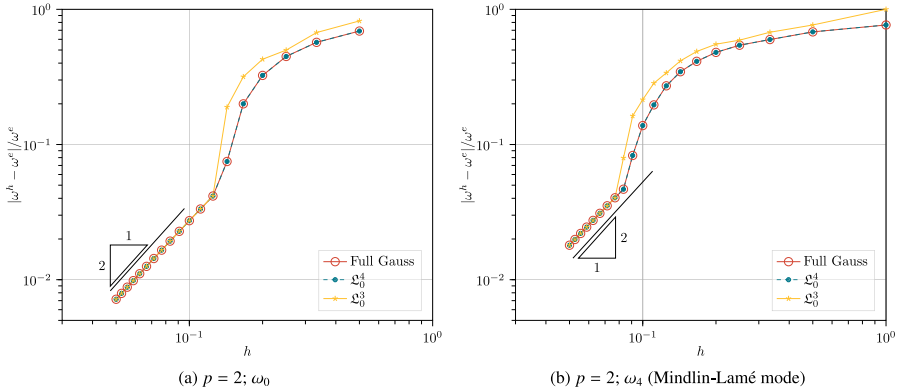


Fig. 22. Free Vibrating Cube: Relative error in eigenfrequency under global h-refinement with lumped mass matrix and quadratic basis functions.

is computed for $p = 4$, and a uniform mesh with 20 knot spans in each spatial direction for all remaining modes. The obtained angular eigenfrequencies are $\omega_1 = 17.712$ rad/s, $\omega_2 = 23.852$ rad/s, $\omega_3 = 24.257$ rad/s, $\omega_4 = 27.554$ rad/s, and $\omega_5 = 28.359$ rad/s, whereby ω_4 approximates the first Mindlin–Lamé eigenfrequency with a relative error of $\approx 10^{-13}$. Fig. 21 depicts the relative errors for $p = 2$, $p = 3$, and $p = 4$ over the knot spans edge length h . In all cases, exact and reduced quadrature rules based on \mathcal{Q}_{r-1}^{2p} , \mathcal{Q}_{r-1}^{2p-1} , and \mathcal{Q}_{r-1}^{2p-2} are compared to full Gaussian quadrature. The respective savings in the number of integration points are shown in Fig. 6. Note that the entire cube contains only full knot span domains. In agreement with the theoretical derivation in Section 3.1.1, \mathcal{Q}_{r-1}^{2p} provides exact quadrature rules and hence optimal convergence. Moreover, the error due to reduced integration associated with \mathcal{Q}_{r-1}^{2p-1} is clearly bounded by the discretization error. Considering the second-order reduced integration scheme corresponding to the target space of \mathcal{Q}_{r-1}^{2p-2} , the integration error becomes more dominant. Overall, the relative error still decreases when h-refinement is applied. However, optimal accuracy is not maintained in all cases. Additionally, nearly singular or negative-defined mass matrices are observed for quadratic basis functions, which are reflected in the peaks in Fig. 21(a)–(b).

For a second investigation, the consistent mass matrix \mathbf{M} is diagonalized using the row-summing technique. Fig. 22 proves that the use of a diagonal mass matrix \mathbf{M}_L limits the accuracy of the eigenfrequencies to second-order [40]. In Section 6.4.2, a predictor multi-corrector scheme is employed to circumvent this limitation. However, in the present example, we focus on the performance of the exact and reduced GGQ rules. Since the limitation imposed by the diagonal mass matrix is independent of the polynomial degree, the results are plotted exemplarily for quadratic basis functions in Fig. 22. Again, the first-order reduced integration rules do not significantly influence the overall accuracy. However, a further reduction of the target space to \mathcal{Q}_{r-1}^{2p-2} leads to quadrature rules that render the mass matrix singular, leading to unsolvable eigenvalue problems.

In summary, first-order reduced quadrature rules are shown to be robust for both the consistent \mathbf{M} and lumped mass matrix \mathbf{M}_L while maintaining full accuracy compared to exact integration. When using second-order reduced integration, occasional instabilities for \mathbf{M} and severe instability problems for \mathbf{M}_L are observed.

Remark. Besides row-summing, diagonal scaling and nodal quadrature methods are common techniques to obtain diagonal mass matrices. Note that the second-order accuracy restriction described above applies to all these lumping methods [90]. However, the diagonal mass matrices generally have a positive effect on the critical explicit time step Δt_{crit} , since they underestimate the high eigenfrequencies. The authors in [91] compare the discrete spectrum obtained using row-summing and diagonal scaling to the consistent mass matrix and show that both lumping schemes lead to a similar increase of Δt_{crit} . To improve the spatial accuracy, higher-order lumping techniques are developed in [92], which yield promising results for vibration analysis but produce non-diagonal mass matrices.

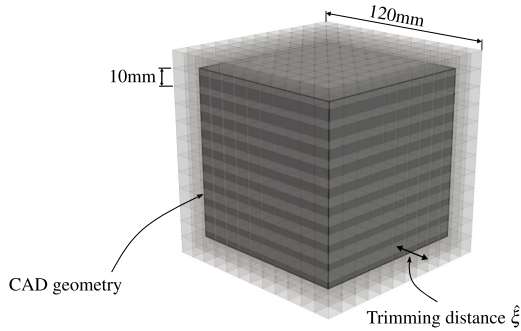


Fig. 23. Trivariate B-Spline cube with embedded CAD geometry. The cube is trimmed symmetrically on all sides according to the trimming distance $\xî$.

6.3.2. Free vibration of a trimmed cube

Small trimmed elements do not only affect the numerical stability, as discussed in Section 4 but may also hinder the use of explicit time integration schemes due to resulting infeasible critical time steps. The central difference scheme, widely used in explicit dynamics, is stable if the time step is smaller than the time it takes for a shock wave to pass through an element [93]. Several guidelines for estimating the critical time step are derived from this simple idea, all based on determining a characteristic element length. In the standard FEM, the nodes are either located on the boundary or inside the element. Therefore, it seems reasonable to establish a direct relation between characteristic length and the physical extent of the element. Consequently, the smaller the element, the smaller the critical time step. Generally, this is a meaningful analogy for simulation methods based on a C^0 continuous discretization field. As a result, trimmed elements, which can be arbitrarily small in practical applications, lead to infeasible simulation times in most embedded boundary methods. Nonetheless, this section demonstrates that the proposed method achieves practically feasible critical time steps despite arbitrarily small trimmed knot spans. Here, the continuity of the B-Spline bases is the crucial property [75]. To illustrate this phenomenon and to demonstrate the method’s potential within an explicit dynamic setting, Eq. (47) is solved for a trimmed cube. Fig. 23 shows the corresponding B-Spline domain as well as the embedded CAD geometry. A uniform open knot vector spans each spatial direction with 12 knot spans. Since explicit dynamic solvers classically do not invert the consistent mass matrix M but decouple the system of equations by a diagonal mass matrix M_L , the angular eigenfrequencies are computed with respect to M_L accordingly. Eq. (47) is solved for different trimming configurations, where the B-Spline domain is fixed and solely the size of the embedded cube changes. The model is kept symmetric, such that the trimming distance $\xî$ is equivalent on all sides of the cube, as depicted in Fig. 23. For all simulations, $E = 100 \text{ N/mm}^2$, $\nu = 0.25$, and $\rho = 30 \text{ kg/mm}^3$ define the linear elastic material.

In a first study, all inner knots are repeated $k = p$ times to create a C^0 continuous discretization field. Fig. 24 plots the maximum angular eigenfrequency ω_{max} over the trimming distance $\xî$ for different polynomial degrees of the underlying B-Spline basis functions. The dashed line represents a reference solution retrieved from a standard FE model using a structured hexahedral mesh and a constant element edge length of 10 mm. When linear basis functions are employed, and no trimming operations are applied ($\xî = 0 \text{ mm}$), the obtained maximum angular eigenfrequency is identical to the FE reference solution. In fact, the same values are repeatedly encountered for $\xî = 10 \text{ mm}$, $\xî = 20 \text{ mm}$, $\xî = 30 \text{ mm}$, and $\xî = 40 \text{ mm}$, as the size of the interior part of the knot spans is constant throughout the entire domain. However, if knot spans are cut at intermediate positions, the angular eigenfrequency rises and tends towards infinity when $\xî$ approaches the knot span boundaries. According to the well-known relation: $\Delta t_{crit} = 2/\omega_{max}$, the critical time step Δt_{crit} drops to infeasible values in these scenarios. Note that an increased polynomial degree reduces Δt_{crit} even further, see Fig. 24. The same undesirable behavior is expected for any trimmed C^0 continuous discretization field.

In contrast to classical methods, the here employed basis functions allow to elevate the continuity across adjacent knot spans by limiting the inner knot multiplicity to $k̄ = 1$. Fig. 25 plots the corresponding graphs for the

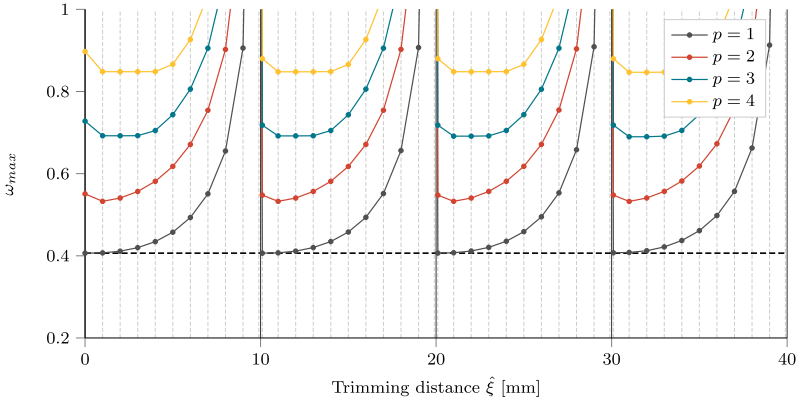


Fig. 24. Maximum angular eigenfrequency ω_{max} over the trimming distance $\hat{\xi}$ for a trivariate B-Spline cube (see Fig. 23) with uniform open knot vectors, C^0 continuity (inner knot multiplicity: $\bar{k} = p$), and varying polynomial degrees. The vertical lines at 10 mm, 20 mm, 30 mm, and 40 mm represent the knot span boundaries. The dashed horizontal line indicates a reference value obtained from a standard FE model with a structured hexahedral mesh and a constant element edge length of 10 mm.

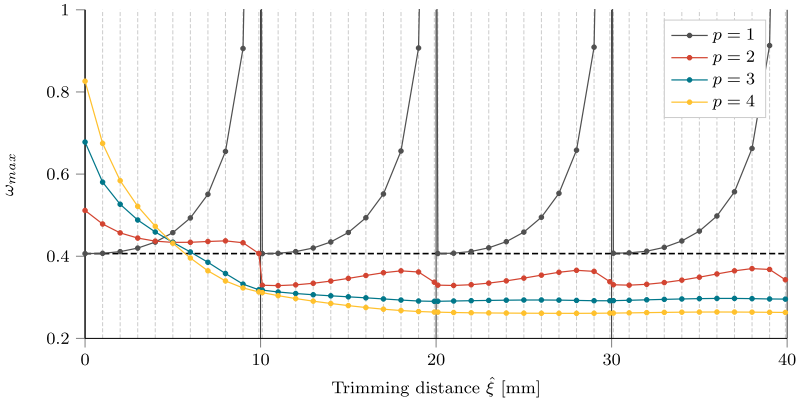


Fig. 25. Maximum angular eigenfrequency ω_{max} over the trimming distance $\hat{\xi}$ for a trivariate B-Spline cube (see Fig. 23) with uniform open knot vectors, C^{p-1} continuity (inner knot multiplicity $\bar{k} = 1$), and varying polynomial degrees. The vertical lines at 10 mm, 20 mm, 30 mm, and 40 mm represent the knot span boundaries. The dashed horizontal line indicates a reference value obtained from a standard FE model with a structured hexahedral mesh and a constant element edge length of 10 mm.

resulting C^{p-1} continuous discretization. Note that open knot vectors are applied in each spatial direction. Since the linear basis functions are still C^0 continuous, the same results as in Fig. 24 are obtained. However, the maximum eigenfrequency for $p > 1$, starts now at a higher level for cut boundary knot spans but converges towards values below the FE reference solution for cut intermediate knot spans. This trend becomes even more pronounced with increasing order of p . Regardless of the polynomial degree, all computed eigenfrequencies are lower than the

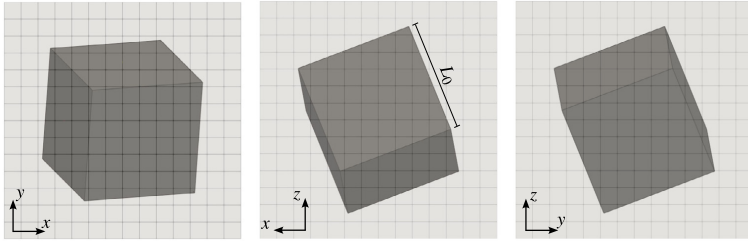


Fig. 26. Trivariate B-Spline cube with embedded CAD-geometry. Embedded cube is rotated by 30° around the x and y axes. L_0 denotes the initial edge length.

reference value for $\hat{\xi} > 10$ mm, representing the transition point from boundary to intermediate knot span. This observation is crucial since it shows that higher continuities can remove the adverse effect of trimming on the critical time step. However, trimming may only be applied to intermediate knot spans to take advantage of this.

These results suggest that the characteristic length does not correlate with the actual size of a knot span but with the zone spanned by the active control points. For intermediate knot spans, the zone grows with increasing polynomial degree and hence reduces the maximum angular eigenfrequency. Moreover, this interpretation also explains the contrary behavior within the boundary knot spans, as those are characterized by a higher control point density and steeper basis functions. Consequently, the corresponding stiffness entries are greater and mass values are smaller, which are both contributions to higher eigenfrequencies. For a more detailed discussion on the effect of boundary knot spans in this context, the interested reader is referred to [75,91].

In a second study, the embedded cube is rotated by 30° around the x and y axes to demonstrate that the above results are reproducible in more complex trimming scenarios. The corresponding setup is depicted in Fig. 26, where $L_0 = 140$ mm defines the initial edge length of the cube. We again employ an open knot vector with inner knot multiplicity of $k = 1$. Note that the cube is embedded such that all boundary knot spans are trimmed off. For the following investigation, the cube’s edge length is parameterized by $L = L_0 - \hat{L}$. Fig. 27 shows the obtained maximum angular eigenfrequency ω_{max} over the reduced length \hat{L} . The results highlight the beneficial effect of the higher continuities on the critical time step Δt_{crit} . While the results associated with the linear basis functions (C^0 continuous) are heavily affected by the different trimming scenarios, the curves for $p > 1$ are practically independent of them.

In conclusion, feasible critical time steps in an explicit dynamic setting can be realized if two crucial conditions are met:

1. The continuity across adjacent knot spans satisfies $C^{r>0}$.
2. If open knot vectors are used, the effect of the boundary knot spans must be eliminated.

The first condition is inherently fulfilled if C^{p-1} continuity in conjunction with quadratic or higher-order basis functions is employed. For the second requirement, the B-Spline domain is chosen large enough such that all boundary knot spans are outside the physical domain and hence trimmed off during the embedding process. Alternatively, non-open knot vectors can be employed, eliminating the adverse boundary effect due to equal and periodic basis functions over the entire patch. If these guidelines are followed, Figs. 25 and 27 indicate that the critical time step for $p = 3$ and $p = 4$ is practically independent of $\hat{\xi}$, and for $p = 2$ only slightly affected by the trimming operations. Moreover, in all cases, the B-Spline bases allow higher critical time steps than the FE reference model with the same number of elements. These results match the observation in [75], where a similar study in the scope of shell element-based IGA is performed. In [94], critical time step estimations for explicit IGA are studied and discussed.

We conclude that trimming in an explicit dynamic setting is only feasible with higher-order basis functions and higher continuity as naturally provided in IGA. This is a limitation for most embedded boundary methods,

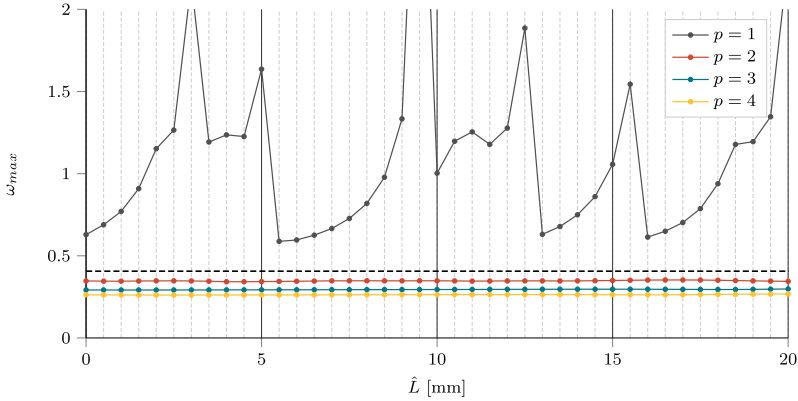


Fig. 27. Maximum angular eigenfrequency ω_{max} of a rotated cube over the reduced length \hat{L} with $L = L_0 - \hat{L}$ (see Fig. 26) for different polynomial degrees. A trivariate B-Spline discretization with uniform open knot vectors and C^{p-1} continuity (inner knot multiplicity $\bar{k} = 1$) is employed. The dashed horizontal line indicates a reference value obtained from a standard FE model with a structured hexahedral mesh and a constant element edge length of 10 mm.

which may fulfill the former requirement, but usually do not provide the necessary continuity due to the use of C^0 discretization fields.

6.4. Dynamic analysis of an elastic rod

In Section 6.3.1, generalized Gaussian quadrature rules are assessed based on the eigenfrequencies of a unit cube. The simple shape of the cube allows the construction of optimal and reduced quadrature rules from one single tensor product domain. In the following, the domain decomposition algorithm presented in Section 3.1.2 is applied to construct efficient integration rules for non-tensor product domains. The performance of different quadrature schemes associated with the target spaces \mathcal{Q}_{r-1}^{2p} , \mathcal{Q}_{r-1}^{2p-1} , and \mathcal{Q}_{r-1}^{2p-2} are compared using the eigenfrequencies of an elastic rod with different cross-sections. Moreover, a transient analysis of the given structure is performed.

6.4.1. Eigenfrequencies of an elastic rod with varying cross-sections

Fig. 28 shows three different cross-sections that are each extruded to form a rod of length $L = 10$ m. The colors indicate the tensor product domains found by the decomposition algorithm presented in Section 3.1.2. Note that the cross-sections are chosen to avoid any trimmed knot spans. The elastic material is defined by $E = 100$ N/m², $\nu = 0.0$, and $\rho = 1$ kg/m³. In all performed simulations, one end of the rod is fixed, while the rest of the structure is free to move in longitudinal direction. The exact natural frequencies of such fixed-free rod are given as

$$\omega_i^e = \frac{(2i - 1)\pi}{2L} \sqrt{\frac{E}{\rho}}, \tag{49}$$

where i is a positive integer [95]. In the first study, the eigenvalue problem (Eq. (47)) is solved for the *Square* cross-section (see Fig. 28) using the consistent mass matrix M . Figs. 29 (a)–(c) show the convergence of the first eigenfrequency over the knot span length h^x in longitudinal direction for $p = 2$, $p = 3$, and $p = 4$. The knot spans in cross-section remain unchanged. If exact integration schemes are applied, the rates of convergence are $O(h^{2p})$, which are optimal according to [40]. Moreover, we observe that the quadrature rules associated with \mathcal{Q}_{r-1}^{2p-1} and \mathcal{Q}_{r-1}^{2p-2} do not lower the accuracy, suggesting that the error due to reduced integration is bounded by the discretization error.

Eq. (49) shows that the natural frequencies of an elastic rod are neither affected by the shape nor the size of its cross-section. Therefore, the eigenfrequencies obtained from structures with varying cross-sections can directly be

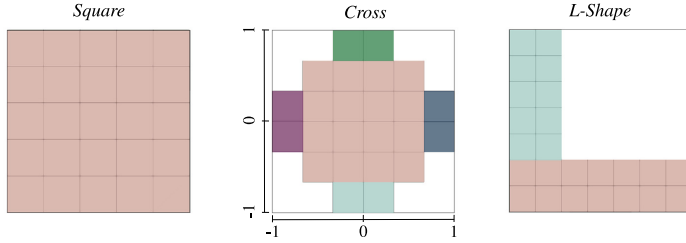


Fig. 28. Cross-sections of an elastic rod. Colors indicate different tensor product domains. Dimensions are given in [m]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

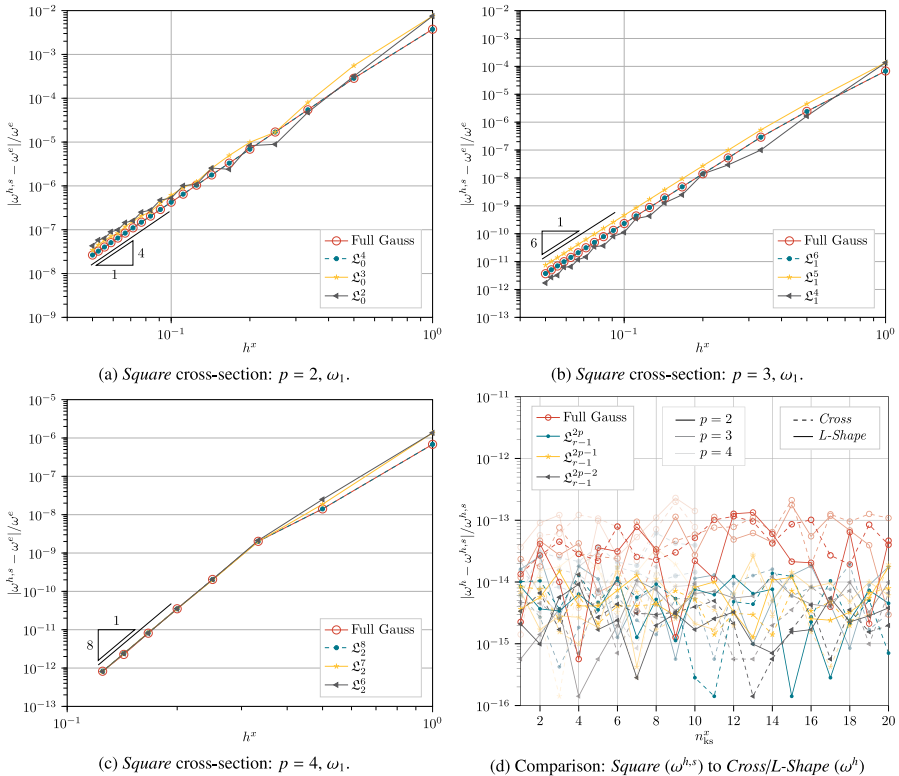


Fig. 29. Free-fixed elastic rod with *Square* cross-section: (a)–(c) Relative error in eigenfrequency with consistent mass matrix and (d) comparison to other cross-sections.

compared. Fig. 29(d) plots the relative discrepancy between the *Cross* and *Square* cross-sections, and the *L-Shape* and *Square* cross-sections, respectively. Regardless of the polynomial degree, the applied integration scheme, or the number of knot spans n_{ks}^x in x -direction, the decomposition of the *Cross* and *L-Shape* domains into tensor product

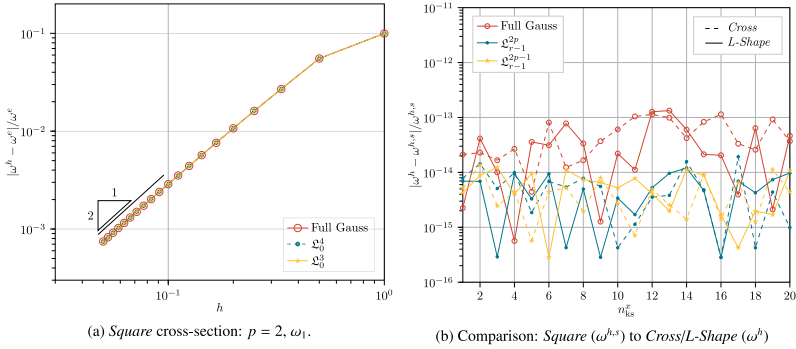


Fig. 30. Free-fixed elastic rod with *Square* cross-section. (a) Relative error in eigenfrequency with lumped mass matrix and (b) comparison to other cross-sections.

domains does not affect the accuracy of the quadrature rules. Note that all values in Fig. 29(d) are less than 10^{-12} , illustrating that the graphs depicted in Figs. 29 (a)–(c) would be the same for all cross-sections examined.

Fig. 30 shows the results obtained when the consistent mass matrix is approximated by a lumped mass matrix M_L using the row summing method. As already observed in Section 6.3, applying a second order-reduced integration scheme based on Ω_{r-1}^{2p-2} does not preserve rank sufficiency and positive definiteness of the lumped mass matrix in all cases. Therefore, the corresponding results are not shown. However, quadrature rules derived from Ω_{r-1}^{2p} and Ω_{r-1}^{2p-1} are robust and provide the same accuracy in the first eigenfrequency as full Gaussian quadrature. This is observed for all cross-sections depicted in Fig. 28. Since the lumped mass matrix restricts the accuracy to second-order independent of the polynomial degree, Fig. 30 provides the results only for $p = 2$. An effective remedy to improve the accuracy of the lumped mass matrix for transient analyses is applied in the next section.

6.4.2. Forced vibration of a trimmed elastic rod

In this section, a transient analysis of a trimmed rod with circular cross-section is performed. We model the structure similar to the configuration of the beam depicted in Fig. 13, where $a = 4$ m, $b = 4$ m, $R = 1$ m, and $L = 10$ m. Due to the cylindrical shape, the computational domain contains full and trimmed knot spans. The number of knot spans discretizing the cross-section is fixed to 5×5 , while the number of knot spans along x is parameterized with n_{ks}^x . Note that in contrast to Fig. 13, the loading is not applied in transverse but in longitudinal direction. A surface load of $p_x = p_0 \sin(\Omega t)$ with $\Omega = 4$ rad/s and $p_0 = 1$ N/m² acts at the right end at $x = 10$ m. Homogeneous Dirichlet conditions are enforced at $x = 0$ m using a penalty factor of $\beta = 5 \times 10^4$ N/m³. Each trimmed knot span domain is parameterized by approximately 1000 boundary triangles to evaluate Eq. (36). Fig. 31 shows the axial tip displacement obtained with different time integration schemes over time t . The implicit Newmark scheme with $\beta_N = 0.25$ and $\gamma_N = 0.5$ and the explicit central difference scheme are compared to a reference response retrieved by the superposition of the first 10 modes from the analytical solution [95]. Fig. 31 (a) ($n_{ks}^x = 10$) reveals that an implicit time integration in conjunction with the consistent mass matrix M achieves accurate results that match the reference solution. The maximum relative error over time is 1.2%. However, the response obtained with an explicit scheme shows a clear discrepancy. This gap stems from the lower accuracy of the employed lumped mass matrix, which has been already observed and discussed in Section 6.4.1. Another simulation with implicit time integration and lumped mass matrix M_L supports this claim, since it produces identical results as the explicit scheme. If h-refinement is performed, all simulation results converge to the analytical solution (see Fig. 31(b)), where the number of knot spans in longitudinal direction is increased to $n_{ks}^x = 50$. In both examples, the implicit time step is fixed to $\Delta t = 0.02$ s. For the explicit analysis, a maximum angular eigenfrequency of $\omega_{max} = 297.93$ rad/s, and $\omega_{max} = 615.05$ rad/s prescribes a critical time step of $\Delta t_{crit} = 6.71 \times 10^{-3}$ s for $n_{ks}^x = 10$, and $\Delta t_{crit} = 3.25 \times 10^{-3}$ s for $n_{ks}^x = 50$. To provide a small buffer the explicit time steps are defined as $\Delta t = 6.5 \times 10^{-3}$ s and $\Delta t = 3 \times 10^{-3}$ s, respectively.

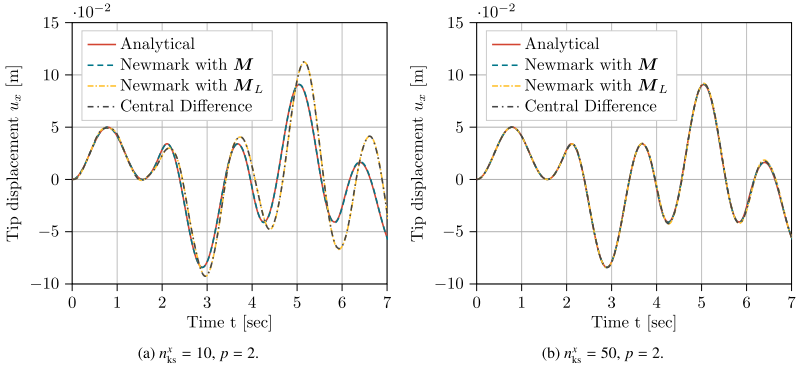


Fig. 31. Forced vibrating free-fixed elastic trimmed rod with circular cross-section: Implicit and explicit simulation with quadratic basis function.

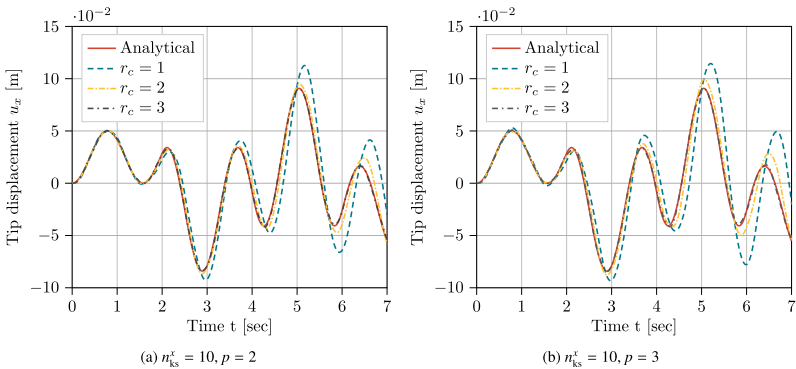


Fig. 32. Forced vibrating free-fixed elastic trimmed rod with circular cross-section: Predictor multi-corrector scheme with (a) quadratic and (b) cubic basis functions.

In a second example, a modified integration scheme addresses the lower accuracy of the lumped mass matrix for higher-order basis functions [40]. The authors in [39,41,96] propose an explicit predictor multi-corrector algorithm that still relies on a decoupling of the system of equations through a diagonal mass matrix but additionally introduces the consistent mass matrix for the computation of the residual vector. According to [41], the explicit algorithm can behave like a classical Newmark method with the consistent mass matrix if a sufficiently large number of corrector passes r_c are conducted. However, already for moderate values of r_c , significant improvements are observed. To demonstrate this, the predictor multi-corrector scheme is applied to the vibrating elastic rod with $n_{ks}^x = 10$ studied above. Our implementation of the explicit scheme follows the algorithmic structure in [41]. Fig. 32 shows the effect of $r_c = 1$, $r_c = 2$, and $r_c = 3$ corrector passes for quadratic and cubic basis functions. If $r_c = 1$, the algorithm breaks down to a forward difference scheme and consequently provides no distinct advantage over the central difference scheme. However, for $r_c \geq 2$ a significant improvement is observed in Fig. 32. Note that the algorithm’s stability is not adversely affected compared to, e.g., a forward difference scheme ($r_c = 1$) since the lumped mass matrix is still inverted and the consistent mass matrix is solely used to enrich the computation of the residual vector [39]. In fact, the eigenvalue problem using the consistent mass matrix for $p = 2$ and $p = 3$ yields the following maximum

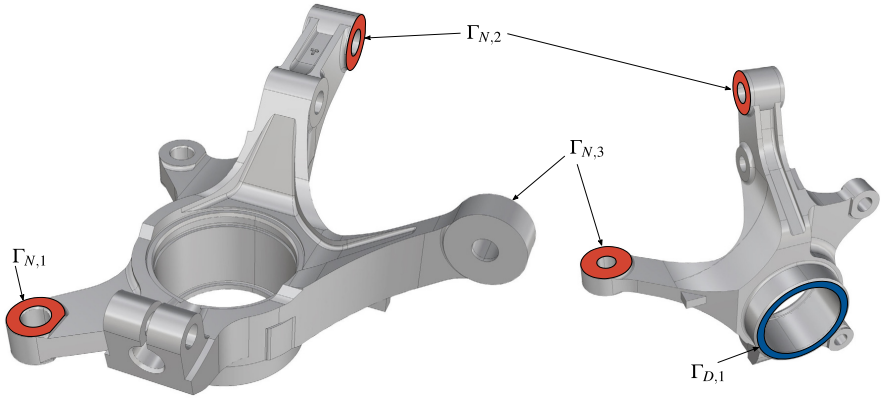


Fig. 33. Model configuration of steering knuckle with Neumann (red) and Dirichlet (blue) boundary conditions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

eigenfrequencies: $\omega_{max} = 1384.07$ rad/s and $\omega_{max} = 1520.83$ rad/s, which would result in $\Delta t_{crit} = 1.445 \times 10^{-3}$ s and $\Delta t_{crit} = 1.315 \times 10^{-3}$ s. However, the actual critical time steps are governed by the eigenvalue problem employing the lumped mass matrix. For $p = 2$ and $p = 3$, the corresponding values are $\omega_{max} = 297.93$ rad/s and $\omega_{max} = 280.11$ rad/s, and $\Delta t_{crit} = 6.71 \times 10^{-3}$ s and $\Delta t_{crit} = 7.14 \times 10^{-3}$ s. Thus, the finally used time steps are again $\Delta t = 6.5 \times 10^{-3}$ s for $p = 2$, and $\Delta t = 7 \times 10^{-3}$ s for $p = 3$.

In conclusion, the predictor multi-corrector scheme is shown to drastically improve the spatial accuracy for higher-order bases while preserving the computational architecture of explicit algorithms. The associated additional cost is directly proportional to the conducted corrector passes r_c . In agreement with [41], the obtained results indicate that 2–3 corrector passes are sufficient for most practical applications. Generally, such a moderate number of corrector passes pays off since the classical central difference scheme requires much finer meshes to attain the same accuracy.

7. Industrial example

The effectiveness of the proposed methodology in analyzing real-world applications will be demonstrated in the following. To this end, we investigate a steering knuckle¹ with complex geometry and detailed features. The model configuration with Neumann and Dirichlet boundary conditions is depicted in Fig. 33. The surface loads of $p_1 = 2$ N/mm², $p_2 = 1$ N/mm², and $p_3 = 0.3$ N/mm² act in inward-pointing normal direction on the Neumann boundaries $\Gamma_{N,1}$, $\Gamma_{N,2}$, and $\Gamma_{N,3}$. Homogeneous Dirichlet conditions are enforced on $\Gamma_{D,1}$ with a penalty factor of $\beta = 10^{10}$ N/mm³. Approximately 1000 boundary triangles parameterize each trimmed knot span domain in order to evaluate the constant terms of the moment fitting equation (see Eq. (36)). The point distribution factor is initialized with $\gamma = 2$. For all simulations performed, $E = 2.1 \times 10^5$ N/mm² and $\nu = 0.3$ define the linear elastic material. To assess the performance of the proposed method, the relative error in total strain energy is computed according to

$$\bar{\epsilon}_r = \sqrt{\frac{|U^h - U^{ref}|}{U^{ref}}}, \tag{50}$$

where U^{ref} is the total strain energy of a FE reference model with 3 million linear tetrahedral elements and U^h is the total strain energy associated with the trimmed B-Spline domain. Fig. 34(a) plots the relative error $\bar{\epsilon}_r$ for different quadratic B-Spline discretizations produced by h-refinement. Furthermore, the exact and reduced quadrature rules related to the target spaces \mathfrak{L}_{r-1}^{2p} , $\mathfrak{L}_{r-1}^{2p-1}$, and $\mathfrak{L}_{r-1}^{2p-2}$ (see Section 3.1.1) are applied to all full knot spans, and their

¹ CAD model of steering knuckle is taken from: <https://grabcad.com/library/steering-knuckle-rh-1>. Designer: Rushikesh Kulkarni.

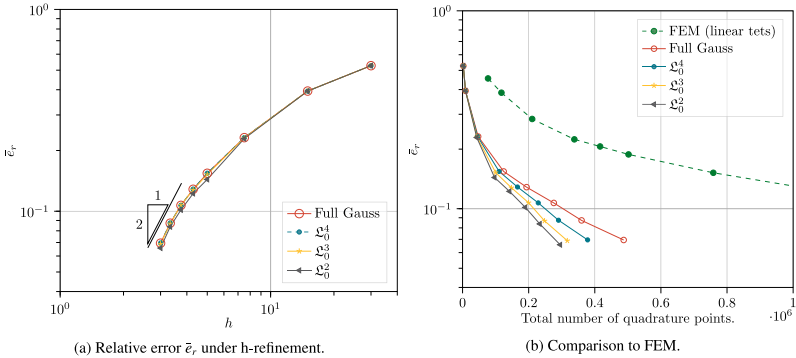


Fig. 34. Steering Knuckle: Relative error in internal energy $\bar{\epsilon}_r$ (a) over knot span edge length h and (b) number of integration points. Comparison between exact and reduced quadrature rules for full knot spans. Quadratic basis functions are employed.

performances are compared. The decomposition algorithm presented in Section 3.1.2 is employed to construct the necessary tensor product domains. Note that all computational models are created automatically and without manual interventions, such as defeaturing the CAD geometry etc. Fig. 34(a) reveals that second-order accuracy is achieved regardless of the quadrature rule used. For the coarsest mesh with $h = 30$ mm, the results are identical since all active knot spans are trimmed. However, as the mesh is refined, the ratio between full and trimmed knot spans increases steadily. The finest discretization ($h = 3$ mm) depicted in Fig. 35 contains 7847 full and 10 243 trimmed knot spans. In this example, exact integration (Ω_0^4) reduces the average number of integration points per full knot span from $n_q = 27$ to $n_q = 13.1$. Using first- and second-order reduced quadrature rules associated with the target spaces Ω_0^3 and Ω_0^2 further reduces these numbers to $n_q = 5.3$ and $n_q = 2.5$, respectively. Since the moment fitting equation is assembled with only $m = (p + 1)^3$ moment fitting bases (see Section 3.2.1), the point elimination algorithm presented in Section 3.2.3 converges to approximately $n_q = 27$ integration points per trimmed knot span. For some trimmed domains, the predefined moment fitting residual of $\delta = 10^{-10}$ is even achieved with $n_q < 27$. Thus, the average number of quadrature points per trimmed knot span aggregates to $n_q = 26.9$. In conjunction with a second-order reduced integration scheme for all full knot spans, the global number of quadrature points per knot span decreases from $n_q = 27$ to $n_q = 16.3$ while attaining the same degree of accuracy. Note that the saving in quadrature points will be even more pronounced when the mesh is finer or, in general, for geometries where the volume to surface ratio is larger.

In Fig. 34(b), the accuracy in internal energy with respect to the number of quadrature points is compared to a classical FE model. The associated tetrahedral meshes are also automatically generated without defeaturing the detailed CAD geometry for a fair comparison. Due to the quadratic basis functions used for the trimmed B-Spline solid, the internal energy corresponding to the FE model is expected to converge slower regarding the number of elements. In fact, to achieve similar accuracy as the finest B-Spline mesh with 18 090 knot spans, the FE model requires approximately 2 million elements. However, since higher-order elements, and especially trimmed higher-order elements, usually require more complex quadrature rules, predicting the accuracy with respect to the number of integration points is less obvious. Note that the number of integration points in the FE model is equivalent to the number of elements, since a single Gauss point exactly evaluates linear tetrahedral elements. Fig. 34(b) shows that the FE model requires significantly more quadrature points than the proposed approach to achieve the same accuracy in internal energy. Even if knot span-wise Gaussian quadrature is applied to all full knot spans within the B-Spline solid, the difference is remarkable. In comparison to the finest B-Spline discretization ($h = 3$ mm), the FE model requires around 4 times more integration points. This discrepancy can be further increased by using exact and reduced generalized Gaussian quadrature rules. When second-order reduced quadrature schemes are employed, the FE model requires approximately 7 times more integration points. Fig. 36 shows the deformed CAD geometry and compares it to the FE reference model with 3 million elements. For the finest discretization studied, the relative error of the maximum displacement is $< 1\%$ with 18 090 active knot spans.

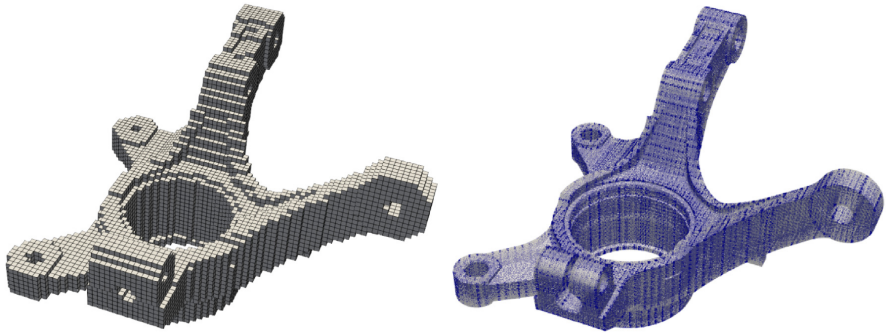


Fig. 35. Steering Knuckle: Active knot spans and integration points for finest discretization studied with $h = 3$ mm.

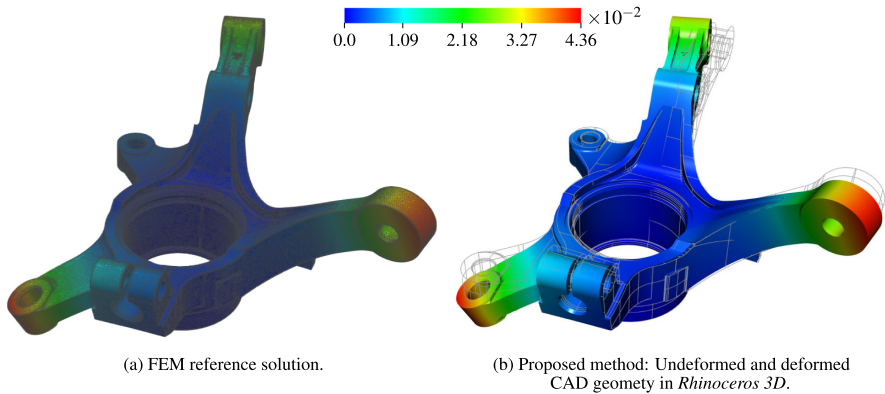


Fig. 36. Steering Knuckle: (a) comparison FE analysis to (b) proposed method. Deformed and undeformed CAD geometry with color mapping in *Rhinoceros 3D*. Displacements are given in [mm]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The results suggest that the proposed approach requires fewer elements and quadrature points than traditional low-order finite element methods to achieve the same degree of accuracy. However, the higher polynomial degree and continuity of the B-Spline bases entail more computationally intensive operations when assembling and solving the system of equations. Note that this is not a peculiarity of the present approach but a common side effect of isogeometric methods. On the one hand, the cost to build the system matrices increases for higher polynomial degrees. On the other hand, the efficiency of direct and iterative solvers may be affected by the higher continuities of the basis functions [97,98]. Generally, the computational overhead of both assembly and solution per degree of freedom increases drastically for higher polynomial degrees. However, since we use only low to moderate values of p , the associated additional cost is still modest. Moreover, compared to the linear FE model, the same accuracy is achieved with 12-times fewer degrees of freedom, which counteracts the extra effort mentioned. Based on the work presented in [99], a more detailed estimation of the actual cost of the simulation is presented in Appendix C. We compare the most refined B-Spline discretization to the FE model that achieves similar accuracy. It is shown that despite the presence of ill-conditioned system matrices due to trimming, the overall cost for this particular

example can be estimated to be comparable to the FE simulation. Moreover, an outlook on explicit solvers is given in Appendix C.3, demonstrating that the proposed approach allows for a drastically larger critical time step. Compared to linear FEM, Δt_{crit} is shown to increase by a factor of 45.

8. Conclusion and outlook

This publication presents a complete workflow for direct analysis of solid B-Rep models from the initial design in CAD to the visualization of simulation results in CAD. A key feature is that the geometric boundary is defined as the trimming surface of a uniform trivariate B-Spline cuboid. The newly developed method features highly efficient quadrature rules that drastically reduce the number of integration points compared to classical embedded boundary methods. As a result, our approach is characterized by fast matrix formations for static and implicit dynamic simulations. Moreover, the proposed method achieves practically feasible critical explicit time steps despite arbitrarily small trimmed knot spans/elements, which traditional embedded techniques on C^0 continuous domains cannot guarantee. Finally, the developed quadrature rules can be applied in explicit dynamic simulations, where function evaluations at each integration point are the predominant cost. The key building blocks tailored to achieve these objectives are summarized below.

- Efficient and robust integration of
 - trimmed knot spans:

For each trimmed knot span domain, we solved the linearized moment fitting equation for the weights of the integration points and optimized their locations during the execution of a point elimination algorithm. A new solution strategy was developed to set an upper bound on the final number of quadrature points. The proposed implementation resorts to a non-negative least squares solver, which inherently ensures positive defined weights and also drastically reduces the number of required elimination loops. Regardless of the polynomial degree, the algorithm converged to $n_q = (p+1)^3$ quadrature points per trimmed domain after the first iteration. Depending on the prescribed maximum residual of the least squares problem, further iterations may be performed, leading to a final set of points with $n_q < (p+1)^3$.
 - full knot spans:

To leverage the higher continuity of the B-Spline bases, generalized Gaussian quadrature schemes were employed, which are superior to knot span-wise Gaussian quadrature. Exact and reduced integration schemes were constructed from precomputed quadrature rules associated with the B-Spline target spaces \mathcal{Q}_{r-1}^{2p} , and \mathcal{Q}_{r-1}^{2p-1} . We developed a novel decomposition algorithm to enable their application to non-tensor product domains. In all simulations performed, the reduced integration associated with the target space \mathcal{Q}_{r-1}^{2p-1} maintained the accuracy of full Gaussian quadrature while achieving another significant efficiency gain compared to the exact rules corresponding to \mathcal{Q}_{r-1}^{2p} . Moreover, we continued the discussion started in [49] and studied second-order reduced quadrature schemes derived from \mathcal{Q}_{r-1}^{2p-2} . The corresponding error arising from reduced integration was bounded by the discretization error for linear static examples, as predicted in [49]. Moreover, optimal convergence rates were attained for the relative error in the first natural frequency of an elastic rod but were not maintained for an elastic cube. In addition, rank insufficient mass matrices were observed in some cases. When applying mass lumping in conjunction with second-order reduced quadrature schemes, negative defined or singular matrices lead to infeasible results in most examples. In contrast, quadrature rules associated with \mathcal{Q}_{r-1}^{2p-1} were stable even when mass lumping was applied and provided the same accuracy as exact integration.

To demonstrate the potential of the proposed method, quadratic B-Splines were applied to a detailed and complex industrial example, where $n_q = 27$ quadrature points per trimmed knot span and $n_q = 2.5$ per full knot span were sufficient to achieve optimal convergence in the energy norm.

- Practically feasible critical explicit time step:

We showed that the critical explicit time step becomes independent of the trimming operations when non-open knot vectors in conjunction with C^1 continuous (or higher) basis functions are employed. If open knot vectors are used, trimming must be applied exclusively at intermediate knot spans in order to allow efficient explicit dynamic simulations.

- Increased spatial accuracy of the lumped mass matrix for higher-order bases:
Even if an exact integration scheme is applied, it is known that mass lumping (e.g., by row summation) limits the accuracy of the calculated natural frequencies to second-order, regardless of the polynomial degree. A predictor multi-corrector scheme was successfully adopted to trimmed B-Spline solids and has proven to be an effective remedy to improve the accuracy of explicit transient analyses significantly.

Future work will focus on local mesh refinement and the representation of discontinuities within smooth patches. The developed integration schemes combined with the presented practical approach to bound the critical time step to feasible values open the door for efficient explicit dynamic simulations. However, stabilization schemes for light-control points may need to be considered for large-scale applications, which will require further investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request. The source code related to this article can be found at <https://github.com/manuelmessmer/TIBRA> [80].

Acknowledgments

This work was supported by the DYNAmore GmbH, Germany and the BMW Group, Germany. Their support is gratefully acknowledged. We thank the reviewers for their input, which improved the quality of this manuscript.

Appendix A. Minimal boundary representation: STL

This section shall give a brief overview of the STL file format and its beneficial properties exploited in this work. The STL is a prominent data format for exchanging geometric information between CAD and analysis. An STL model is a minimal boundary representation, as it is limited to the essential information. Since several innovative processes such as rapid prototyping and 3D printing rely on STL models, their construction from classical NURBS-based B-Rep models is a standard feature of modern CAD programs. At the core of this process is a tessellation algorithm generating the boundary triangles. Besides the triangle's vertices, the STL also stores a normal vector, pointing in outward direction of the geometry.

The accuracy of the final geometry represented by the STL can be determined by a number of different parameters, such as the maximum aspect ratio and the maximum/minimum edge length of the triangles. However, unlike the classical FEM, the proposed method does not impose high requirements on the mesh quality. Since it is not used to discretize the field variables but solely serves as the delimitation of the integration domains, the aspect ratio does not affect the overall solution. Furthermore, a general upper bound for the edge length of the triangles does not necessarily need to be enforced either. In fact, the only goal regarding the quality of the boundary tessellation is to represent the original B-Rep model accurately. This objective can be achieved with a single parameter: the chordal tolerance or chordal deviation, which controls the maximum distance between the surface mesh and the exact geometric boundary. If the chordal tolerance is the only constraint on the tessellation algorithm, relatively flat or straight regions will be discretized with large triangles, and small elements will emerge at curved surfaces, sharp corners and edges. Thus, very efficient boundary representations with high accuracy are achieved. For example, a simple cube can be exactly represented with only twelve elements. At the same time, the cylinder depicted in Fig. 1 requires a higher density of elements at the curved boundaries. In any case, the entire process can be fully automated. Fig. A.37 plots the relative error in the volume over the number of respective boundary triangles for the cylinder just mentioned. The corresponding error is defined according to

$$e_v = \frac{|V^{STL} - V^e|}{V^e}, \quad (\text{A.1})$$

where V^{STL} is the volume computed based on the STL representation and V^e is the exact volume of the cylinder. In this example, a quadratic convergence rate is achieved, considering the relative error e_v and the number of triangles

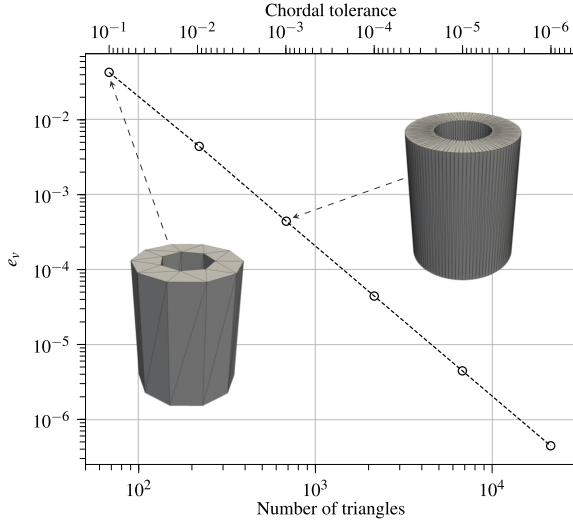


Fig. A.37. Relative error in volume over the number of triangles in the STL model.

within the STL mesh. This is not meant as a representative study with general applicability but rather to give an approximate idea of the feasible accuracy. Since a boundary mesh is sufficient, fewer elements can achieve higher accuracy compared to standard FEM, which inevitably requires a volumetric mesh. Moreover, an extremely fine boundary description may slow down required preprocessing operations but does not influence the actual simulation time due to the decoupling of the geometric description and the discretization of the field variables.

Appendix B. Second-order reduced quadrature rules for C^{p-1} continuous splines

Table B.3

Quadrature points corresponding to the target space Ω_0^2 .

n_{ks}	Position	Weight	n_{ks}	Position	Weight
1	0.211324865405187	0.5000000000000000	2	0.1666666666666666	0.3750000000000000
	0.788675134594812	0.5000000000000000		0.5000000000000000	0.2500000000000000
	-	-		0.8333333333333333	0.3750000000000000
3	0.1111111111111111	0.2500000000000000	4	0.0833333333333333	0.1875000000000000
	0.375774001250011	0.2500000000000000		0.3055555555555555	0.241071428571428
	0.62422598749988	0.2500000000000000		0.5000000000000000	0.142857142857142
	0.8888888888888888	0.2500000000000000		0.6944444444444444	0.241071428571428
	-	-		0.9166666666666666	0.1875000000000000
5	0.0666666666666666	0.1500000000000000	6	0.0555555555555555	0.1250000000000000
	0.2444444444444444	0.192857142857142		0.203703703703703	0.160714285714285
	0.424121308936067	0.157142857142857		0.368686868686868	0.166208791208791
	0.575878691063932	0.157142857142857		0.5000000000000000	0.096153846153845
	0.7555555555555555	0.192857142857142		0.631313131313131	0.166208791208791
	0.9333333333333333	0.1500000000000000		0.796296296296296	0.160714285714285
	-	-		0.9444444444444444	0.1250000000000000

(continued on next page)

Table B.3 (continued).

n_{ks}	Position	Weight	n_{ks}	Position	Weight
7	0.047619047619047	0.107142857142857	8	0.041666666666666	0.093750000000000
	0.174603174603174	0.137755102040816		0.152777777777777	0.120535714285713
	0.316017316017316	0.142464678178963		0.276515151515151	0.124656593406593
	0.445731404374050	0.112637362637362		0.401422764227642	0.124975218080888
	0.554268595625950	0.112637362637362		0.500000000000000	0.072164948453608
	0.683982683982683	0.142464678178963		0.598577235772357	0.124975218080888
	0.825396825396825	0.137755102040816		0.723484848484848	0.124656593406593
	0.952380952380952	0.107142857142857		0.847222222222222	0.120535714285713
	–	–		0.958333333333333	0.093750000000000
	–	–		–	–
9	0.037037037037037	0.083333333333333	10	0.033333333333333	0.075000000000000
	0.135802469135802	0.107142857142857		0.122222222222222	0.096428571428571
	0.245791245791245	0.110805860805860		0.221212121212121	0.099725274725274
	0.356820234869015	0.111089082738567		0.321138211382113	0.099980174464710
	0.457787206903364	0.087628865979381		0.421132897603485	0.09998576066526
	0.542212793096636	0.087628865979381		0.500000000000000	0.057734806629834
	0.643179765130984	0.111089082738567		0.578867102396514	0.09998576066526
	0.754208754208754	0.110805860805860		0.678861788617886	0.099980174464710
	0.864197530864197	0.107142857142857		0.778787878787878	0.099725274725274
	0.962962962962963	0.083333333333333		0.877777777777777	0.096428571428571
–	–	0.966666666666666	0.075000000000000		

In the following, the second-order reduced quadrature rules for uniform C^{p-1} continuous splines are provided. Tables B.3, B.4, B.5 list the respective integration points for the numerical integration of quadratic, cubic, and quartic basis functions. The positions and weights are computed with a relative error of $<10^{-15}$.

Table B.4

Quadrature points corresponding to the target space Ω_1^4 .

n_{ks}	Position	Weight	n_{ks}	Position	Weight
1	0.112701665379257	0.277777777777777	2	0.084001595740497	0.204166185672590
	0.500000000000000	0.444444444444444		0.353667436436311	0.295832814327409
	0.887298334620742	0.277777777777777		0.646332563563688	0.295833814327409
	–	–		0.915998404259502	0.204166185672590
3	0.055307959538964	0.134383670129083	4	0.042302270496914	0.102836135188702
	0.232008127012760	0.190719210529352		0.178540270746367	0.151209936088574
	0.410698113579587	0.174897119341563		0.335067537628327	0.165363166232140
	0.589301886420412	0.174897119341563		0.500000000000000	0.161181524981165
	0.767991872987239	0.190719210529352		0.664932462371672	0.165363166232140
	0.944692040461035	0.134383670129083		0.821459729253632	0.151209936088574
–	–	0.957697729503086	0.102836135188702		
5	0.033825647049692	0.082228488484279	6	0.028201513664609	0.068557423459134
	0.142739413107186	0.120781740225645		0.119026847164245	0.100806624059049
	0.267383546533899	0.131305988133937		0.223378358418885	0.110242110821427
	0.393434348254817	0.112350449822804		0.333469112711755	0.107894565719166
	0.500000000000000	0.106666666666667		0.444444801812292	0.112499275941222
	0.606565651745182	0.112350449822804		0.55555198187707	0.112499275941222
	0.732616453466100	0.131305988133937		0.666530887288244	0.107894565719166
	0.857260586892813	0.120781740225645		0.776621641581114	0.110242110821427
	0.966174352950307	0.082228488484279		0.880973152835755	0.100806624059049
	–	–		0.971798486335390	0.068557423459134
7	0.024172701827349	0.058763445397266	8	0.021151135248457	0.051418067594351
	0.102022872988446	0.086405398333432		0.089270135373183	0.07560496804287

(continued on next page)

Table B.4 (continued).

n_{ks}	Position	Weight	n_{ks}	Position	Weight
	0.191466170226540	0.0944911808034350		0.167533768814163	0.082681583116070
	0.285521376837719	0.091484077091862		0.250101839718338	0.080920941088586
	0.378383847551573	0.091689963236568		0.333333646037763	0.084374547320314
	0.461048573830697	0.077165307906519		0.416666711318236	0.084374909580740
	0.538951426169302	0.077165307906519		0.500000000000000	0.081249966511299
	0.621616152448426	0.091689963236568		0.583333288681763	0.084374909580740
	0.714478623162281	0.091484077091862		0.666666353962236	0.084374547320314
	0.808533829773459	0.0944911808034350		0.749898160281661	0.080920941088586
	0.897977127011553	0.08640539833432		0.832466231185836	0.082681583116070
	0.975827298172650	0.058763445397266		0.910729864626816	0.075604968044287
	–	–		0.978848864751543	0.051418067594351
9	0.018801009109739	0.045704948972756	10	0.016920908198765	0.041134454075480
	0.079351231442830	0.067204416039366		0.071416108298547	0.060483974435429
	0.148918905612590	0.073494740547618		0.134027015051331	0.066145266492856
	0.222306384934177	0.071909111681224		0.200081471774670	0.064736752870869
	0.296241798312773	0.074889092297216		0.266666916830210	0.067499637856251
	0.369983340610217	0.074427408306718		0.333333369054589	0.067499927664592
	0.440686894185870	0.062740652525469		0.400000004121687	0.064999986604520
	0.500000000000000	0.059259259259259		0.466666666666667	0.067499999999999
	0.559313105814129	0.062740652525469		0.533333333333332	0.067499999999999
	0.630016659389782	0.074427408306718		0.599999995878313	0.064999986604520
	0.703758201687226	0.074889092297216		0.66666630945410	0.067499927664592
	0.777693615068822	0.071909111681224		0.733333083169789	0.067499637856251
	0.851081094387409	0.073494740547618		0.799918528225329	0.064736752870869
	0.920648768557169	0.067204416039366		0.865972984948668	0.066145266492856
	0.981198990890260	0.045704948972756		0.928583891701453	0.060483974435429
	–	–		0.983079091801234	0.041134454075480

Table B.5

Quadrature points corresponding to the target space Ω_2^6 .

nk	Position	Weight	nk	Position	Weight
1	0.069431844202971	0.173927422568723	2	0.046212737218260	0.115024181444676
	0.330009478207568	0.326072577431276		0.213797850600020	0.203072613437833
	0.669990521792431	0.326072577431276		0.413962200649005	0.181903205117489
	0.930568155797028	0.173927422568723		0.586037799350994	0.181903205117489
	–	–		0.786202149399979	0.203072613437833
	–	–		0.953787262781739	0.115024181444676
3	0.032509212332345	0.080935503234345	4	0.024674350061534	0.061435702093655
	0.150617986025161	0.143497669038852		0.114388066516855	0.109125580973741
	0.294914900084714	0.137136269815659		0.225696248333799	0.109440065519939
	0.429193875888645	0.138430557911142		0.338034400421371	0.117618543966108
	0.570806124111354	0.138430557911142		0.451622850188173	0.102380107446554
	0.705085099915285	0.137136269815659		0.548377149811826	0.102380107446554
	0.849382013974838	0.143497669038852		0.661965599578628	0.117618543966108
	0.967490787667655	0.080935503234345		0.774303751666200	0.109440065519939
	–	–		0.885611933483144	0.109125580973741
	–	–		0.975325649938465	0.061435702093655
5	0.019796220904766	0.049291039978813	6	0.016580606164153	0.041105385951518
	0.091787579286592	0.087594474874909		0.076547082893084	0.073056382382853
	0.181541216020813	0.089204170945074		0.151495324087457	0.074702389409214
	0.274797052289731	0.098993104267740		0.229972811622965	0.083636771294890
	0.371943037927439	0.090194507546743		0.313018298389042	0.079318952819657
	0.456885245389880	0.084727202386718		0.391207050914918	0.079501094709966

(continued on next page)

Table B.5 (continued).

nk	Position	Weight	nk	Position	Weight
	0.543114754610120	0.084722702386718		0.467553762645136	0.068679023431878
	0.628056962072560	0.090194507546743		0.532446237354863	0.068679023431878
	0.725202947710268	0.098993104267740		0.608792949085081	0.079501094709966
	0.818458783979186	0.089204170945074		0.6686981701610957	0.079318952819677
	0.908212420713407	0.087594474874909		0.7700271883777034	0.083636771294890
	0.980203779095233	0.049291039978813		0.848504675912542	0.074702389409214
	–	–		0.923452917106915	0.073056382382853
	–	–		0.983491393835846	0.041105385951518
7	0.014152764059326	0.035239541651249	8	0.012384225491674	0.030835997499107
	0.065624147934996	0.062632886080504		0.057423850693207	0.054806665556838
	0.129898953088507	0.064110618987887		0.113671692274620	0.056114474224266
	0.197332988106867	0.071940415673692		0.172713533232678	0.063003575169824
	0.269035732180605	0.069122133946639		0.235572849776790	0.060747103491937
	0.338420711069781	0.071627956854866		0.296849601057879	0.063533327687617
	0.40833520242371	0.064746706239499		0.359592609050278	0.059762992256298
	0.469180175658879	0.060579740565660		0.418373312590155	0.059670136789748
	0.530819824341120	0.060579740565660		0.475657800642327	0.051525272324360
	0.591666479757629	0.064746706239499		0.524342199357672	0.051525272324360
	0.661579288930218	0.071627956854866		0.581626687409844	0.059670136789748
	0.730964267819394	0.069122133946639		0.640407390949721	0.059762992256298
	0.802667011893132	0.071940415673692		0.703150398942120	0.063533327687617
	0.870101046911492	0.064110618987887		0.764427150223209	0.060747103491937
	0.934375852065003	0.062632886080504		0.827286466767321	0.063003575169824
	0.985847235940673	0.035239541651249		0.886328307725370	0.056114474224266
	–	–		0.942576149306792	0.054806665556838
	–	–		0.987615774508326	0.030835997499107
9	0.011008325205230	0.027410088865353	10	0.009907520963014	0.024669150989656
	0.051044032467356	0.048717683543367		0.045939767391172	0.043846061941698
	0.101043770058877	0.049883498131122		0.090939906667056	0.044896047319972
	0.153533718064065	0.056015674884574		0.138182744190679	0.050416940709795
	0.209435701838085	0.054057750384956		0.188500673590387	0.048665699009045
	0.264032718441349	0.056670800869697		0.237667301043392	0.051048534167829
	0.320210113857124	0.054003404171116		0.288322679596029	0.048814929863247
	0.374297895841799	0.055749470152259		0.337453272587259	0.050861702245569
	0.428695931786807	0.050371508064125		0.387667015672957	0.047821735608030
	0.476028082276644	0.047120120933427		0.434697333269808	0.047737938026192
	0.523971917723355	0.047120120933427		0.480525931013719	0.041221259227061
	0.571304068213192	0.050371508064125		0.519474068986280	0.041221259227061
	0.625702104158200	0.055749470152259		0.565302666730919	0.047737938026192
	0.679789886142875	0.054003404171116		0.612332984327042	0.047821735608030
	0.735967281558651	0.056670800869697		0.662546727412740	0.050861702245569
	0.790564298161914	0.054057750384956		0.711677320430971	0.048814929863247
	0.846466281935934	0.056015674884574		0.762332698956608	0.051048534167829
	0.898956229941122	0.049883498131122		0.811499326409612	0.048665699009045
	0.948955967532643	0.048717683543367		0.861817255809320	0.050416940709795
	0.988991674794769	0.027410088865353		0.909060093332943	0.044896047319972
	–	–		0.954006232608827	0.043846061941698
	–	–		0.990092479036985	0.024669150989656

Appendix C. Comparison of estimated costs between trimmed trivariate IGA and traditional FEM

This section provides an estimation of the total computational cost associated with the trimmed B-Spline and FE models presented in Section 7. Note that the corresponding figures are not intended for a rigorous comparison but to give an insight into the simulation times, independent of the solver implementation. We are interested in the computational effort to obtain results of the same quality. To this end, the finest B-Spline discretization ($h = 3$ mm) is compared with the corresponding linear FE model that achieves a similar relative error in total strain energy $\bar{\epsilon}_r$.

Table C.6
Representative numbers of the trimmed B-Spline model and the linear FE model.

	$\bar{\epsilon}_r$	p	n_{dofs}	\hat{n}_{dofs}	n_q	n_{nz}
IGA (C^{p-1})	6.9×10^{-2}	2	9.2×10^4	81	3.1×10^5 (Ω_0^3)	2.6×10^7
FEM (C^0)	7.7×10^{-2}	1	1.1×10^6	12	1.9×10^6	4.5×10^7

Table C.6 lists the error in strain energy $\bar{\epsilon}_r$, the polynomial degree p , the total number of degrees of freedom n_{dofs} , the degrees of freedom per element/ knot span \hat{n}_{dofs} , the number of quadrature points n_q , and the number of non-zero entries of the sparse system matrices n_{nz} for both models. In the following, we estimate the cost required to build and solve the system matrices for linear elasticity problems and give an outlook for explicit dynamic simulations.

C.1. Matrix formation

The predominant cost of constructing the system matrices results from the matrix multiplications of $\mathbf{B}^T \mathbb{C} \mathbf{B} \text{det}(\mathbf{J})w$, where \mathbf{B} denotes the B-operator and \mathbb{C} is the constitutive tensor of linear elasticity. In three dimensions, \mathbf{B} and \mathbb{C} are of size $[6 \times \hat{n}_{\text{dofs}}]$ and $[6 \times 6]$, respectively. If the above product is performed from right to left

$$11 \hat{n}_{\text{dofs}}^2 + 72 \hat{n}_{\text{dofs}} \tag{C.1}$$

floating point operations (flops) are required [99]. Since $\mathbf{B}^T \mathbb{C} \mathbf{B} \text{det}(\mathbf{J})w$ is evaluated at each quadrature point, the total cost adds up to 2.4×10^{10} flops (IGA) and 4.7×10^9 flops (FEM). Note that the difference is less than one order of magnitude. Nevertheless, the additional cost to perform the required matrix formation must be considered, especially for large values of p , since $\hat{n}_{\text{dofs}} = 3(p + 1)^3$ yields a complexity of $O(p^6)$.

C.2. Solution of linear systems of equations

It is known that continuous basis functions such as B-Splines and NURBS can affect the efficiency of linear solvers [97,98]. The authors in [97] compare the cost of direct solvers for systems of equations associated with C^0 and C^{p-1} B-Spline spaces. The corresponding complexity for the entire solution process is estimated to be $O(n_{\text{dofs}} p^6 + n_{\text{dofs}}^2)$ for C^0 basis functions and $O(n_{\text{dofs}}^2 p^3)$ for C^{p-1} continuous bases. Consequently, for large n_{dofs} , the solution of C^{p-1} is p^3 times more expensive compared to C^0 . Therefore, given the same number of degrees of freedom, a C^{p-1} B-Spline space can slow down the solution process. However, Table C.6 shows that the FEM model requires 12-times more degrees of freedom to achieve the same level of accuracy, which relativizes the above statement for the discussed example.

A similar study on the performance of iterative solvers in the context of IGA is presented in [98]. The most expensive operation associated with iterative solvers is the necessary matrix–vector product in each iteration. Its computational cost is proportional to the number of non-zero entries n_{nz} in the sparse system matrices [98]. Note that the FE model contains almost 2-times more non-zero entries than the trimmed IGA model. This indicates that the proposed approach can achieve simulation times similar to those obtained with conventional FE tools, even when the additional cost of preconditioners (see Section 4) is taken into account.

C.3. Explicit algorithms

Although the discussed example is linear static, a brief outlook for explicit dynamics shall be given. We assume that both models produce results of similar accuracy also in a dynamic simulation. If the first-order derivatives are pre-evaluated and stored at each quadrature point, the prevailing cost of the explicit solver can be attributed to the calculation of $\mathbf{B}^T \mathbb{C} \mathbf{B} \dot{\mathbf{u}} \text{det}(\mathbf{J})w$, where $\dot{\mathbf{u}}$ is the local displacement vector [99]. The required number of flops per time iteration is given as

$$13 \hat{n}_{\text{dofs}} + 19. \tag{C.2}$$

Consequently, the evaluation of (C.2) at all quadrature points requires 3.3×10^8 flops (IGA) and 3.3×10^8 flops (FEM). Accordingly, the cost per time step of the FEM model and the B-Spline model is identical. However, due to

the uniform mesh and the C^{p-1} continuous basis functions, the proposed method is distinguished by a significantly larger critical time step Δt_{crit} . Section 6.3.2 shows that Δt_{crit} is practically independent of the trimming operations. In fact, Δt_{crit} is reduced below the values of a uniform hexahedral mesh with the same element length. Therefore, we assume that the classical time step estimation for hexahedral elements provides a conservative lower bound for the Δt_{crit} of the B-Spline discretization. Thus, we predict the critical time step with $\Delta t_{crit} = cl_c$, where l_c is the characteristic length of the smallest element in the mesh and c is a constant. The same equation can be applied to the tetrahedral mesh. While the characteristic length of the B-Spline mesh is $l_c = 3$ mm, the tetrahedral mesh contains elements with $l_c < 0.065$ mm, demanding a drastically smaller critical time step. Note that even the largest tetrahedron has a characteristic length of $l_c < 2$ mm.

References

- [1] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Comput. Methods Appl. Mech. Engrg.* 194 (2005) 4135–4195, <http://dx.doi.org/10.1016/j.cma.2004.10.008>.
- [2] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, *Isogeometric Analysis: Toward Integration of CAD and FEA*, John Wiley & Sons, 2009, <http://dx.doi.org/10.1002/9780470749081>.
- [3] I. Stroud, *Boundary Representation Modelling Techniques*, first ed., Springer, 2006, <http://dx.doi.org/10.1007/978-1-84628-616-2>.
- [4] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, T.W. Sederberg, Isogeometric analysis using T-splines, *Comput. Methods Appl. Mech. Engrg.* 199 (5) (2010) 229–263, <http://dx.doi.org/10.1016/j.cma.2009.02.036>, *Computational Geometry and Analysis*.
- [5] T.W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCs, *ACM Trans. Graph.* 22 (3) (2003) 477–484, <http://dx.doi.org/10.1145/882262.882295>.
- [6] W. Wang, Y. Zhang, G. Xu, T.J.R. Hughes, Converting an unstructured quadrilateral/hexahedral mesh to a rational T-spline, *Comput. Mech.* 50 (2012) 65–84, <http://dx.doi.org/10.1007/s00466-011-0674-6>.
- [7] Y. Zhang, W. Wang, T.J.R. Hughes, Solid T-spline construction from boundary representations for genus-zero geometry, *Comput. Methods Appl. Mech. Engrg.* 249–252 (2012) 185–197, <http://dx.doi.org/10.1016/j.cma.2012.01.014>.
- [8] L. Liu, Y. Zhang, T.J.R. Hughes, M.A. Scott, T.W. Sederberg, Volumetric T-spline construction using Boolean operations, *Eng. Comput.* 30 (2014) 425–439, <http://dx.doi.org/10.1007/s00366-013-0346-6>.
- [9] X. Wei, Y.J. Zhang, D. Toshniwal, H. Speleers, X. Li, C. Manni, J.A. Evans, T.J.R. Hughes, Blended B-spline construction on unstructured quadrilateral and hexahedral meshes with optimal convergence rates in isogeometric analysis, *Comput. Methods Appl. Mech. Engrg.* 341 (2018) 609–639, <http://dx.doi.org/10.1016/j.cma.2018.07.013>.
- [10] H. Al Akhras, T. Elguej, A. Gravouil, M. Rochette, Isogeometric analysis-suitable trivariate NURBS models from standard B-Rep models, *Comput. Methods Appl. Mech. Engrg.* 307 (2016) 256–274, <http://dx.doi.org/10.1016/j.cma.2016.04.028>.
- [11] F. Massarwi, P. Antolin, G. Elber, Volumetric untrimming: Precise decomposition of trimmed trivariates into tensor products, *Comput. Aided Geom. Design* 71 (2019) 1–15, <http://dx.doi.org/10.1016/j.cagd.2019.04.005>.
- [12] S. Klinkel, M. Chasapi, Isogeometric analysis of solids in boundary representation, in: *Novel Finite Element Technologies for Solids and Structures*, Springer International Publishing, Cham, 2020, pp. 153–197, http://dx.doi.org/10.1007/978-3-030-33520-5_6.
- [13] M. Chasapi, L. Mester, B. Simeon, S. Klinkel, Isogeometric analysis of 3D solids in boundary representation for problems in nonlinear solid mechanics and structural dynamics, *Internat. J. Numer. Methods Engrg.* 123 (5) (2022) 1228–1252, <http://dx.doi.org/10.1002/nme.6893>.
- [14] T. Belytschko, R. Gracie, G. Ventura, A review of extended/generalized finite element methods for material modeling, *Modelling Simulation Mater. Sci. Engrg.* 17 (4) (2009) 043001, <http://dx.doi.org/10.1088/0965-0393/17/4/043001>.
- [15] A.R. Khoei, *Extended Finite Element Method: Theory and Applications*, John Wiley & Sons, Ltd, 2014, pp. i–xviii, <http://dx.doi.org/10.1002/9781118869673>.
- [16] E. Burman, S. Claus, P. Hansbo, M.G. Larson, A. Massing, CutFEM: Discretizing geometry and partial differential equations, *Internat. J. Numer. Methods Engrg.* 104 (7) (2015) 472–501, <http://dx.doi.org/10.1002/nme.4823>.
- [17] J. Parvizian, A. Düster, E. Rank, Finite cell method, *Comput. Mech.* 41 (1) (2007) 121–133, <http://dx.doi.org/10.1007/s00466-007-0173-y>.
- [18] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, *Comput. Methods Appl. Mech. Engrg.* 197 (45) (2008) 3768–3782, <http://dx.doi.org/10.1016/j.cma.2008.02.036>.
- [19] E. Rank, M. Ruess, S. Kollmannsberger, D. Schilling, A. Düster, Geometric modeling, isogeometric analysis and the finite cell method, *Comput. Methods Appl. Mech. Engrg.* 249–252 (2012) 104–115, <http://dx.doi.org/10.1016/j.cma.2012.05.022>.
- [20] R. Glowinski, Y. Kuznetsov, Distributed Lagrange multipliers based on fictitious domain method for second order elliptic problems, *Comput. Methods Appl. Mech. Engrg.* 196 (8) (2007) 1498–1506, <http://dx.doi.org/10.1016/j.cma.2006.05.013>.
- [21] I. Ramière, P. Angot, M. Belliard, A general fictitious domain method with immersed jumps and multilevel nested structured meshes, *J. Comput. Phys.* 225 (2) (2007) 1347–1387, <http://dx.doi.org/10.1016/j.jcp.2007.01.026>.
- [22] M. Elhaddad, N. Zander, S. Kollmannsberger, A. Shadavakhsh, V. Nübel, E. Rank, Finite cell method: High-order structural dynamics for complex geometries, *Int. J. Struct. Stab. Dyn.* 15 (07) (2015) 1540018, <http://dx.doi.org/10.1142/S0219455415400180>.
- [23] D. D’Angella, S. Kollmannsberger, A. Reali, E. Rank, T.J.R. Hughes, An accurate strategy for computing reaction forces and fluxes on trimmed locally refined meshes, *J. Mech.* 38 (2022) 60–76, <http://dx.doi.org/10.1093/jom/ufac006>.

- [24] D. Schillinger, M. Ruess, N. Zander, Y. Bazilevs, A. Düster, E. Rank, Small and large deformation analysis with the p- and B-spline versions of the finite cell method, *Comput. Mech.* 50 (2012) 445–478, <http://dx.doi.org/10.1007/S00466-012-0684-Z>.
- [25] R. Schmidt, R. Wüchner, K.-U. Bletzinger, Isogeometric analysis of trimmed NURBS geometries, *Comput. Methods Appl. Mech. Engrg.* 241–244 (2012) 93–111, <http://dx.doi.org/10.1016/j.cma.2012.05.021>.
- [26] M. Breitenberger, A. Apostolatos, B. Philipp, R. Wüchner, K.-U. Bletzinger, Analysis in computer aided design: Nonlinear isogeometric B-Rep analysis of shell structures, *Comput. Methods Appl. Mech. Engrg.* 284 (2015) 401–457, <http://dx.doi.org/10.1016/j.cma.2014.09.033>.
- [27] T.J.R. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, *Comput. Methods Appl. Mech. Engrg.* 199 (5–8) (2010) 301–313, <http://dx.doi.org/10.1016/j.cma.2008.12.004>.
- [28] A.P. Nagy, D.J. Benson, On the numerical integration of trimmed isogeometric elements, *Comput. Methods Appl. Mech. Engrg.* 284 (2015) 165–185, <http://dx.doi.org/10.1016/j.cma.2014.08.002>.
- [29] G. Legrain, Non-negative moment fitting quadrature rules for fictitious domain methods, *Comput. Math. Appl.* 99 (2021) 270–291, <http://dx.doi.org/10.1016/j.camwa.2021.07.019>.
- [30] B. Wassermann, S. Kollmannsberger, T. Bog, E. Rank, From geometric design to numerical analysis: A direct approach using the finite cell method on constructive solid geometry, *Comput. Math. Appl.* 74 (7) (2017) 1703–1726, <http://dx.doi.org/10.1016/j.camwa.2017.01.027>.
- [31] N.M. Patrikakis, T. Sakkalis, G. Shen, Boundary representation models: Validity and rectification, in: *The Mathematics of Surfaces IX*, Springer London, London, 2000, pp. 389–409, http://dx.doi.org/10.1007/978-1-4471-0495-7_23.
- [32] E. Cohen, R.F. Riesenfeld, G. Elber, *Geometric Modeling with Splines: An Introduction*, first ed., A K Peters/CRC Press, 2001, <http://dx.doi.org/10.1201/9781439864203>.
- [33] C. de Boor, A Practical Guide to Splines, in: *Applied Mathematical Sciences*, vol. 27, Springer, New York, 1978, <http://dx.doi.org/10.2307/2006241>.
- [34] L. Piegl, W. Tiller, *The NURBS Book*, first ed., Springer, 1995, <http://dx.doi.org/10.1007/978-3-642-97385-7>.
- [35] G.A. Holzapfel, *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*, first ed., John Wiley & Sons, 2000.
- [36] B.D. Reddy, *Introductory Functional Analysis: With Applications to Boundary Value Problems and Finite Elements*, first ed., Springer-Verlag, New York, 1998, <http://dx.doi.org/10.1007/978-1-4612-0575-3>.
- [37] I. Babuska, The finite element method with penalty, *Math. Comp.* 27 (122) (1973) 221–228, <http://dx.doi.org/10.2307/2005611>.
- [38] A. Apostolatos, R. Schmidt, R. Wüchner, K.-U. Bletzinger, A Nitsche-type formulation and comparison of the most common domain decomposition methods in isogeometric analysis, *Internat. J. Numer. Methods Engrg.* 97 (7) (2014) 473–504, <http://dx.doi.org/10.1002/nme.4568>.
- [39] T.J.R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publications, 2000.
- [40] J.A. Cottrell, A. Reali, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of structural vibrations, *Comput. Methods Appl. Mech. Engrg.* 195 (41) (2006) 5257–5296, <http://dx.doi.org/10.1016/j.cma.2005.09.027>.
- [41] F. Auricchio, L. Beirão da Veiga, T.J.R. Hughes, A. Reali, G. Sangalli, Isogeometric collocation for elastostatics and explicit dynamics, *Comput. Methods Appl. Mech. Engrg.* 249–252 (2012) 2–14, <http://dx.doi.org/10.1016/j.cma.2012.03.026>.
- [42] F.P. Preparata, M.I. Shamos, *Computational Geometry: An Introduction*, first ed., Springer, New York, 1985, <http://dx.doi.org/10.1007/978-1-4612-1098-6>.
- [43] B. Wassermann, S. Kollmannsberger, S. Yin, L. Kudela, E. Rank, Integrating CAD and numerical analysis: ‘Dirty geometry’ handling using the finite cell method, *Comput. Methods Appl. Mech. Engrg.* 351 (2019) 808–835, <http://dx.doi.org/10.1016/j.cma.2019.04.017>.
- [44] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Lévy, *Polygon Mesh Processing*, AK Peters, Natick, Mass., 2010, p. 250.
- [45] M. Campen, L. Kobbelt, Exact and robust (self-)intersections for polygonal meshes, *Comput. Graph. Forum* 29 (2) (2010) 397–406, <http://dx.doi.org/10.1111/j.1467-8659.2009.01609.x>.
- [46] The CGAL Project, *CGAL User and Reference Manual*, fifth, fourth ed., CGAL Editorial Board, 2022, URL <https://doc.cgal.org/5.4/Manual/packages.html>.
- [47] S. Lorient, M. Rouxel-Labbé, J. Tournois, I.O. Yaz, *Polygon mesh processing*, in: *CGAL User and Reference Manual*, fifth, fourth ed., CGAL Editorial Board, 2022.
- [48] L. Kudela, U. Almac, S. Kollmannsberger, E. Rank, Direct numerical analysis of historical structures represented by point clouds, in: *Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection*, Springer, Cham, 2018, pp. 64–75, http://dx.doi.org/10.1007/978-3-030-01762-0_6.
- [49] R.R. Hiemstra, F. Calabrò, D. Schillinger, T.J.R. Hughes, Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis, *Comput. Methods Appl. Mech. Engrg.* 316 (2017) 966–1004, <http://dx.doi.org/10.1016/j.cma.2016.10.049>.
- [50] F. Auricchio, F. Calabrò, T.J.R. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis, *Comput. Methods Appl. Mech. Engrg.* 249–252 (2012) 15–27, <http://dx.doi.org/10.1016/j.cma.2012.04.014>.
- [51] C. Adam, T.J.R. Hughes, S. Bouabdallah, M. Zarroug, H. Maitournam, Selective and reduced numerical integrations for NURBS-based isogeometric analysis, *Comput. Methods Appl. Mech. Engrg.* 284 (2015) 732–761, <http://dx.doi.org/10.1016/j.cma.2014.11.001>.
- [52] R. Ait-Haddou, M. Bartoň, V.M. Calo, Explicit Gaussian quadrature rules for C1 cubic splines with symmetrically stretched knot sequences, *J. Comput. Appl. Math.* 290 (2015) 543–552, <http://dx.doi.org/10.1016/j.cam.2015.06.008>.
- [53] M. Bartoň, V.M. Calo, Gaussian quadrature for splines via homotopy continuation: Rules for C2 cubic splines, *J. Comput. Appl. Math.* 296 (2016) 709–723, <http://dx.doi.org/10.1016/j.cam.2015.09.036>.
- [54] M. Bartoň, R. Ait-Haddou, V.M. Calo, Gaussian quadrature rules for C1 quintic splines with uniform knot vectors, *J. Comput. Appl. Math.* 322 (2017) 57–70, <http://dx.doi.org/10.1016/j.cam.2017.02.022>.

- [55] Z. Zou, T.J.R. Hughes, M.A. Scott, R.A. Sauer, E.J. Savitha, Galerkin formulations of isogeometric shell analysis: Alleviating locking with Greville quadratures and higher-order elements, *Comput. Methods Appl. Mech. Engrg.* 380 (2021) 113757, <http://dx.doi.org/10.1016/j.cma.2021.113757>.
- [56] Z. Zou, T.J.R. Hughes, M.A. Scott, D. Miao, R.A. Sauer, Efficient and robust quadratures for isogeometric analysis: Reduced Gauss and Gauss–Greville rules, *Comput. Methods Appl. Mech. Engrg.* 392 (2022) 114722, <http://dx.doi.org/10.1016/j.cma.2022.114722>.
- [57] S. Loehnert, D.S. Mueller-Hoeppe, P. Wriggers, 3D corrected XFEM approach and extension to finite deformation theory, *Internat. J. Numer. Methods Engrg.* 86 (4–5) (2011) 431–452, <http://dx.doi.org/10.1002/nme.3045>.
- [58] L. Kudela, N. Zander, S. Kollmannsberger, E. Rank, Smart octrees: Accurately integrating discontinuous functions in 3D, *Comput. Methods Appl. Mech. Engrg.* 306 (2016) 406–426, <http://dx.doi.org/10.1016/j.cma.2016.04.006>.
- [59] S. Hubrich, P. Di Stolfo, L. Kudela, S. Kollmannsberger, E. Rank, A. Schröder, A. Düster, Numerical integration of discontinuous functions: moment fitting and smart octree, *Comput. Mech.* 60 (5) (2017) 863–881, <http://dx.doi.org/10.1007/s00466-017-1441-0>.
- [60] H. Xiao, Z. Gimbutas, A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions, *Comput. Math. Appl.* 59 (2) (2010) 663–676, <http://dx.doi.org/10.1016/j.camwa.2009.10.027>.
- [61] M. Joulaian, S. Hubrich, A. Düster, Numerical integration of discontinuities on arbitrary domains based on moment fitting, *Comput. Mech.* 57 (2016) 979–999, <http://dx.doi.org/10.1007/s00466-016-1273-3>.
- [62] B. Müller, F. Kummer, M. Oberlack, Highly accurate surface and volume integration on implicit domains by means of moment-fitting, *Internat. J. Numer. Methods Engrg.* 96 (8) (2013) 512–528, <http://dx.doi.org/10.1002/nme.4569>.
- [63] Y. Sudhakar, W.A. Wall, Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods, *Comput. Methods Appl. Mech. Engrg.* 258 (2013) 39–54, <http://dx.doi.org/10.1016/j.cma.2013.01.007>.
- [64] P. Antolin, T. Hirschler, Quadrature-free immersed isogeometric analysis, *Eng. Comput.* (2022) <http://dx.doi.org/10.1007/s00366-022-01644-3>.
- [65] S.E. Mousavi, N. Sukumar, Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons, *Comput. Mech.* 47 (5) (2011) 535–554, <http://dx.doi.org/10.1007/s00466-010-0562-5>.
- [66] S. Hubrich, M. Joulaian, A. Düster, Numerical integration in the finite cell method based on moment-fitting, in: *Proceedings of 3rd ECCOMAS Young Investigators Conference; 6th GACM Colloquium on Computational Mechanics*, Aachen, Germany, 2015.
- [67] C.L. Lawson, R.J. Hanson, *Solving Least Squares Problems*, Vol. 15, SIAM, 1995, pp. I–XII, 1–337.
- [68] F. de Prenter, C.V. Verhoosel, E.H. van Brummelen, Preconditioning immersed isogeometric finite element methods with application to flow problems, *Comput. Methods Appl. Mech. Engrg.* 348 (2019) 604–631, <http://dx.doi.org/10.1016/j.cma.2019.01.030>.
- [69] R. Glowinski, T.-W. Pan, J. Periaux, A fictitious domain method for Dirichlet problem and applications, *Comput. Methods Appl. Mech. Engrg.* 111 (3) (1994) 283–303, [http://dx.doi.org/10.1016/0045-7825\(94\)90135-X](http://dx.doi.org/10.1016/0045-7825(94)90135-X).
- [70] D. Elfverson, M.G. Larson, K. Larsson, CutIGA with basis function removal, *Adv. Model. Simul. Eng. Sci.* 5 (6) (2018) <http://dx.doi.org/10.1186/s40323-018-0099-2>.
- [71] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method, *Appl. Numer. Math.* 62 (4) (2012) 328–341, <http://dx.doi.org/10.1016/j.apnum.2011.01.008>.
- [72] J.N. Jomo, F. de Prenter, M. Elhaddad, D. D’Angella, C.V. Verhoosel, S. Kollmannsberger, J.S. Kirschke, V. Nübel, E.H. van Brummelen, E. Rank, Robust and parallel scalable iterative solutions for large-scale finite cell analyses, *Finite Elem. Anal. Des.* 163 (2019) 14–30, <http://dx.doi.org/10.1016/j.finel.2019.01.009>.
- [73] J.N. Jomo, O. Oztoprak, F. de Prenter, N. Zander, S. Kollmannsberger, E. Rank, Hierarchical multigrid approaches for the finite cell method on uniform and multi-level hp-refined grids, *Comput. Methods Appl. Mech. Engrg.* 386 (2021) 114075, <http://dx.doi.org/10.1016/j.cma.2021.114075>.
- [74] S.C. Brenner, L.R. Scott, *The Mathematical Theory of Finite Element Methods*, third ed., Springer, 2008, <http://dx.doi.org/10.1007/978-0-387-75934-0>.
- [75] L.F. Leidinger, *Explicit Isogeometric B-Rep Analysis for Nonlinear Dynamic Crash Simulations* (Dissertation), Technische Universität München, 2020.
- [76] T. Teschemacher, A.M. Bauer, T. Oberbichler, M. Breitenberger, R. Rossi, R. Wüchner, K.-U. Bletzinger, Realization of CAD-integrated shell simulation based on isogeometric B-Rep analysis, *Adv. Model. Simul. Eng. Sci.* 5 (19) (2018) <http://dx.doi.org/10.1186/s40323-018-0109-4>.
- [77] R. McNeel, et al., *Rhinoceros 3D, Version 7.11*, Robert McNeel & Associates, Seattle, WA, 2021.
- [78] T. Teschemacher, A.M. Bauer, R. Aristio, M. Meßner, R. Wüchner, K.-U. Bletzinger, Cocodrilo, <https://github.com/CocodriloCAD/Cocodrilo>.
- [79] T. Teschemacher, A.M. Bauer, R. Aristio, M. Meßner, R. Wüchner, K.-U. Bletzinger, Concepts of data collection for the CAD-integrated isogeometric analysis, *Eng. Comput.* (2022) accepted for publication.
- [80] M. Meßner, TIBRA, <https://github.com/manuelmessner/TIBRA>.
- [81] P. Dadvand, R. Rossi, E. Oñate, An object-oriented environment for developing finite element codes for multi-disciplinary applications, *Arch. Comput. Methods Eng.* 17 (2010) 253–297, <http://dx.doi.org/10.1007/s11831-010-9045-2>.
- [82] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S. Idelsohn, E. Oñate, Migration of a generic multi-physics framework to HPC environments, *Comput. & Fluids* 80 (2013) 301–309, <http://dx.doi.org/10.1016/j.compfluid.2012.02.004>.
- [83] V.M. Ferrández, P. Bucher, R. Rossi, J. Cotela, J. Carbonell, R. Zorrilla, R. Tosi, et al., KratosMultiphysics (Version 8.0), Zenodo, 2020, <http://dx.doi.org/10.5281/zenodo.3234644>.
- [84] M. Meßner, L.F. Leidinger, S. Hartmann, F. Bauer, F. Duddeck, R. Wüchner, K.-U. Bletzinger, Isogeometric analysis on trimmed solids: A B-spline-based approach focusing on explicit dynamics, in: *Proceedings of 13th European LS-DYNA Conference*, Ulm, Germany, 2021.

- [85] J.O. Hallquist, *LS-DYNA Theory Manual*, Livermore Software Technology Corporation (LSTC), 2017.
- [86] S.P. Timoshenko, J.N. Goodier, *Theory of Elasticity*, McGraw-Hill, 1951.
- [87] T. Kaneko, On Timoshenko's correction for shear in vibrating beams, *J. Phys. D: Appl. Phys.* 8 (16) (1975) 1927–1936, <http://dx.doi.org/10.1088/0022-3727/8/16/003>.
- [88] L.P. Gould, Y. Feng, *Introduction to Linear Elasticity*, fourth ed., Springer, 2018, <http://dx.doi.org/10.1007/978-3-319-73885-7>.
- [89] R.D. Mindlin, Simple modes of vibration of crystals, *J. Appl. Phys.* 27 (12) (1956) 1462–1466, <http://dx.doi.org/10.1063/1.1722290>.
- [90] C. Anitescu, C. Nguyen, T. Rabczuk, X. Zhuang, Isogeometric analysis for explicit elastodynamics using a dual-basis diagonal mass formulation, *Comput. Methods Appl. Mech. Engrg.* 346 (2019) 574–591, <http://dx.doi.org/10.1016/j.cma.2018.12.002>.
- [91] C. Adam, S. Bouabdallah, M. Zarroug, H. Maitournam, Stable time step estimates for NURBS-based explicit dynamics, *Comput. Methods Appl. Mech. Engrg.* 295 (2015) 581–605, <http://dx.doi.org/10.1016/j.cma.2015.03.017>.
- [92] D. Wang, W. Liu, H. Zhang, Novel higher order mass matrices for isogeometric structural vibration analysis, *Comput. Methods Appl. Mech. Engrg.* 260 (2013) 92–108, <http://dx.doi.org/10.1016/j.cma.2013.03.011>.
- [93] T. Belytschko, W.K. Liu, B. Moran, K. Elkhodary, *Nonlinear Finite Elements for Continua and Structures*, second ed., John Wiley & Sons, 2014.
- [94] L.F. Leidinger, M. Breitenberger, A.M. Bauer, S. Hartmann, R. Wüchner, K.-U. Bletzinger, F. Duddeck, L. Song, Explicit dynamic isogeometric B-Rep analysis of penalty-coupled trimmed NURBS shells, *Comput. Methods Appl. Mech. Engrg.* 351 (2019) 891–927, <http://dx.doi.org/10.1016/j.cma.2019.04.016>.
- [95] K.D. Hjelmstad, *Fundamentals of Structural Dynamics*, first ed., Springer, 2022, <http://dx.doi.org/10.1007/978-3-030-89944-8>.
- [96] T.J.R. Hughes, K.S. Pister, R.L. Taylor, Implicit-explicit finite elements in nonlinear transient analysis, *Comput. Methods Appl. Mech. Engrg.* 17–18 (1979) 159–182, [http://dx.doi.org/10.1016/0045-7825\(79\)90086-0](http://dx.doi.org/10.1016/0045-7825(79)90086-0).
- [97] N. Collier, D. Pardo, L. Dalcin, M. Paszynski, V.M. Calo, The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers, *Comput. Methods Appl. Mech. Engrg.* 213–216 (2012) 353–361, <http://dx.doi.org/10.1016/j.cma.2011.11.002>.
- [98] N. Collier, L. Dalcin, D. Pardo, V.M. Calo, The cost of continuity: Performance of iterative solvers on isogeometric finite elements, *SIAM J. Sci. Comput.* 35 (2) (2013) A767–A784, <http://dx.doi.org/10.1137/120881038>.
- [99] D. Schilling, J.A. Evans, A. Reali, M.A. Scott, T.J.R. Hughes, Isogeometric collocation: Cost comparison with Galerkin methods and extension to adaptive hierarchical NURBS discretizations, *Comput. Methods Appl. Mech. Engrg.* 267 (2013) 170–232, <http://dx.doi.org/10.1016/j.cma.2013.07.017>.

Publication II

Comput. Methods Appl. Mech. Engrg. 419 (2024) 116670

Robust numerical integration of embedded solids described in boundary representation

Manuel Meßmer, Stefan Kollmannsberger, Roland Wüchner,
Kai-Uwe Bletzinger

Highlights

- Fast assembly of the moment fitting equations for arbitrarily complex faceted B-Reps.
- Efficient intersection algorithm to facilitate the application of the divergence theorem.
- Generalized and parallel-executable element classification scheme based on flood filling.
- Successful application of the framework to 4948 valid and flawed STLs.

Doi: <https://doi.org/10.1016/j.cma.2023.116670>

The included publication is an open access article published by Elsevier B.V. under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).



Robust numerical integration of embedded solids described in boundary representation

Manuel Meßmer^{a,*}, Stefan Kollmannsberger^b, Roland Wüchner^c,
Kai-Uwe Bletzinger^a

^a Chair of Structural Analysis, Technical University of Munich, Arcisstr. 21, 80333 München, Germany

^b Chair of Computational Modeling and Simulation, Technical University of Munich, Arcisstr. 21, 80333 München, Germany

^c Institute of Structural Analysis, Technische Universität Braunschweig, Beethovenstr. 51, 38106 Braunschweig, Germany

ARTICLE INFO

Dataset link: <https://github.com/manuelmessmer/QuESo>

Keywords:

Embedded finite elements
Boundary representation (B-rep)
Moment fitting equations
Clipping and intersection algorithms
Flawed geometries

ABSTRACT

Embedded and immersed methods have become essential tools in computational mechanics, as they allow discretizing arbitrarily complex geometries without the need for boundary-fitted meshes. One of their main challenges is the accurate numerical integration of cut elements. Among the various integration schemes developed for this purpose, moment fitting has proven to be a powerful technique that provides highly efficient and accurate integration rules.

This publication presents a framework for the robust and efficient numerical integration of embedded solids described by oriented boundary meshes using moment fitting. The developments include an intersection algorithm that aims to drastically accelerate the computation of the necessary moments while achieving high accuracy. A closed surface parameterization of each cut domain is computed to facilitate the direct application of the divergence theorem. The algorithm is subject to a single quality criterion that guarantees the accurate evaluation of boundary integrals. At the same time, it allows to disregard classical mesh criteria, such as high aspect ratios, strongly varying angles, etc., resulting in extremely fast runtimes. In addition, an existing robust flood fill-based element classification scheme is further developed to initiate filling from arbitrary seed elements and to enable parallel execution, increasing its flexibility and efficiency.

The successful application of all proposed algorithms to 4948 valid and flawed STLs from the Thing10K database (Zhou and Jacobson, 2016) demonstrates their extraordinary robustness. In all cases, the wall-clock time scales at most linearly with the number of elements in the background mesh. We show that higher-order quadrature rules on the boundary elements enable efficient computation of the moments via the divergence theorem with near-machine precision. Finally, the presented methodologies are used to perform direct FE analyses on clean and flawed B-Rep models. All proposed algorithms are publicly available in the open-source C++ framework QuESo – Quadrature for Embedded Solids (<https://github.com/manuelmessmer/QuESo>), where the moment fitting equations are assembled and solved.

* Corresponding author.

E-mail address: manuel.messmer@tum.de (M. Meßmer).

1. Introduction

Numerical integration plays a vital role in various scientific and engineering fields. In the particular case of the Finite Element Method (FEM), quadrature rules are used to evaluate the weak formulation of the underlying Partial Differential Equation (PDE). Since each finite element is classically mapped to a standardized subspace, the positions and weights of the integration points are known a priori. Thus, the quadrature rules can be tabulated for each element when a boundary-fitted discretization explicitly describes the integration domain. However, the generation of such boundary-fitted models for arbitrarily complex geometries is, until today, a delicate problem. In state-of-the-art development pipelines, FE discretizations are usually derived from explicitly defined geometries described by Non-Uniform Rational B-Splines (NURBS). This process, known as meshing, is time-consuming, error-prone, and often requires manual interventions. To circumvent the undesirable model conversion, Hughes et al. introduced the Isogeometric Analysis (IGA), which allows FE analyses to be performed directly on the exact NURBS-based geometries [1,2]. However, the so-called Boundary Representation (B-Rep) of three-dimensional shapes used in modern CAD software poses severe challenges for analyzing solids by isogeometric methods. Since volumes are exclusively described by their bounding faces, a suitable volumetric function space must still be created during preprocessing. Embedded and immersed boundary methods provide a generic approach for discretizing complex domains and have, therefore, become increasingly important in the isogeometric community, among other fields.

Their main philosophy is to solve PDEs on a simple computational mesh, i.e., a regular grid, often also referred to as background mesh, where the geometric boundaries are allowed to intersect elements. Prominent examples are the eXtended Finite Element Method (XFEM) [3], the Cut Finite Element Method (CutFEM) [4,5], the Aggregated Finite Element Method (AgFEM) [6,7], IGA of trimmed Non-Uniform Rational B-Spline (NURBS) patches [8–10], and the Finite Cell Method (FCM) [11,12]. These technologies, recently reviewed in [13], represent the geometry not explicitly by a boundary-fitted discretization but implicitly by integrating discontinuous functions along the interface. The main challenge is that the discontinuity's location, topology, and geometry can be arbitrary, preventing the use of classical quadrature rules. To this end, it is essential for embedded or immersed boundary methods to construct appropriate integration schemes automatically and without user interaction as fast and robustly as possible.

A common approach is subdividing cut elements in order to decompose and simplify the given integrals. In many applications, tessellation algorithms [14–18] or space trees, i.e., quadtree in 2D, and octree in 3D, [12,19,20] are applied for this purpose. However, due to the low-order approximation of the geometry, the number of integration points may be unnecessarily high. Adaptive space trees can reduce the number of points by Boolean operations [21,22] or merging of subcells [23]. In [24], the octree is equipped with a node relocation algorithm and a specific mapping scheme to increase its flexibility and improve the quadrature rule's efficiency. Other approaches retain the original integration domain but modify the integrand accounting for appearing discontinuities [25,26]. The works in [27,28] successfully apply the divergence theorem to reduce the dimension of the respective integral by one. Another method, which is characterized by highly efficient quadrature rules, involves the solution of the moment fitting equations [29–31]. The main idea is to optimize a set of integration points based on known reference integrals. Challenges that arise during the assembly and solution of the moment fitting equations have been an active area of research over the past decade. One problem is the strong nonlinear dependency on the location of integration points. The authors in [32] address this issue with an optimization strategy approximating the nonlinear solution. To drastically simplify the system of equations, [30] suggests redefining the locations of the integration points a priori. In addition, point elimination may be employed to find the best points from a discrete set [31,33]. Applying a Non-Negative-Least-Square (NNLS) solver guarantees positive defined weights [34,35]. In [36], the concept of point elimination is coupled with the NNLS achieving $n_q \leq (p+1)^3$ quadrature points with positive weights and locations restricted to the material domain for arbitrarily cut three-dimensional elements. Nevertheless, regardless of the chosen solution technique, the quality of the integrated moments determines the overall accuracy. Technically, each previously mentioned quadrature scheme is also a potential candidate for evaluating the moment fitting basis functions. Due to the simplicity of these bases, other methods have also gained increasing attention. The authors in [37] utilize Lasserre's theorem to reduce the integral dimension, simplifying the integration of homogeneous functions over convex polygons or polytopes. The divergence theorem may be applied for the same purpose, expanding this concept to arbitrarily shaped domains [38,39]. Since the moment fitting bases are known and well-defined, also their anti-derivatives are available, rendering the divergence theorem an ideal tool for the computation of the moments. However, its application to embedded methods, as in [40], requires a suitable surface parameterization for each cut element, which must be computed through Boolean operations. Since this is a complex task, it is usually outsourced to available computational geometry libraries, e.g., CGAL [41], OpenCascade [42], or GTS [43]. However, these libraries aim at general algorithms with a wide range of applications. They are not specifically designed for the intersection between one geometrical boundary and thousands or millions of background elements. Therefore, the geometrical operations can quickly become the bottleneck in the entire simulation pipeline. Moreover, most algorithms only accept watertight (closed) geometries, which in many cases, are not available.

This work presents a framework for the robust and efficient numerical integration of embedded B-Reps. The developments involve a clipping and intersection algorithm tailored to efficiently assemble the moment fitting equations using the divergence theorem. We do not aim at a novel solution strategy for the moment fitting equations but at the accurate computation of the moments, which is a crucial step in all moment fitting methods. In particular, the focus lies on the necessary geometrical operations. The algorithm is designed to work with arbitrarily complex geometries described by oriented boundary meshes, e.g., STereoLithography (STL) meshes, which can be retrieved from any CAD program. We assume that each integration domain (element) is a rectangular cuboid, corresponding to the standard discretization used in classical embedded methods. Note that this does not only include C^0

continuous background meshes but also higher continuous function spaces defined by trivariate B-Splines [44,45]. In contrast to the intersection algorithm recently proposed in [46], our implementation does not target a volumetric representation of the cut domain through, e.g., polytopes. Instead, the core idea is to drastically accelerate required geometrical operations by providing only the absolute minimum information needed for the application of the divergence theorem. This is realized by computing a closed surface mesh for each cut element, which is completely free of unnecessary and computationally intensive quality requirements. Since the mesh is solely used for integration purposes, high aspect ratios, hanging or duplicated vertices, zero-area triangles, etc., do not cause errors or limit accuracy. The algorithm is subject to a single quality measure based on known solutions of the divergence theorem, which ensures an accurate evaluation of the boundary integrals. The second important development concerns the classification of the background discretization into interior, exterior, and cut elements. We extend the robust cell classification scheme developed in [47] for solid geometries containing cavities. In addition, the algorithm is further improved to allow its execution in a parallel environment, resulting in substantial runtime reductions.

All algorithms presented in this paper are verified by implementation in C++ and are available to the interested reader in a coherent library called QuEso [48] (Quadrature for Embedded Solids). QuEso combines the new developments with the point elimination algorithm proposed in [36] to solve the moment fitting equations for highly efficient and accurate quadrature rules. The result is a complete preprocessing framework from the initial design represented as STL to an analysis-ready embedded FE model. For a subsequent FE analysis, only the background mesh and a set of integration points are passed to the FE solver. This renders the integration of QuEso into existing simulation software very easy.

An overview of the paper at hand is given in the following.

- Section 2 discusses the fundamentals of embedded boundary methods.
- Section 3 summarizes a state-of-the-art approach to construct efficient quadrature rules for arbitrarily shaped domains by means of moment fitting.
- Section 4 introduces the notation and functions used in our pseudo-codes.
- Section 5 presents the proposed embedding pipeline for the fast and robust integration of volumetric B-Rep models.
- Section 6 demonstrates the potential of the proposed workflow. A comparison with classical octree integration and the results of an extensive robustness and efficiency analysis are presented. Moreover, the proposed framework is used to perform direct FE analysis of valid and flawed geometries.
- Section 7 concludes the present work.

2. Embedded methods

Let $\Omega \in \mathbb{R}^3$ be the physical domain in which a set of PDEs shall be approximated. The boundary of Ω is denoted as $\Gamma = \partial\Omega$, which is split into a Dirichlet boundary Γ_D and a Neumann boundary Γ_N , such that $\Gamma_D \cup \Gamma_N = \Gamma$ and $\Gamma_D \cap \Gamma_N = \emptyset$. The core idea of traditional embedded methods is to discretize Ω by means of an unfitted background partition \mathcal{P}_{bg} . Reducing the meshing effort to an absolute minimum, \mathcal{P}_{bg} is classically defined by a simple grid-based discretization with hexahedral elements/cells $\mathcal{B} \in \mathcal{P}_{\text{bg}}$. Consequently, if domain Ω is embedded in \mathcal{P}_{bg} , cut cells \mathcal{B}_{cut} are inevitable. For uncult cells, a distinction is made between interior \mathcal{B}_{in} and exterior cells \mathcal{B}_{out} . The active subset of \mathcal{P}_{bg} is defined as $\mathcal{P} = \mathcal{P}_{\text{bg}} \setminus \mathcal{P}_{\text{out}}$. This publication aims to provide a framework for the efficient construction of quadrature rules based on Γ as the only input. In particular, we assume that Γ is given as an oriented surface mesh, i.e., an STL, which is a common scenario in industrial and scientific workflows.

Classically, the weak form of the finite element formulation contains bulk and surface terms. Due to their definition as piecewise polynomials, these terms must be evaluated in a cell-wise decomposition. Considering the background mesh \mathcal{P} , the integrals over the domain Ω take the following well-known form

$$\int_{\Omega} (\cdot) \, d\Omega = \sum_{\mathcal{B} \in \mathcal{P}} \int_{\mathcal{B} \cap \Omega} (\cdot) \, d\Omega. \quad (1)$$

Since the faces and, thus, the individual integration domains of Γ do not coincide with the boundaries of $\mathcal{B} \in \mathcal{P}$, the corresponding integrals are also performed in a cell-wise decomposition

$$\int_{\Gamma_{\beta}} (\cdot) \, d\Gamma = \sum_{\mathcal{B} \in \mathcal{P}} \sum_{F_{\beta} \in \mathcal{I}_{\beta}} \int_{F_{\beta} \cap \mathcal{B}} (\cdot) \, d\Gamma, \quad \beta \in \{D, N\}, \quad (2)$$

where F_D and F_N are the Dirichlet and Neumann boundary conditions defined on Γ_D and Γ_N , respectively. Since $\mathcal{B}_{\text{in}} = \mathcal{B}_{\text{in}} \cap \Omega$ always holds, the integration of all $\mathcal{B}_{\text{in}} \in \mathcal{P}$ in Eq. (1) is straightforward, and standard Gauss quadrature schemes can be applied. However, the evaluation of $\mathcal{B}_{\text{cut}} \cap \Omega$ requires more sophisticated strategies. The next section outlines the basic steps to convert the respective volume integral to a surface integral over $\partial(\mathcal{B}_{\text{cut}} \cap \Omega)$. Section 5 presents the main algorithm that performs all necessary geometrical operations to obtain $\partial(\mathcal{B}_{\text{cut}} \cap \Omega)$ (required for Eq. (1)) and $\Gamma \cap \mathcal{B}_{\text{cut}}$ (required for Eq. (2)).

3. Construction of efficient quadrature rules for arbitrarily cut domains

The common difficulty of most embedded and immersed methods is the accurate integration of arbitrarily cut domains. In this section, we outline a state-of-the-art approach to construct efficient quadrature rules and emphasize the necessary inputs required for this method.

3.1. Moment fitting

The moment fitting method is a powerful tool for finding a set of optimized integration points based on known reference integrals. Given the physical domain $\Omega_c = \mathcal{B}_{\text{cut}} \cap \Omega$, its classical form can be stated as

$$\begin{bmatrix} f_1(\mathbf{x}_1) & \cdots & f_1(\mathbf{x}_{n_q}) \\ \vdots & \ddots & \vdots \\ f_m(\mathbf{x}_1) & \cdots & f_m(\mathbf{x}_{n_q}) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_{n_q} \end{bmatrix} = \begin{bmatrix} \int_{\Omega_c} f_1(\mathbf{x}) \, d\Omega \\ \vdots \\ \int_{\Omega_c} f_m(\mathbf{x}) \, d\Omega \end{bmatrix}, \quad (3)$$

where \mathbf{x}_i and w_i are each point's distinct positions and weights, and f_j are the moment fitting basis functions. The integrals on the right-hand side of Eq. (3) are called the moments or the constant terms of the moment fitting equations. Generally, any set of linearly independent functions that is capable of representing the target integrand may be used as moment fitting bases. However, orthogonal bases, e.g., Legendre polynomials, are often preferred over monomials due to better conditioning of Eq. (3) [31,36,40]. Since the moment fitting equations are highly nonlinearly dependent on the positions \mathbf{x}_i , Eq. (3) is classically linearized by defining the locations of the integration points a priori [30,39,49]. Consequently, Eq. (3) transforms into a linear system of equations, which only depends on the integration weights w_i . The authors in [50] show that Lagrange polynomials through Gauss–Legendre points achieve a diagonal moment fitting matrix, which circumvents having to solve Eq. (3). A point elimination algorithm may be employed to find the most suitable point locations from a predefined discrete set [31,33]. Moreover, a Non-Negative Least Square (NNLS) solver inherently guarantees positively defined weights without additional feasibility constraints [34,51]. In [36], the point elimination algorithm presented in [31,33] is combined with the proposed NNLS solver and successfully applied to three-dimensional domains. It is shown that accurate quadrature rules can be constructed for arbitrarily cut elements with $n_q \leq (p+1)^d$ quadrature points, where p denotes the polynomial degree of the basis functions, and d is the dimension of the integration space. In addition, all integration weights are positive, and point locations are restricted to the material domain, which is advantageous in the presence of geometric and material nonlinearities [32,35]. These results are obtained using a tensor product of functions as the moment fitting bases. The function space is defined as

$$\mathbf{F} = \{f_j(\mathbf{x}) = L_r(x)L_s(y)L_t(z); \, r, s, t = 0, 1, 2, \dots, p\}, \quad (4)$$

where L_r is the r th Legendre polynomial.

This brief overview aims to give a glimpse of the diversity of existing moment fitting methods, providing quadrature rules with distinct properties that fulfill the needs of varying applications. However, regardless of the solution strategy used for Eq. (3), the accuracy of the final quadrature rule is always limited by the quality of the computed moments. *Thus, this work provides a framework to robustly and accurately evaluate the moments over arbitrarily complex cut domains that can be combined with any moment fitting method.* In the following, we use the moment fitting scheme presented in [36] to demonstrate the potential of the proposed developments.

3.2. Computation of the moments

As discussed in the previous subsection, different solution strategies involving the moment fitting equations are available to achieve highly efficient and accurate integration points. Nevertheless, when explicitly defined B-Reps are considered, the computation of the moments can quickly become the bottleneck in the entire process. Generally, all available integration schemes that can deal with discontinuous functions are potential candidates for the assembly of the right-hand side in Eq. (3) [52]. However, most of these methods rely on a spatial refinement or tessellation of the B-Rep, which often negates the advantages of embedded methods. In this work's scope, we hence employ the divergence theorem to transform the necessary volume integrals into surface integrals [38–40]

$$\int_{\Omega_c} f_j(\mathbf{x}) \, d\Omega = \int_{\Omega_c} \nabla \cdot \mathbf{g}_j(\mathbf{x}) \, d\Omega = \int_{\Gamma_c} \mathbf{g}_j(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) \, d\Gamma, \quad (5)$$

where \mathbf{n} denotes the outward pointing unit normal on the boundary $\Gamma_c = \partial\Omega_c$. For the anti-derivatives \mathbf{g}_j , we adopt the notation from [39]

$$\mathbf{g}_j(\mathbf{x}) = \frac{1}{3} \begin{bmatrix} \int f_j(\mathbf{x}) \, dx \\ \int f_j(\mathbf{x}) \, dy \\ \int f_j(\mathbf{x}) \, dz \end{bmatrix}. \quad (6)$$

As a result, Eqs. (5)–(6) allow the integration of f_j over Ω without the need for a spatial refinement or tessellation of the three-dimensional domain.

3.3. Application to B-Reps

Since Eq. (5) only contains boundary integrals, it is predestined for evaluating domains enclosed by a B-Rep model. Assuming the boundary parameterization Γ_c is available as an oriented triangle mesh, i.e., STL, Eq. (5) may be evaluated by

$$\int_{\Omega_c} f_j(\mathbf{x}) \, d\Omega = \sum_{f \in \Gamma_c} \int_f \mathbf{g}_j(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) \, d\Gamma, \quad (7)$$

with f denoting a triangular face. However, the evaluation of Eq. (7) requires a closed surface parameterization of each cut domain $\Gamma_c = \partial(\mathcal{B}_{\text{cut}} \cap \Omega)$. To this end, Section 5 presents an efficient intersection algorithm tailored to the Boolean operations between an arbitrarily complex surface mesh and a bounding box (cell/element).

4. Definitions and notation

This section introduces the notation and basic functions used in the following. Generally, we define a set A as $A = \{a, b, c\}$, whose i th element can be accessed by A^i . The indices of A range from $0 \leq i < n$. All algorithmic functions and variables are written as FUNCTIONS() and VARIABLES. Some of the rudimentary functions used in the upcoming pseudo-codes are listed and briefly described in the following:

- INITIALIZESET(n, c): Creates a set of size n with all values equal to c .
- RANGE($0, n$): Returns a sequence of numbers from 0 to $n - 1$.
- $A[i]$: Returns the i th element contained in A .
- SIZE(A): Returns the number of elements contained in A .
- LAST(A): Returns the last element contained in A .
- APPEND(A, B): Returns $A \cup B$.
- ZIP(A, B): Returns $\{\{A^0, B^0\}, \{A^1, B^1\}, \dots, \{A^{n-1}, B^{n-1}\}\}$.
- VERTS(G): Returns all vertices contained in G .
- FACES(G): Returns all faces, e.g., triangles, contained in G .

5. Embedding of B-Reps

In the following, we present the core algorithms tailored to a fast and robust numerical integration of solids described in boundary representation. These developments aim to facilitate the direct application of the divergence theorem for the assembly of the moment fitting equations. Therefore, the algorithm is designed to provide reliable results for any solid geometry model that exists as an oriented surface mesh, e.g., STereoLithography (STL). These models may represent arbitrarily complex geometries, where hundreds or even tens of thousands of triangles intersect with one background cell $\mathcal{B}_{\text{bg}} \in \mathcal{P}_{\text{bg}}$. Furthermore, the embedding workflow is robust against flaws commonly present in B-Rep models, such as duplicate geometrical entities, gaps, wrong orientations, etc. We explicitly refer to [47] for a concise definition of flaws.

As discussed in Section 2, embedded methods are characterized by a very simple discretization, which in most cases takes the form of a regular grid. This is true for classical C^0 discretizations [11,12] but also for B-Spline bases [45]. Thus, in the following, we assume a background discretization composed of hexahedral elements/cells and consider each cell $B \in \mathcal{P}_{\text{bg}}$ as an Axis-Aligned Bounding Box (AABB). This property is exploited to simplify and speed up the overall process. Rotational transformations may be applied if the background mesh \mathcal{P}_{bg} is not strictly axis-aligned. Algorithm 1 outlines the general workflow. Based on $\Gamma = \partial\Omega$, each cell B is classified as cut \mathcal{B}_{cut} , interior \mathcal{B}_{in} , or exterior \mathcal{B}_{out} , see Line 3. The developed scheme is designed to provide robust results even for severely flawed geometries. For all $\mathcal{B}_{\text{cut}} \in \mathcal{P}_{\text{bg}}$, the input STL Γ is clipped against the corresponding bounding boxes (Line 6). The resulting mesh $\Gamma_{c,c} = \mathcal{B}_{\text{cut}} \cap \Gamma$ can be used directly to impose essential boundary conditions, see Eq. (2). Subsequently, Line 8 triangulates all open faces to obtain a closed surface parameterization $\Gamma_c = \partial(\mathcal{B}_{\text{cut}} \cap \Omega)$ for the evaluation of the moments in Eq. (7). Fig. 1 visualizes this process for an exemplary cut cell.

Algorithm 1: Proposed workflow.

Input: Γ : Geometry boundary, \mathcal{P}_{bg} : Background mesh

```

1: function MAIN( $\Gamma, \mathcal{P}_{\text{bg}}$ )
2:    $\mathcal{T} \leftarrow$  BUILD AABB TREE( $\Gamma$ )
3:    $S \leftarrow$  CLASSIFY CELLS( $\mathcal{T}, \mathcal{P}_{\text{bg}}$ )
4:   for all  $\{B, s\} \in$  ZIP( $\mathcal{P}_{\text{bg}}, S$ ) do
5:     if  $s$  is  $s_{\text{cut}}$  then
6:        $\Gamma_{c,c} \leftarrow$  CLIP BOUNDARY( $B, \mathcal{T}$ )
7:        $\mathcal{T}_c \leftarrow$  BUILD AABB TREE( $\Gamma_{c,c}$ )
8:        $\Gamma_c \leftarrow$  CLOSE DOMAIN( $B, \mathcal{T}_c$ )
9:        $K \leftarrow$  COMPUTE MOMENTS( $\Gamma_c$ )
10:      ...
11:     else if  $s$  is  $s_{\text{in}}$  then
12:      ...
13:     end if
14:   end for
15: end function

```

\triangleright See: Section 5.1.
 \triangleright See: Section 5.2.
 $\triangleright B = \mathcal{B}_{\text{cut}}$.
 \triangleright See: Section 5.3.
 \triangleright See: Section 5.1.
 \triangleright See: Section 5.4.
 \triangleright See: Eq. (7).
 $\triangleright B = \mathcal{B}_{\text{in}}$.

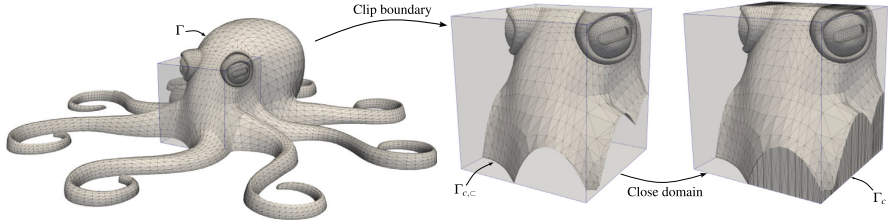


Fig. 1. Main steps required for the embedding of cut cells: (1.) Clip boundary and (2.) close domain. The depicted STL is taken from the Thing10K database (id: 69 930) [53].

The presented algorithms are implemented and provided in the open-source framework QuEsO - Quadrature for Embedded Solids [48]. QuEsO is written in C++ and has a user-friendly Python interface for easy integration with other software packages. The novel embedding technologies are coupled with the point elimination algorithm developed in [36] to compute highly efficient quadrature rules for arbitrarily complex domains. Thus, QuEsO provides a complete and automated workflow for generating analysis-ready FE models from solids in boundary representation. The resulting model is available as a list of integration points and the corresponding unfitted background discretization (e.g., classical C^0 discretization or B-Spline mesh), which can be imported into any FE solver. An interface to the open-source framework Kratos Multiphysics [54–56] is provided in [48]. Before Sections 5.2–5.5 present the key developments of this work, we discuss in Section 5.1 all required basic geometrical operations and search strategies that are inevitable for an efficient implementation. Note that the core functions of the presented algorithms are exclusively rudimentary operations for linear triangles, allowing for a simple data structure and straightforward implementation.

5.1. Search strategy and basic geometrical algorithms

To efficiently process large STL models with potentially millions of faces, a fast search algorithm is required. In the scope of this work, we use a dynamic Bounding Volume Hierarchy (BVH) [57] with AABBs as its primitives. Such AABB trees are often used in game engines for collision detection, as they allow fast intersection and distance queries against a set of geometric objects. In our application, the AABB tree \mathcal{T} is built for each triangle f in the input STL Γ (see Algorithm 1, Line 2). However, the tree \mathcal{T} does not store the triangle itself but only the AABB that bounds the triangle. These AABBs represent the leaf nodes, which are hierarchically clustered by possibly several layers of branch nodes. The idea is that each branch node is again a simple AABB that contains all its children. Therefore, potential intersection candidates between any geometrical object and the triangle mesh can be determined by testing the geometrical object against a few AABBs by walking from the root node to the respective leaves. As a result, the actual intersection test between the query object and a triangle has to be performed only for a small fraction of all $f \in \Gamma$ suggested by the conservative tree search. We use the implementation provided in [58] and equip the AABB tree with functions to quickly find all $f \in \Gamma$ that are potentially intersected by an AABB or a ray. Thus, the necessary geometrical operations are:

- Intersection query between AABB-AABB: Trivial operation.
- Intersection query between Ray-AABB: Standard algorithm in computer graphics and game development [57,59].

A subsequent exact intersection test requires the following methods:

- Intersection query between AABB-Triangle: Can be performed using the Separating Axis Theorem (SAT) [57,60].
- Intersection query between Ray-Triangle: This operation can be performed using the Möller-Trumbore (MT) algorithm [61].

Another necessary geometrical operation used in the work is:

- Clip triangle against AABB: We use a specialization of the Sutherland–Hodgman polygon clipping algorithm for axis-aligned planes, which is performed six times for each planar face of the AABB [57,62].

5.2. Classification of cells

The first step in traditional embedded boundary methods is classifying cells/elements as cut \mathcal{B}_{cut} , interior \mathcal{B}_{in} , or exterior \mathcal{B}_{out} . Generally, cells fully contained within the physical domain Ω can be integrated by standard Gauss quadrature, whereas cut cells require more sophisticated strategies, see Section 3.

Most embedded methods classify each cell based on the point membership of discrete seed points distributed throughout the cell. While for Constructive Solid Geometry (CSG) models, the necessary inside/outside queries for each point are inherently given by the model description, this task is much more involved for solids explicitly defined in boundary representation. Classically, ray tracing methods are employed where the number of intersections of each ray with the geometrical boundary decides the classification of

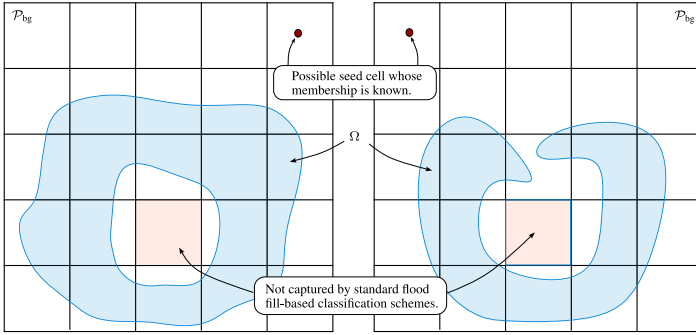


Fig. 2. Limitations of standard flood fill-based cell classification schemes.

the examined point [63]. This approach works robustly on B-Reps, which are an orientable 2-manifold without a boundary. Such models are often loosely termed valid, watertight, or closed B-Reps. Nevertheless, ray-tracing approaches are unfavorable for flawed geometries, which only seemingly describe a solid and are non-watertight, wrongly oriented, or contain internal boundaries. For example, a small gap represents such an internal boundary for which the point membership can no longer be uniquely determined using a single ray probe. The number of intersections of the ray with the domain’s boundary now depends on the direction of the ray. Moreover, potential misclassifications are not limited to the vicinity of the geometrical flaw but can occur in the entire domain. This dramatically affects the robustness of the simulation workflow. In fact, a single cell outside the physical domain, which is falsely classified as inside, can lead to an infeasible FE discretization. Majority voting based on multiple rays is a common approach to significantly reduce the likelihood of such misclassifications. However, this entails expensive schemes whose cost scales linearly with the number of ray tests carried out, which, in turn, is directly determined by the desired degree of robustness. We note that an inaccurate integration of cut cells due to geometric flaws is less severe because it is a local error that does not necessarily lead to an unsolvable system. Therefore, a robust cell classification scheme is the basis for a universally applicable embedding workflow that can handle erroneous geometries.

A major step towards this goal is presented in [47], employing a flood fill algorithm. Generally, given a multi-dimensional array, flood fill algorithms can determine all objects that are connected to a given starting object and match a certain attribute. Typical applications are image processing tools, which change the color of a similarly-colored area of adjacent pixels. For more information on flood fill algorithms, the interested reader is referred to [64]. In [47], flood filling is used to group connected cells that are located inside or outside the domain. Since cells are classified as a group rather than individually, misclassification of cells that are not in immediate proximity to geometric flaws is avoided. Technically, flood fill-based schemes combined with a triangle-cell intersection query can provide robust cell classifications without a single cast ray. However, this requires flooding to be initiated from a seed cell whose membership must be known. Since this information decides the classification of the entire group, it is imperative that it is correct. Therefore, flooding is usually started from a cell outside the domain, e.g., a corner cell of \mathcal{P}_{bg} , while cut cells serve as delimitations. Consequently, all unvisited cells are assumed to be inside. This approach, however, prevents the detection of cavities or regions enclosed by a concave domain, as illustrated in Fig. 2.

The presented algorithm is inspired by the flood fill-based classification scheme proposed in [47] and is further developed in such a way that flooding can be initiated from any seed cell without prior knowledge of its membership. We aim to combine the advantages of flood filling and ray casting to achieve an efficient and generally applicable cell classification algorithm. The novel classification approach has the following characteristics:

- Interior and exterior cells are first grouped and then classified, whereby multiple groups are allowed.
- Local ray tracing, which only operates on a very small section of the boundary Γ , is performed to automatically classify each group.
- An incorrect classification of cells, which are not in immediate proximity to the geometric flaw, is excluded.
- Robust results for any kind of geometrical features, including cavities, as depicted in Fig. 2.
- Wall-clock times scale linearly with the number of cells in \mathcal{P}_{bg} .

Algorithm 2 outlines the necessary operations. First, Lines 4–10 identify all cut cells $\mathcal{B}_{cut} \in \mathcal{P}_{bg}$ using Algorithm 3. The Separating Axis Theorem employed in our implementation allows the detection of all possible intersection patterns, including arbitrarily small cuts, see Section 5.1. All potentially intersected triangles \mathcal{F}_{pot} are obtained from an AABB tree query, see Algorithm 3, Line 2. Thus, the exact triangle-AABB intersection test is only performed for a minimal set of triangles, decreasing the execution time drastically. If at least one triangle is intersected, the cell is classified as cut $s = s_{cut}$.

In the following, the cut cells are utilized as a watertight boundary, delimiting interior and exterior cells. Note that the cut cells will always provide a watertight delimitation as long as there is no gap in the model large enough for the outer and inner cells to

Algorithm 2: Classify cells.

Input: \mathcal{T} : AABB tree (built on Γ), Background mesh \mathcal{P}_{bg}

```

1: function CLASSIFYCELLS( $\mathcal{T}$ ,  $\mathcal{P}_{\text{bg}}$ )
2:    $S \leftarrow \text{INITIALIZESET}(\text{SIZE}(\mathcal{P}_{\text{bg}}), s_{\text{out}})$ ;  $i \leftarrow 0$                                 ▷ Initialize all cells as outside.
3:    $V \leftarrow \text{INITIALIZESET}(\text{SIZE}(\mathcal{P}_{\text{bg}}), \text{FALSE})$                                 ▷ Initialize all cells as unvisited.
4:   for all  $B \in \mathcal{P}_{\text{bg}}$  do                                                            ▷ Note that  $B = \mathcal{P}_{\text{bg}}[i]$ .
5:     if  $\text{ISCUT}(B, \mathcal{T})$  then                                                       ▷ See: Algorithm 3.
6:        $S[i] \leftarrow s_{\text{cut}}$                                                        ▷ Mark as cut.
7:        $V[i] \leftarrow \text{TRUE}$                                                        ▷ Mark as visited.
8:     end if
9:      $i \leftarrow i + 1$ 
10:  end for
11:  while  $\exists V^i \in V : (V^i = \text{FALSE})$  do                                         ▷ Repeat until all cells are visited.
12:     $i \leftarrow i : (V^i = \text{FALSE})$                                              ▷ Get the index of an unvisited cell.
13:     $S \leftarrow \text{FLOODFILL}(i, S, V, \mathcal{P}_{\text{bg}})$                                        ▷ See: Fig. 3.
14:  end while
15:  return  $S$ 
16: end function

```

Algorithm 3: Detect whether a cell B is cut or not.

Input: B : AABB, \mathcal{T} : AABB tree (built on Γ)

```

1: function ISCUT( $B$ ,  $\mathcal{T}$ )
2:    $\mathcal{F}_{\text{pot}} \leftarrow \text{QUERYAABBTREE}(B, \mathcal{T})$                                        ▷ Get potential intersections.
3:   for all  $f \in \mathcal{F}_{\text{pot}}$  do
4:     if  $\text{DOINTERSECT}(B, f)$  then                                             ▷ Test for actual intersection, see Section 5.1.
5:       return  $\text{TRUE}$ 
6:     end if
7:   end for
8:   return  $\text{FALSE}$ 
9: end function

```

be connected by an uncut cell. This is not a severe limitation, however, since gaps are usually due to mathematical inaccuracies in CAD and are, therefore, rarely large enough for an entire uncut cell to fit through. Once all cut cells are identified, Algorithm 2 repeatedly invokes the flood fill algorithm until all cells are visited. Fig. 3 visualizes this process. Given any unvisited seed cell $B \neq B_{\text{cut}}$, the currently active group is extended until all neighbors are either part of the group or cut. Note that we employ a 6-connected or 6-neighbor version in three dimensions [64]. This means neighbors are only considered if they share a face and not only a vertex. For the algorithm to be generic, the found group must be automatically classified as inside or outside. To this end, we detect all cells that are part of the active group \mathcal{G} and have a neighboring cut cell B_{cut} . The main idea is to examine each of these boundary cells locally, combine the information, and base the final classification of the entire group on a majority vote. The boundary cells are rated according to the location of their center point x relative to the boundary Γ . Algorithm 4 uses ray tracing for the respective point membership classification. However, this query operates only on a very small section of Γ , which guarantees a local classification of each boundary cell $B \in \mathcal{G}$ and drastically reduces the potential intersection candidates for each ray r . To be more precise, given a cell $B \in \mathcal{G}$, the boundary Γ is clipped against all its neighboring cut cells B_{cut} to obtain $\Gamma_{c,c}$, see Fig. 3. For a detailed discussion of the respective clipping algorithm, we refer to the next subsection. Before $\Gamma_{c,c}$ is passed to Algorithm 4, a local AABB tree \mathcal{T}_c is built to allow a quick search for possible intersection candidates.

Note that the classical approach, where the number of intersections between a ray and the boundary determines the point's membership, is only valid for closed meshes. Since $\Gamma_{c,c}$ is an open mesh, Algorithm 4 ($\text{ISINSIDE}()$) classifies the query point x based on the orientation of the closest triangle $f \in \Gamma_{c,c}$ that is intersected by the ray r . For this purpose, all intersected triangles are detected and compared based on their distance to x . To make sure r does not miss $\Gamma_{c,c}$ entirely, we initially shoot toward the center of the first triangle $f = \text{FACES}(\Gamma_{c,c})^0$, see Lines 5–6. Subsequently, all other triangles intersecting with r are found. The orientation of the nearest triangle determines whether $x \in \Omega$ or $x \notin \Omega$. Ambiguous results caused by rays falling on the vertex or edge between two or multiple neighboring triangles are marked as unsuccessful. Triangles that are found to be numerically parallel to r are also not considered. The Möller–Trumbore algorithm recognizes both scenarios. If the current ray is unsuccessful, a new ray towards the center of the next triangle is cast until only unequivocal intersection states are found. Furthermore, $a_{\text{count}}^{\text{min}}$ allows us to base the decision about the membership of x on multiple rays. If $a_{\text{count}}^{\text{min}} > 1$, a majority vote decides whether x is classified as inside or outside, see Algorithm 4, Line 10 and Line 14. In this case, multiple rays are shot toward the clipped boundaries $\Gamma_{c,c}$ for each boundary

Algorithm 4: Point membership classification for open meshes.

Input: x : Point in 3D, \mathcal{T}_c : AABB tree (built on clipped mesh $\Gamma_{c,c}$)

```

1: function IsINSIDE( $x$ ,  $\mathcal{T}_c$ )
2:    $i \leftarrow 0$ ;  $a_{\text{count}} \leftarrow 0$ ;  $b_{\text{count}} \leftarrow 0$ 
3:    $\mathcal{F}_{\text{clip}} \leftarrow \text{FACES}(\mathcal{T}_c)$  ▷ Get list of triangles of clipped mesh  $\Gamma_{c,c}$ .
4:   while  $a_{\text{count}} < a_{\text{count}}^{\min} \wedge i < \text{SIZE}(\mathcal{F}_{\text{clip}})$  do
5:      $x_c \leftarrow \text{CENTER}(\mathcal{F}_{\text{clip}}[i])$ 
6:      $r \leftarrow \text{RAY}(x, x_c - x)$  ▷ Cast ray through center of  $\mathcal{F}_{\text{clip}}[i]$ .
7:      $\{a, b\} \leftarrow \text{IsINSIDE}(x, \mathcal{T}_c, r)$  ▷  $a = \text{TRUE} \mapsto$  successful test.  $b = \text{TRUE} \mapsto x$  is inside.
8:     if  $a$  then
9:        $a_{\text{count}} \leftarrow a_{\text{count}} + 1$ 
10:       $(b) ? (b_{\text{count}} \leftarrow b_{\text{count}} + 1) : (b_{\text{count}} \leftarrow b_{\text{count}} - 1)$ 
11:    end if
12:     $i \leftarrow i + 1$ 
13:  end while
14:  return  $b_{\text{count}} > 0$ 
15: end function

```

Input: x : Point, \mathcal{T}_c : AABB tree (built on clipped mesh $\Gamma_{c,c}$), r : Ray

```

16: function IsINSIDE( $x$ ,  $\mathcal{T}_c$ ,  $r$ )
17:    $\mathcal{F}_{\text{pot}} \leftarrow \text{QUERYAABBTREE}(r, \mathcal{T}_c)$  ▷ Get potential intersections, see Section 5.1.
18:    $d_{\min} \leftarrow \infty$ ;  $b \leftarrow \text{FALSE}$ 
19:   for all  $f \in \mathcal{F}_{\text{pot}}$  do
20:     if  $\text{DOINTERSECT}(r, f) \wedge \neg \text{AREPARALLEL}(r, f)$  then ▷ Test for actual intersection, see Section 5.1.
21:       if  $\text{ONTriangleEDGE}(r, f)$  then
22:         return  $\{\text{FALSE}, \text{FALSE}\}$  ▷ Cast new ray.
23:       end if
24:        $d \leftarrow \text{DISTANCE}(r, f)$ 
25:       if  $d < d_{\min}$  then
26:          $b \leftarrow \text{ISBACKFACING}(r, f)$ ;  $d_{\min} \leftarrow d$ 
27:       end if
28:     end if
29:   end for
30:   return  $\{\text{TRUE}, b\}$ 
31: end function

```

cell in the active group \mathcal{G} . Fig. 3 shows an example of two rays per neighboring cut cell ($a_{\text{count}}^{\min} = 2$). Note that the number of rays that can be shot is limited by the number of triangles intersected by each cell. Once all boundary cells are locally classified based on a majority vote of all ray tracing tests, the majority of all cells decides about the classification of the entire group. As discussed previously, we note that majority voting will also increase the robustness of classical ray tracing, where multiple rays are cast for each cell individually. The advantages of the proposed scheme over such approaches are listed in the following.

- Rays are only intersected with the clipped mesh $\Gamma_{c,c}$, which covers a small fraction of the global mesh Γ .
- Rays are only cast for boundary cells, see Fig. 3. When the background grid is refined, many cells are classified without a single ray shot.
- Cells are not individually classified but profit from the information available in the entire group, which might contain thousands or even millions of cells.

Embedded boundary methods usually rely on cell-wise computations, allowing efficient, embarrassingly parallel implementations. Since standard flood fill schemes are sequential algorithms, they can quickly become the bottleneck in a parallel execution. For an efficient implementation, we partition the background grid \mathcal{P}_{bg} into n_{procs} stripes, where n_{procs} is the number of available processes. This allows each partition to be flood filled in parallel. Subsequently, the groups from all n_{procs} partitions are merged and finally classified. The partitioning is applied along the axis of \mathcal{P}_{bg} with the most number of elements. Fig. 4 illustrates the main steps. Section 6.2 applies the presented classification scheme to 4948 valid and flawed STLs with varying complexity.

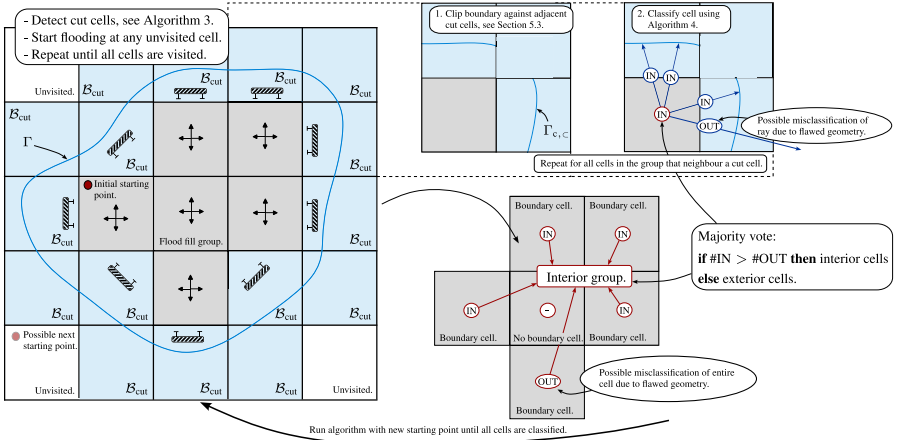


Fig. 3. Main steps of the generalized flood fill-based classification scheme for flawed geometries.

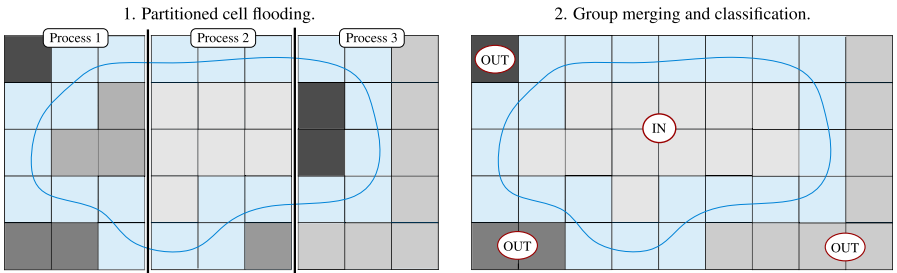


Fig. 4. Partitioned flood fill-based cell classification for parallel implementation. As an example, the algorithm is shown for three available processes $n_{\text{process}} = 3$.

5.3. Clip boundary

In the following, we discuss the necessary mesh clipping algorithm to clip the boundary Γ against any $\mathcal{B}_{cut} \in \mathcal{P}_{bg}$, see Fig. 1. The resulting mesh is denoted as $\Gamma_{c,c}$. A main feature of the presented embedding scheme is that the clipping algorithm is utilized to partition the global mesh Γ into many local sub-meshes $\Gamma_{c,c}$. All subsequent operations in Section 5.4 are designed to rely exclusively on $\Gamma_{c,c}$, drastically reducing the computational overhead. Furthermore, $\Gamma_{c,c}$ can also be directly used to evaluate surface integrals for imposing essential boundary conditions (see Eq. (2)).

Algorithm 5 outlines the clipping workflow, taking the bounding box \mathcal{B} and the AABB tree \mathcal{T} built on Γ as input arguments. Line 3 determines all intersected triangles using Algorithm 6, which again employs the AABB tree \mathcal{T} to accelerate this process. The query for the intersection between a triangle and the bounding box (Algorithm 6, Line 5) is hence only performed for the potential candidates suggested by \mathcal{T} . Once all intersected triangles are found, Algorithm 5 adds those that are fully contained within \mathcal{B} to the output mesh \mathcal{F}_{out} . Since the remaining triangles intersect \mathcal{B} but do not lie entirely within \mathcal{B} , they must cross its boundaries. Therefore, they are clipped by the six half-spaces that enclose the bounding box. The Sutherland–Hodgman polygon clipping algorithm introduced in Section 5.1 performs the respective operations. Note that a triangle clipped by a bounding box results in a polygon with ≤ 9 vertices. To get a pure triangle mesh again, we triangulate the domain by introducing an additional vertex at the polygon's center.

Remark. The clipped mesh $\Gamma_{c,c}$ of $\mathcal{B}_{cut} \in \mathcal{P}_{bg}$ is the basis for many operations. Thus, Algorithm 5 is listed in several places in this work for a better understanding, although it is executed only once per cell.

Algorithm 5: Clip the boundary Γ against a bounding box B .

Input: B : Bounding box, \mathcal{T} : AABB tree (built on Γ)

```

1: function CLIPBOUNDARY( $B, \mathcal{T}$ )
2:    $\mathcal{F}_{\text{out}} \leftarrow \emptyset$  ▷ Initialize empty list of triangles.
3:    $\mathcal{F}_{\text{int}} \leftarrow \text{INTERSECTEDTRIANGLES}(B, \mathcal{T})$  ▷ See: Algorithm 6.
4:   for all  $f \in \mathcal{F}_{\text{int}}$  do
5:     if ISINSIDE( $f, B$ ) then
6:        $\mathcal{F}_{\text{out}} \leftarrow \text{APPEND}(\mathcal{F}_{\text{out}}, f)$ 
7:     else
8:        $\mathcal{F}_{\text{clip}} \leftarrow \text{CLIP}(f, B)$  ▷ See: Sutherland-Hodgman algorithm discussed in Section 5.1.
9:        $\mathcal{F}_{\text{out}} \leftarrow \text{APPEND}(\mathcal{F}_{\text{out}}, \mathcal{F}_{\text{clip}})$ 
10:    end if
11:  end for
12:  return  $\mathcal{F}_{\text{out}}$ 
13: end function

```

Note: ISINSIDE(f, B) returns TRUE, if all vertices of f are inside B .

Algorithm 6: Find all intersected triangles.

Input: B : Bounding box, \mathcal{T} : AABB tree (built on Γ)

```

1: function INTERSECTEDTRIANGLES( $B, \mathcal{T}$ )
2:    $\mathcal{F}_{\text{out}} \leftarrow \emptyset$ 
3:    $\mathcal{F}_{\text{pot}} \leftarrow \text{QUERYAABBTREE}(B, \mathcal{T})$  ▷ Get potential intersections, see: Section 5.1.
4:   for all  $f \in \mathcal{F}_{\text{pot}}$  do
5:     if DOINTERSECT( $f, B$ ) then ▷ Test for actual intersection, see: Section 5.1.
6:        $\mathcal{F}_{\text{out}} \leftarrow \text{APPEND}(\mathcal{F}_{\text{out}}, f)$ 
7:     end if
8:   end for
9:   return  $\mathcal{F}_{\text{out}}$ 
10: end function

```

5.4. Close domain

Evaluating the boundary integrals in Eq. (7) requires a closed surface parameterization of each cut domain. Therefore, this section presents a fast and robust triangulation algorithm that allows the direct assembly of the moment fitting equations from B-Rep models. Given the bounding box B and the clipped boundary $\Gamma_{c,c}$ obtained in Section 5.3, the proposed algorithm outputs a closed boundary mesh Γ_c representing the cut domain, as depicted in Fig. 1. To speed up all necessary operations, $\Gamma_{c,c}$ is stored in a local AABB tree \mathcal{T}_c , which is built in Algorithm 1, Line 7.

Algorithm 7 summarizes the essential steps for generating a closed surface mesh. The main routines COLLECTEDGESONPLANE(), CATEGORIZEEDGES(), CLOSEUPPERPOLYLINE(), SPLITEDGES(), and TRIANGULATEOPENFACES() are performed successively for all six planar faces P of the bounding box B . Each step is visualized in Figs. 5 and 6 and described individually in the following subsections.

5.4.1. Collect and categorize edges on the plane

As depicted in Fig. 5, we collect all edges on the current planar face $p \in P$ and map them to a local coordinate system $v = \{\xi, \eta\} \in \mathbb{R}^2$. Subsequently, all edges \mathcal{E} are classified according to their normals as upper edges \mathcal{E}_{up} , lower edges \mathcal{E}_{low} , or vertical edges $\mathcal{E}_{\text{vert}}$, such that

$$\mathcal{E} = \mathcal{E}_{\text{up}} \cup \mathcal{E}_{\text{low}} \cup \mathcal{E}_{\text{vert}}, \quad (8)$$

as shown in Fig. 6(b). Note that edges located on one of the six planar faces can be directly marked in the Sutherland–Hodgman polygon clipping algorithm since these are the newly introduced ones. Analogously, the required normal vector n_e can be retrieved from the respective intersected triangle. For valid faceted B-Reps, all triangles are oriented so that their normals point outward.

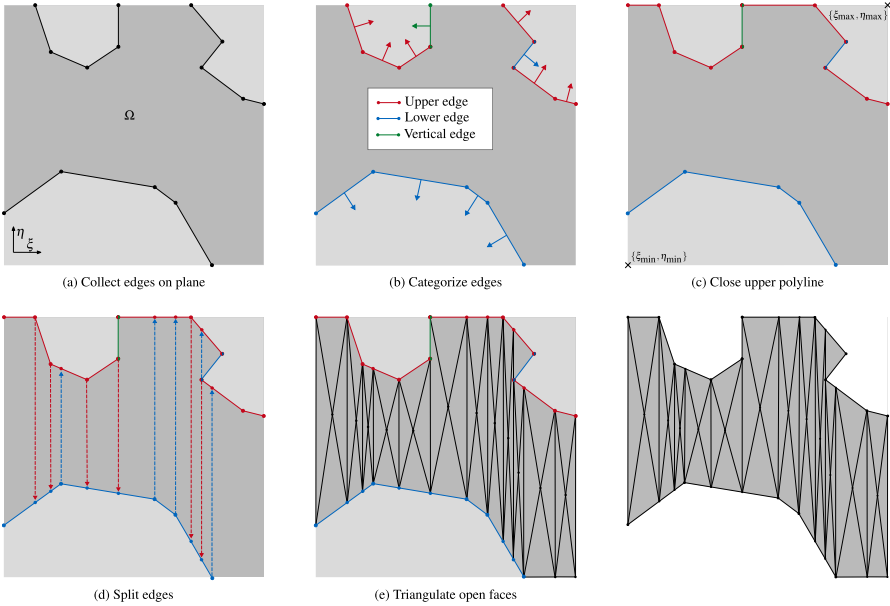


Fig. 6. Illustration of main steps in Algorithm 7: CLOSEDOMAIN().

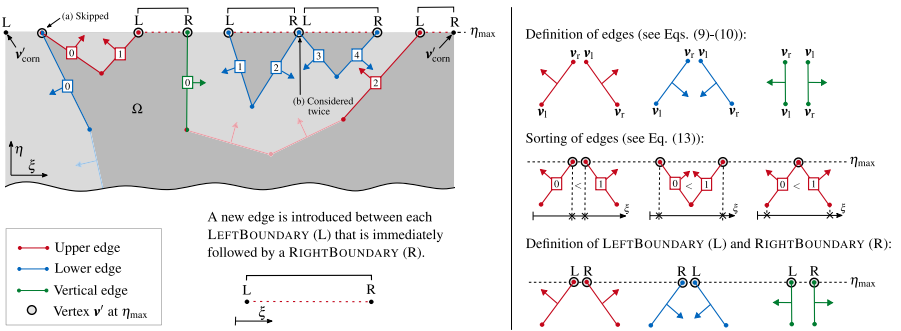


Fig. 7. Illustration of main steps in Algorithm 8: CLOSEUPPERPOLYLINE().

Subsequently, we mark each $v^j \in \mathcal{V}$ as LEFTBOUNDARY or RIGHTBOUNDARY and introduce a new edge ϵ between every $\mathcal{V}^{j,i} = \{v^j, \text{LEFTBOUNDARY}\}$ that is immediately followed by $\mathcal{V}^{j,i+1} = \{v^j, \text{RIGHTBOUNDARY}\}$, see Fig. 7. Thereby, the orientation of the connected edges decides the classification into LEFTBOUNDARY or RIGHTBOUNDARY. A detailed discussion of this procedure is provided below.

Given the classified edges \mathcal{E}_{up} , \mathcal{E}_{low} , and $\mathcal{E}_{\text{vert}}$, the corresponding face p of \mathcal{B} , and the local AABB tree \mathcal{T}_c built on the clipped mesh $\Gamma_{c,c}$, Algorithm 8 performs the necessary operations. Firstly, Lines 4-5 filter all edges $\mathcal{E}'_{\text{up}} \subseteq \mathcal{E}_{\text{up}}$, $\mathcal{E}'_{\text{low}} \subseteq \mathcal{E}_{\text{low}}$, and $\mathcal{E}'_{\text{vert}} \subseteq \mathcal{E}_{\text{vert}}$ that contain exactly one vertex $v^j = \{\xi, \eta = \eta_{\max}\}$ and sort each subset in ascending order from left to right such that

$$\forall (0 \leq i < n - 1) : \begin{cases} \xi \in (v = v^j) \in \mathcal{E}'_{\beta}{}^j < \xi \in (v = v^j) \in \mathcal{E}'_{\beta}{}^{j+1}, & \text{if } v^j \in \mathcal{E}'_{\beta}{}^j \neq v^j \in \mathcal{E}'_{\beta}{}^{j+1}, \\ \xi \in (v \neq v^j) \in \mathcal{E}'_{\beta}{}^j < \xi \in (v \neq v^j) \in \mathcal{E}'_{\beta}{}^{j+1}, & \text{else,} \end{cases} \quad (13)$$

Algorithm 8: Close upper polyline.

Input: \mathcal{E}_{up} : Upper edges, \mathcal{E}_{low} : Lower edges, $\mathcal{E}_{\text{vert}}$: Vertical edges, p : Face of B ,
 \mathcal{T}_c : AABB tree (built on clipped mesh $\Gamma_{c,C}$)

- 1: **function** CLOSEUPPERPOLYLINE(\mathcal{E}_{up} , \mathcal{E}_{low} , $\mathcal{E}_{\text{vert}}$, p , \mathcal{T}_c)
- 2: $\mathcal{V}' \leftarrow \emptyset$; $\mathcal{V}'_{\text{corn}} \leftarrow \emptyset$;
- 3: $\{\{\xi_{\text{min}}, \eta_{\text{min}}\}, \{\xi_{\text{max}}, \eta_{\text{max}}\}\} \leftarrow \text{BOUNDINGBOX}(p)$ ▷ See: Fig. 6 (c).
- 4: $\mathcal{E}'_{\text{up}} \leftarrow \text{EDGESATÉTA}\text{MAX}(\mathcal{E}_{\text{up}})$ ▷ Find all edges that have exactly one vertex at $\eta = \eta_{\text{max}}$.
- 5: $\mathcal{E}'_{\text{up}} \leftarrow \text{SORT}(\mathcal{E}'_{\text{up}})$ ▷ Sort in ascending order from left to right, see Fig. 7 and Eq. (13).
- 6: $i_{\text{up}} \leftarrow 0$; $n_{\text{up}} \leftarrow \text{SIZE}(\mathcal{E}'_{\text{up}})$ ▷ Repeat operation (Lines 4-6) for \mathcal{E}_{low} and $\mathcal{E}_{\text{vert}}$.
- 7: **while** $i_{\text{up}} < n_{\text{up}} \vee i_{\text{low}} < n_{\text{low}} \vee i_{\text{vert}} < n_{\text{vert}}$ **do**
- 8: $d_{\text{up}} \leftarrow \infty$; $d_{\text{low}} \leftarrow \infty$; $d_{\text{vert}} \leftarrow \infty$
- 9: **if** $i_{\text{up}} < n_{\text{up}}$ **then**
- 10: $v'_{\text{up}} \leftarrow \text{VERTEXATÉTA}\text{MAX}(\mathcal{E}'_{\text{up}}[i_{\text{up}}])$ ▷ Get vertex at $\eta = \eta_{\text{max}}$.
- 11: $d_{\text{up}} \leftarrow v'_{\text{up}}[0] - \xi_{\text{min}}$ ▷ Measure distance to left boundary $\xi = \xi_{\text{min}}$.
- 12: **end if** ▷ Repeat operation (Lines 9-12) for v'_{low} and v'_{vert} .
- 13: $d_{\text{min}} \leftarrow \text{MIN}(d_{\text{up}}, d_{\text{low}}, d_{\text{vert}})$
- 14: $q_{\text{up}} \leftarrow (d_{\text{up}} = d_{\text{min}})$ ▷ Repeat operation (Line 14) for q_{low} and q_{vert} .
- 15: **if** $q_{\text{up}} + q_{\text{low}} + q_{\text{vert}} > 1$ **then**
- 16: $i_{\text{up}} \leftarrow i_{\text{up}} + q_{\text{up}}$; $i_{\text{low}} \leftarrow i_{\text{low}} + q_{\text{low}}$; $i_{\text{vert}} \leftarrow i_{\text{vert}} + q_{\text{vert}}$ ▷ Skip edge, if q_{up} , q_{low} , or q_{vert} is TRUE.
- 17: **else if** q_{up} **then**
- 18: $\mathcal{V}' \leftarrow \text{CLASSIFYBOUNDARYVERTEX}(\mathcal{V}', v'_{\text{up}}, \mathcal{E}'_{\text{up}}[i_{\text{up}}], \text{UPPEREDGE})$ ▷ See: Algorithm 9.
- 19: $i_{\text{up}} \leftarrow i_{\text{up}} + 1$
- 20: **else if** q_{low} **then**
- 21: $\mathcal{V}' \leftarrow \text{CLASSIFYBOUNDARYVERTEX}(\mathcal{V}', v'_{\text{low}}, \mathcal{E}'_{\text{low}}[i_{\text{low}}], \text{LOWEREDGE})$ ▷ See: Algorithm 9.
- 22: $i_{\text{low}} \leftarrow i_{\text{low}} + 1$
- 23: **else if** q_{vert} **then**
- 24: $\mathcal{V}' \leftarrow \text{CLASSIFYBOUNDARYVERTEX}(\mathcal{V}', v'_{\text{vert}}, \mathcal{E}'_{\text{vert}}[i_{\text{vert}}], \text{VERTICALEGE})$ ▷ See: Algorithm 9.
- 25: $i_{\text{vert}} \leftarrow i_{\text{vert}} + 1$
- 26: **end if**
- 27: **end while**
- 28: **if** $\{\xi_{\text{min}}, \eta_{\text{max}}\} \notin (\mathcal{E}_{\text{up}} \cup \mathcal{E}_{\text{low}} \cup \mathcal{E}_{\text{vert}})$ **then**
- 29: $\mathcal{V}'_{\text{corn}} \leftarrow \text{APPEND}(\mathcal{V}'_{\text{corn}}, \{\{\xi_{\text{min}}, \eta_{\text{max}}\}, \text{LEFTBOUNDARY}\})$ ▷ Mark corner vertex as LEFTBOUNDARY.
- 30: **end if**
- 31: **if** $\{\xi_{\text{max}}, \eta_{\text{max}}\} \notin (\mathcal{E}_{\text{up}} \cup \mathcal{E}_{\text{low}} \cup \mathcal{E}_{\text{vert}})$ **then**
- 32: $\mathcal{V}'_{\text{corn}} \leftarrow \text{APPEND}(\mathcal{V}'_{\text{corn}}, \{\{\xi_{\text{max}}, \eta_{\text{max}}\}, \text{RIGHTBOUNDARY}\})$ ▷ Mark corner vertex as RIGHTBOUNDARY.
- 33: **end if**
- 34: $\mathcal{E}_{\text{up}} \leftarrow \text{INSERTEDGES}(\mathcal{E}_{\text{up}}, \mathcal{V}', \mathcal{V}'_{\text{corn}}, p, \mathcal{T}_c)$ ▷ See: Algorithm 10.
- 35: **return** \mathcal{E}_{up}
- 36: **end function**

with $\beta \in \{\text{up}, \text{low}, \text{vert}\}$. Consequently, two edges $\mathcal{E}'_{\beta}{}^i$ and $\mathcal{E}'_{\beta}{}^{i+1}$ are generally sorted by comparing their vertices $v' \in \mathcal{E}'_{\beta}{}^i$ and $v' \in \mathcal{E}'_{\beta}{}^{i+1}$, as shown in Fig. 7. In the particular case where $\mathcal{E}'_{\beta}{}^i$ and $\mathcal{E}'_{\beta}{}^{i+1}$ share a vertex at η_{max} , the two opposite vertices are used. After the initialization, Algorithm 8 loops over all $\epsilon_{\text{up}} \in \mathcal{E}'_{\text{up}}$, $\epsilon_{\text{low}} \in \mathcal{E}'_{\text{low}}$, and $\epsilon_{\text{vert}} \in \mathcal{E}'_{\text{vert}}$, see Line 7. During each iteration, the vertices v'_{up} , v'_{low} , and v'_{vert} are obtained from the edges ϵ_{up} , ϵ_{low} , and ϵ_{vert} . To ensure the ordering defined in Eq. (12), the vertex v' with the shortest distance to the left boundary $\xi = \xi_{\text{min}}$ is always processed first. In the special case where v' is connected to two differently oriented edges, e.g., $v'_{\text{up}} \in \epsilon_{\text{up}} = v'_{\text{low}} \in \epsilon_{\text{low}}$ (see example (a) in Fig. 7), both edges are skipped in Algorithm 8, Line 16. In all other scenarios, Algorithm 9 (CLASSIFYBOUNDARYVERTEX) classifies v' as LEFTBOUNDARY or RIGHTBOUNDARY and adds it to the ordered set \mathcal{V}' . The classification obeys the following simple rules. If $\epsilon = \epsilon_{\text{up}} \vee \epsilon = \epsilon_{\text{vert}}$, $(v_l = v') \mapsto \text{RIGHTBOUNDARY}$ (Line 5) and $(v_r = v') \mapsto \text{LEFTBOUNDARY}$ (Line 7). On the other hand, if $\epsilon = \epsilon_{\text{low}}$, $(v_l = v') \mapsto \text{LEFTBOUNDARY}$ (Line 11) and $(v_r = v') \mapsto \text{RIGHTBOUNDARY}$ (Line 13), also see Fig. 7. Note that v' may also be connected to two edges with the same orientation (see example (b) in Fig. 7). In that case, v' is added twice to \mathcal{V}' , whereby the ordering defined in Eq. (13) guarantees that v' is first added by the edge to its left and then by the edge to its right.

Algorithm 9: Classify boundary vertex.

Input: \mathcal{V}' : List of vertices, v' : Vertex at $\eta = \eta_{\max}$, ϵ : Edge, t_e : Edge type

```

1: function CLASSIFYBOUNDARYVERTEX( $\mathcal{V}'$ ,  $v'$ ,  $\epsilon$ ,  $t_e$ )
2:    $\{v_l, v_r\} \leftarrow \epsilon$ 
3:   if  $t_e = \text{UPPEREDGE} \vee t_e = \text{VERTICALEdge}$  then
4:     if  $v_l = v'$  then
5:        $\mathcal{V}' \leftarrow \text{APPEND}(\mathcal{V}', \{v', \text{RIGHTBOUNDARY}\})$ 
6:     else if  $v_r = v'$  then
7:        $\mathcal{V}' \leftarrow \text{APPEND}(\mathcal{V}', \{v', \text{LEFTBOUNDARY}\})$ 
8:     end if
9:   else if  $t_e = \text{LOWEREDGE}$  then
10:    if  $v_l = v'$  then
11:       $\mathcal{V}' \leftarrow \text{APPEND}(\mathcal{V}', \{v', \text{LEFTBOUNDARY}\})$ 
12:    else if  $v_r = v'$  then
13:       $\mathcal{V}' \leftarrow \text{APPEND}(\mathcal{V}', \{v', \text{RIGHTBOUNDARY}\})$ 
14:    end if
15:  end if
16:  return  $\mathcal{V}'$ 
17: end function

```

Before \mathcal{V}' is used to introduce new edges to close the upper polyline, as shown in Fig. 7, the two corner points $v'_{\text{corn}} = \{\xi_{\min}, \eta_{\max}\} \in \mathcal{V}'_{\text{corn}}$ (Algorithm 8, Line 29), and $v'_{\text{corn}} = \{\xi_{\max}, \eta_{\max}\} \in \mathcal{V}'_{\text{corn}}$ (Algorithm 8, Line 32) must also be considered. Note that the corner points are only introduced when not already contained in \mathcal{E}_{up} , \mathcal{E}_{low} , or $\mathcal{E}_{\text{vert}}$.

Algorithm 10: Insert Edges.

Input: \mathcal{E}_{up} : List of edges, \mathcal{V}' : List of vertices, $\mathcal{V}'_{\text{corn}}$: List of corner vertices, p : Planar face, \mathcal{T}_c : AABB tree (built on clipped mesh $\Gamma_{c,c}$)

```

1: function INSERTEDGES( $\mathcal{E}_{\text{up}}$ ,  $\mathcal{V}'$ ,  $\mathcal{V}'_{\text{corn}}$ ,  $p$ ,  $\mathcal{T}_c$ )
2:   if  $\mathcal{V}' \neq \emptyset$  then
3:      $\mathcal{V}' \leftarrow \text{APPEND}(\mathcal{V}'_{\text{corn}}[0], \mathcal{V}', \mathcal{V}'_{\text{corn}}[1])$ 
4:      $n \leftarrow \text{SIZE}(\mathcal{V}')$ 
5:     for all  $i \in \text{RANGE}(0, n - 1)$  do
6:       if  $\text{LEFTBOUNDARY} \in \mathcal{V}'[i] \wedge \text{RIGHTBOUNDARY} \in \mathcal{V}'[i + 1]$  then
7:          $\mathcal{E}_{\text{up}} \leftarrow \text{APPEND}(\mathcal{E}_{\text{up}}, \{v' \in \mathcal{V}'[i], v' \in \mathcal{V}'[i + 1]\})$ 
8:       end if
9:     end for
10:  else
11:     $v_c \leftarrow 0.5 (v'_{\text{corn}} \in \mathcal{V}'_{\text{corn}}[0] + v'_{\text{corn}} \in \mathcal{V}'_{\text{corn}}[1])$  ▷ Compute center.
12:     $x_c \leftarrow \text{MAPTO3D}(v_c, \text{PLANEOFFSET}(p))$  ▷ Map point to 3D space:  $v_c \in \mathbb{R}^2 \mapsto x_c \in \mathbb{R}^3$ .
13:    if  $\text{ISINSIDE}(x_c, \mathcal{T}_c)$  then ▷ See: Algorithm 4.
14:       $\mathcal{E}_{\text{up}} \leftarrow \text{APPEND}(\mathcal{E}_{\text{up}}, \{v'_{\text{corn}} \in \mathcal{V}'_{\text{corn}}[0], v'_{\text{corn}} \in \mathcal{V}'_{\text{corn}}[1]\})$ 
15:    end if
16:  end if
17:  return  $\mathcal{E}_{\text{up}}$ 
18: end function

```

Subsequently, both \mathcal{V}' and $\mathcal{V}'_{\text{corn}}$ are passed to Algorithm 10 (INSERTEDGES). The additional input parameters are the edges \mathcal{E}_{up} , the planar face p , and the local AABB tree \mathcal{T}_c built on the clipped mesh $\Gamma_{c,c}$. If $\mathcal{V}' \neq \emptyset$, a new edge is introduced between every LEFTBOUNDARY that is immediately followed by a RIGHTBOUNDARY, see Algorithm 10, Line 7. Lines 11–15 account for the case where no vertex v' at $\eta = \eta_{\max}$ exists. If $\mathcal{V}' = \emptyset$, the only two vertices available are v'_{corn}^0 and v'_{corn}^1 . However, since v'_{corn}^0 and v'_{corn}^1 are not connected to any edge, no information about the cut domain is provided. Therefore, the question of whether the edge between v'_{corn}^0 and v'_{corn}^1 is part of Ω is still to be answered. To this end, we test if the midpoint v_c between both corner points v'_{corn}^0 and v'_{corn}^1 is inside or outside Ω , see Algorithm 10, Line 13. Consequently, a new edge is only introduced if $v_c \in \Omega$. Algorithm 4 performs the necessary point membership classification of v_c .

5.4.3. Split edges

After modifying the upper polyline to meet the requirements discussed in Section 5.4.2, we split certain edges $\varepsilon \in \mathcal{E}_{\text{up}} \cup \mathcal{E}_{\text{low}}$ to create simple triangular or quadrilateral domains that can be easily tessellated. During this process, each vertex $\nu \in \mathcal{E}_{\text{low}}$ is projected along η onto $\varepsilon_{\text{up}} \in \mathcal{E}_{\text{up}}$ and vice versa, as depicted in Fig. 6(d). If a target edge ε^1 is found, such that $(\xi_i \in \varepsilon^1) < (\xi \in \nu) < (\xi_r \in \varepsilon^1)$, a new vertex is introduced, and ε^1 is split accordingly. When multiple edges meet this condition, only the nearest edge is considered. If no ε^1 can be found, ν is skipped, as shown in Fig. 6(d) for the last upper edge on the right.

5.4.4. Triangulate open faces

The last step of Algorithm 7 (CLOSEDOMAIN()) is the triangulation of all open faces. For each $\varepsilon_{\text{up}} \in \mathcal{E}_{\text{up}}$, we search for its corresponding partner $\varepsilon_{\text{low}} \in \mathcal{E}_{\text{low}}$, such that $\xi_l \in \varepsilon_{\text{low}} = \xi_l \in \varepsilon_{\text{up}}$ and $\xi_r \in \varepsilon_{\text{low}} = \xi_r \in \varepsilon_{\text{up}}$. The vertices $\mathcal{V} = \{\varepsilon_{\text{up}}^0, \varepsilon_{\text{up}}^1, \varepsilon_{\text{low}}^1, \varepsilon_{\text{low}}^0\}$ represent the polygon enclosed by ε_{up} and ε_{low} in clockwise orientation. Note that $\varepsilon_{\text{up}}^0$ and $\varepsilon_{\text{low}}^0$, or $\varepsilon_{\text{up}}^1$ and $\varepsilon_{\text{low}}^1$ might be identical, resulting in a triangular region, which can be directly added to Γ_c . If we do not find a partner edge for ε_{up} , the polygon is given by $\mathcal{V} = \{\varepsilon_{\text{up}}^0, \varepsilon_{\text{up}}^1, \{\xi \in \varepsilon_{\text{up}}^1, \eta_{\text{min}}\}, \{\xi \in \varepsilon_{\text{up}}^0, \eta_{\text{min}}\}\}$. All resulting quadrilateral domains are triangulated based on their center point, as depicted in Fig. 6(e).

Remark. The computed mesh Γ_c may contain hanging vertices because each open face on $p \in P$ is tessellated completely independently. However, since Γ_c is only used as a parameterization for evaluating Eq. (7), these do not pose a problem or limit accuracy. The only necessary condition is that Γ_c is closed. This criterion is ensured in Section 5.5.

5.5. Quality assurance for computed intersections

Since the algorithms presented above rely on inexact arithmetic, slight fluctuations in the floating point operations are inevitable. In some corner cases, these inaccuracies can lead to unlikely but possible misclassifications, e.g., whether a vertex is classified on the plane or not. Therefore, in this section, we present a simple but effective approach that dramatically increases the robustness of our workflow for arbitrary input geometries. In order to improve the accuracy, erroneous results must be detected and corrected. Thus, Algorithm 1 is extended to guarantee certain quality requirements. Algorithm 11 summarizes the respective improvements. Line 9 estimates the quality of all computed intersections Γ_c^{trial} using four different error measures to assess if the tessellation of all open faces of $\Gamma_{c,c}$ was successful. Since Γ_c is exclusively used to evaluate the divergence theorem in Eq. (7), the results of Eq. (7) serve as the only indicator for determining a successful triangulation. Note that hanging vertices, high aspect ratios, etc., do not affect the accuracy of the boundary integrals and are therefore neglected.

Let $h(g_i)$ be the solution of Eq. (7) integrated over the boundary Γ_c . Function ESTIMATEQUALITY() evaluates $h(g_i)$ for the following anti-derivatives

$$g_1(x) = \begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix}, \quad g_2(x) = \begin{bmatrix} 0 \\ y \\ 0 \end{bmatrix}, \quad g_3(x) = \begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix}, \quad g_4(x) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \quad (14)$$

Since $h(g_1)$, $h(g_2)$, and $h(g_3)$ all compute the volume enclosed by Γ_c , they must provide the same results. Their respective error measures e_i are defined as

$$e_i = \frac{|h(g_i) - \bar{h}|}{\bar{h}}, \quad \text{for } 1 \leq i \leq 3, \quad (15)$$

where

$$\bar{h} = \frac{1}{3} (h(g_1) + h(g_2) + h(g_3)). \quad (16)$$

The constant anti-derivatives g_4 yield a weighted evaluation of all normal vectors n . Therefore, if Γ_c is closed, $h(g_4) \stackrel{!}{=} 0$. We estimate the quality of Γ_c based on the most significant error, as per

$$q = \text{MAX} (e_1, e_2, e_3, |h(g_4)| / A_{\Gamma_c}), \quad (17)$$

where A_{Γ_c} denotes the surface area of Γ_c . If q is larger than a predefined threshold q_{max} , the bounding box B is slightly perturbed, see Algorithm 11, Line 16. We move all six faces ($0 \leq i < 6$) in the direction n_i by the value $\delta_i = \bar{r}_i \delta$, with $\bar{r}_i \in \{1 \dots 100\}$ being a random number and δ denoting a small value close to machine precision, e.g., $\delta = 10^{-14}$. This process is repeated until a surface parameterization with $q < q_{\text{max}}$ is obtained or the maximum number of iterations i_{max} is reached. When $i = i_{\text{max}}$, the best previous solution is returned, see Algorithm 11, Lines 13–14.

If flawed and non-watertight geometries are considered, the computed intersections cannot be expected to be closed. In these cases, boundary meshes with $q > q_{\text{max}}$ may also be accepted. We notice that intersections containing geometric flaws may lead to local integration errors. However, a major advantage is that they can still be evaluated using the divergence theorem in Eq. (7), while Eq. (17) allows quantifying the associated error. Consequently, depending on the size of q , it is possible to decide whether to consider or neglect the corresponding cut element during analysis or even to initiate a recursive subdivision of the domain to further localize the integration error. Section 6.2 demonstrates the effectiveness of the method introduced in this subsection by assessing 4948 STLs from the Thingi10K database [53].

Algorithm 11: Proposed workflow with quality assurance.

Input: Γ : Geometry Boundary, \mathcal{P}_{bg} : Background mesh

```

1: function MAIN( $\Gamma$ ,  $\mathcal{P}_{\text{bg}}$ )
2:   ...
3:   if  $s$  is  $s_{\text{cut}}$  then                                     ▷ See: Algorithm 1.
4:      $\Gamma_c \leftarrow \emptyset$ ;  $i \leftarrow 0$ ;  $q_{\text{best}} \leftarrow \infty$                                      ▷  $\mathcal{B} = \mathcal{B}_{\text{cut}}$ .
5:     while  $i < i_{\text{max}}$  do
6:        $\Gamma_{c,c} \leftarrow \text{CLIPBOUNDARY}(\mathcal{B}, \mathcal{T})$                                      ▷ See: Algorithm 5.
7:        $\mathcal{T}_c \leftarrow \text{BUILDAAABBTREE}(\Gamma_{c,c})$ 
8:        $\Gamma_c^{\text{trial}} \leftarrow \text{CLOSEDOMAIN}(\mathcal{B}, \mathcal{T}_c)$                                      ▷ See: Algorithm 7.
9:        $q \leftarrow \text{ESTIMATEQUALITY}(\Gamma_c^{\text{trial}})$                                      ▷ See: Eq. (17).
10:      if  $q < q_{\text{max}}$  then
11:         $\Gamma_c \leftarrow \Gamma_c^{\text{trial}}$ 
12:        break
13:      else if  $q < q_{\text{best}}$  then
14:         $\Gamma_c \leftarrow \Gamma_c^{\text{trial}}$ ;  $q_{\text{best}} \leftarrow q$ 
15:      end if
16:       $\mathcal{B} \leftarrow \text{PERTURB}(\mathcal{B})$ ;  $i \leftarrow i + 1$ 
17:    end while
18:     $\mathbf{K} \leftarrow \text{COMPUTEMOMENTS}(\Gamma_c)$                                      ▷ See: Eq. (7).
19:    ...
20:  else if  $s$  is  $s_{\text{in}}$  then                                     ▷  $\mathcal{B} = \mathcal{B}_{\text{in}}$ .
21:    ...
22:  end if
23:  ...
24: end function

```

6. Numerical experiments

This section demonstrates the potential of the proposed workflow for embedded and immersed boundary methods. Section 6.1 applies the divergence theorem given in Eq. (7) to assemble the moment fitting equations and compares the performance of different quadrature rules used on the boundary triangles. The presented cell classification, clipping, and intersection algorithms are subjected to an intensive robustness and performance analysis in Sections 6.2 and 6.3. Finally, the entire framework is utilized to perform structural analyses of valid and flawed B-Rep models in Section 6.4.

In all numerical examples, comparisons between vertices are performed with a relative snap tolerance $\delta_{\text{snap}}^{\text{rel}} = \text{MAX}(d_{\text{max}} \delta_{\text{snap}}, \delta_{\text{snap}})$, where d_{max} is the largest axis of the bounding box \mathcal{B} , and $\delta_{\text{snap}} = 10^{-12}$. General floating point operations use a zero tolerance of $\delta_{\text{zero}} = 10^{-14}$. The parameters q_{max} and i_{max} in Algorithm 11 are set to $q_{\text{max}} = 10^{-5}$ and $i_{\text{max}} = 5$. All experiments are conducted on a local workstation with an Intel(R) Xeon(R) W-2255 CPU @ 3.70 GHz and 96 GB system memory.

6.1. Performance comparison of triangle quadrature rules for the evaluation of boundary integrals

In this section, we study the quality of the computed moments (right-hand side of the moment fitting equations) for four differently cut cells, as depicted in Fig. 8. The complexities of the respective domains range from a simple tetrahedron in Fig. 8(a) to clipped miniature figures with detailed features in Fig. 8(d). The main objective is to assess the impact of the quadrature rule used on the boundary triangles for the assembly of the moment fitting equations. For comparison, the same computation is performed with a classical octree. Note that our octree implementation does not rely on discrete seed points to detect intersections but uses the generalized test described in Algorithm 3. We define orthogonal Legendre polynomials as moment fitting bases and construct a trivariate function space from the tensor product given in Eq. (4).

Fig. 9 plots the relative error in the L2-norm, as per

$$e_r = \frac{\|\mathbf{K} - \mathbf{K}_{\text{ref}}\|_{L_2}}{\|\mathbf{K}_{\text{ref}}\|_{L_2}}, \quad (18)$$

where \mathbf{K} denotes the right-hand side in Eq. (3) containing all computed moments, and \mathbf{K}_{ref} is a reference solution. The respective integrands contain functions of order $0 \leq p \leq 2$, resulting in a vector \mathbf{K} with 27 entries. According to [36], this is a sufficient function space to derive accurate quadrature rules for quadratic finite elements. In Fig. 9, e_r is plotted for increasing refinement levels while both schemes are tested for a different number of quadrature points per octant or triangle. Each octant is integrated

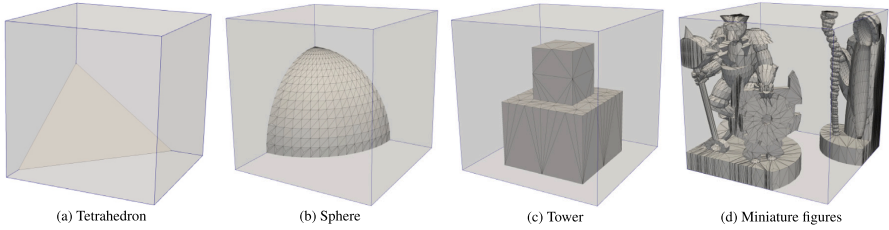


Fig. 8. Four differently cut cells with increasing complexity. Cell (d) is intersected by an STL (id: 82.536) from the Thingi10k database.

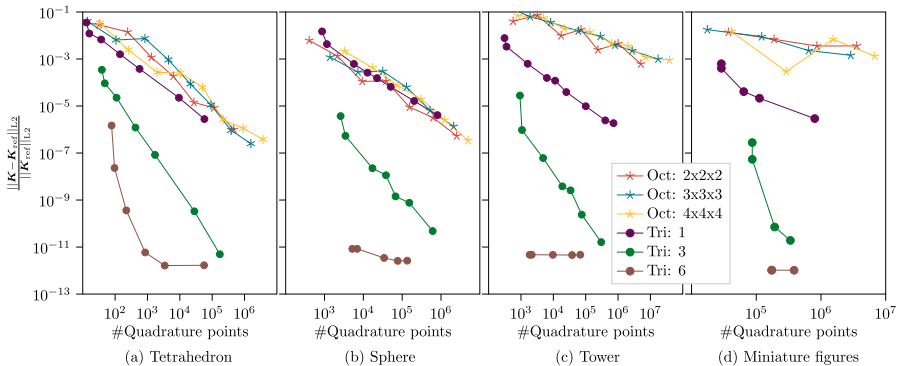


Fig. 9. Quadratic moments: Relative error (see Eq. (18)) of the moments for integrands of order $0 \leq p \leq 2$ (resulting in 27 functions) and four differently cut cells (see Fig. 8). Comparison between octree (Oct) and boundary integration (Tri) with different numbers of quadrature points per octant/triangle.

with $2 \times 2 \times 2$, $3 \times 3 \times 3$, or $4 \times 4 \times 4$ Gauss quadrature points. For all cut leaf octants, only the interior points are considered. The boundary integrals are evaluated with $n_{tri} = 1$, $n_{tri} = 3$, and $n_{tri} = 6$ integration points per triangle [65]. The reference K_{ref} results from an *overkill* solution with $n > 10^6$ boundary triangles, each evaluated with $n_{tri} = 12$ integration points. To refine the boundary mesh Γ_c , we introduce three new vertices at the center of each edge and split the respective triangle into four new sub-triangles. Note that this does not affect the geometry, and the enclosed domain Ω_c remains unchanged.

Fig. 9 indicates that the chosen quadrature rule has no significant effect on the accuracy of the integrals for all studied octree versions. Since the octree delivers only a low-order approximation of the geometrical boundary, this is an expected result. On the contrary, the accuracy of the boundary integrals strongly depends on the integration scheme. It becomes apparent that increasing the order of the triangle quadrature rules can significantly improve the rate of convergence of the relative integration error. In some cases, e.g., in Fig. 9(a) and (b), the boundary integrals with $n_{tri} = 1$ produce similar relative errors as the octrees. However, using $n_{tri} = 3$ or $n_{tri} = 6$ integration points per triangle reduces the maximum error to $e_r = 3.46 \times 10^{-4}$ and $e_r = 1.48 \times 10^{-6}$. Note that these are obtained for the initial discretization of the tetrahedral domain, which contains only one single triangle along the cutting plane, see Fig. 8(a). When the boundary is refined with $n = 200$ triangles, the errors drop to $e_r < 10^{-5}$ and $e_r < 10^{-9}$. In all other examples, where more detailed geometries are represented and hence more triangles are involved (Fig. 9(b)–(d)), the maximum relative errors with $n_{tri} = 3$ and $n_{tri} = 6$ are $e_r < 3 \times 10^{-5}$ and $e_r < 1 \times 10^{-11}$. Fig. 10 shows the respective errors for moment fitting bases of order $0 \leq p \leq 4$, which are suitable to derive quadrature rules for quartic finite elements. Since the function space is more complex, the errors in the computed moments are slightly larger. However, in all cases, $n > 200$ with $n_{tri} = 6$ and $n_{tri} = 12$ produces the following maximum errors $e_r < 10^{-5}$ and $e_r < 10^{-8}$, respectively.

We conclude that boundary integrals are well suited for accurately integrating the right-hand side of Eq. (3). Since the corresponding set of integrands contains not only constant but also polynomial functions with $p > 0$, higher-order triangle quadrature rules can provide better accuracy. It is shown that a suitable quadrature scheme drastically improves the rate of convergence of the relative integration error. This allows all moments to be calculated with near-machine precision and reasonable computational effort.

Remark. In this context, a triangle quadrature rule of a certain degree does not necessarily integrate all polynomials of the same degree exactly. Two factors increase the complexity of the given integral. Firstly, the anti-derivatives in Eq. (6) elevate the

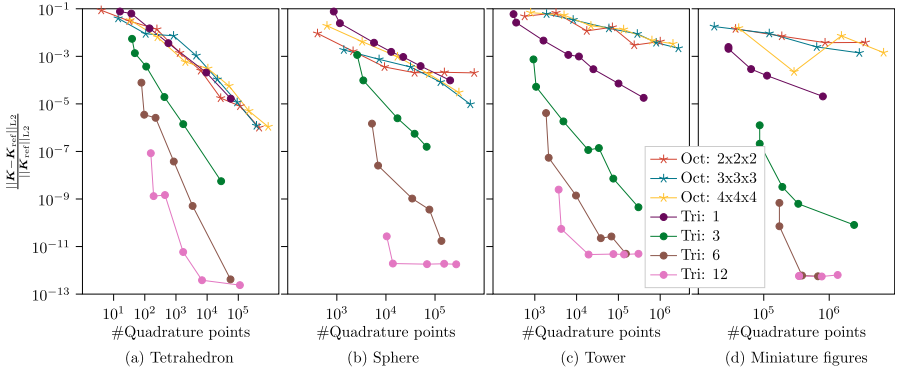


Fig. 10. Quartic moments: Relative error (see Eq. (18)) of the moments for integrands of order $0 \leq p \leq 4$ (resulting in 125 functions) and four differently cut cells (see Fig. 8). Comparison between octree (Oct) and boundary integration (Tri) with different numbers of quadrature points per octant/triangle.



Fig. 11. Sample STLs from the Thinki10K database [53] with their respective model ids.

polynomial order by one. Secondly, oblique cuts do not maintain a tensor product space. Hence, the integral along x can no longer be evaluated for y and z being constant, and vice versa.

6.2. Robustness analysis of cell classification, clipping, and intersection algorithms

For a seamless embedding workflow, the underlying geometrical operations must be reliable and robust. Sections 6.2.1 and 6.2.2 investigate the performance of the proposed methods regarding valid and flawed geometries.

6.2.1. Valid geometries

In the following, the proposed clipping and intersection algorithms are applied to various STLs from the Thingi10K database [53], similar to the study performed in [46]. To this end, we filter all available solid models by simply typing *is solid* on the corresponding web page. Of the 5113 resulting models, 4 are no STLs or could not be parsed into memory. Another 28 STLs are identified as broken, e.g., containing normals that falsely point inward instead of outward and are hence discarded from the set. To ensure valid geometries, we also measure each STL's quality using Eq. (17) and consider only surface meshes with $q \leq 10^{-10}$. A total of 4948 STLs are recovered, meeting all requirements.

Fig. 11 shows 11 samples from the given set and illustrates their diversity. For the following study, each of the 4948 STLs is embedded into a background partition \mathcal{P}_{bg} with ρ_{bg}^{min} and ρ_{bg}^{max} being its extreme points. Based on the bounding box of the STL defined by ρ_{stl}^{min} and ρ_{stl}^{max} , we create the background mesh similarly as in [46]. A Cartesian grid is constructed that is approximately 20% larger in each direction (x, y, z) than the bounding box of the STL. This ensures that every model contains not only interior and

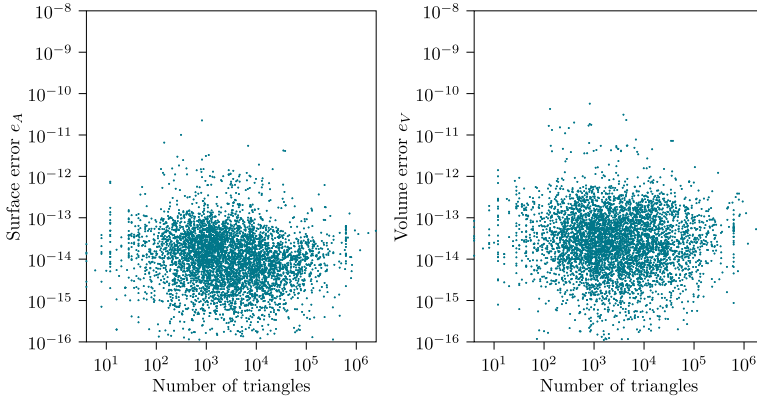


Fig. 12. Performance of clipping and intersection algorithms: Relative surface and volume errors (see Eq. (22)) for 4948 STLs from the Thingi10k database [53].

cut cells but also a considerable number of exterior cells. Each of them must be classified and processed correctly. Consequently, the cell's edge length h is computed as

$$h = 1.2 \operatorname{Min} \left\{ \frac{\operatorname{MAX} \{ \Delta p_{\text{stl}}^0, \Delta p_{\text{stl}}^1, \Delta p_{\text{stl}}^2 \}}{n_{\text{max}}}, \frac{\operatorname{MIN} \{ \Delta p_{\text{stl}}^0, \Delta p_{\text{stl}}^1, \Delta p_{\text{stl}}^2 \}}{n_{\text{min}}} \right\}, \quad (19)$$

where

$$\Delta p_{\text{stl}} = p_{\text{stl}}^{\text{max}} - p_{\text{stl}}^{\text{min}}, \quad (20)$$

and

$$p_{\text{bg}}^{\text{min}} = p_{\text{stl}}^{\text{min}} - 0.1 \Delta p_{\text{stl}}, \quad p_{\text{bg}}^{\text{max}} = p_{\text{bg}}^{\text{min}} + h \left[\frac{1.2 \Delta p_{\text{stl}}}{h} \right]. \quad (21)$$

Eqs. (19)–(21) create a background mesh containing at least n_{max} and n_{min} elements along the longest and shortest axis of \mathcal{P}_{bg} . To assess the clipping and intersection algorithms, we compute the relative errors in surface area and volume, as per

$$e_A = \frac{|\sum A_c - A|}{A}, \quad e_V = \frac{|\sum V_c - V|}{V}, \quad (22)$$

where A_c is the area of the clipped surface mesh $\Gamma_{c,c}$, and A is the area of the input STL Γ . If cell \mathcal{B} is inside Ω and uncut ($\mathcal{B} = \mathcal{B}_{\text{in}}$), V_c is given by the volume of \mathcal{B} . Whereas, if the cell is cut ($\mathcal{B} = \mathcal{B}_{\text{cut}}$), V_c denotes the volume enclosed by Γ_c . Note that for V_c and V , we employ Eq. (7) to compute the respective volumes by integrating over Γ_c and Γ . Fig. 12 depicts the relative surface and volume errors for all 4948 STLs studied, where $n_{\text{min}} = 5$ and $n_{\text{max}} = 50$. The maximum overall error for e_A and e_V is less than 10^{-10} . In addition, no relation between the number of triangles in the STL and the accuracy can be observed.

These results demonstrate the reliability and robustness of the proposed algorithms for computing clipped domains and intersections of arbitrary input geometries. Moreover, since the present workflow relies on a preceding classification of cells, Fig. 12 also implies that all cells are correctly classified as interior, exterior, or cut.

6.2.2. Flawed geometries

A second study shall demonstrate the robustness of the proposed cell classification scheme regarding flawed geometries. We consider the same 4948 STLs processed in Section 6.2.1 but introduce artificial flaws to each of them. When the STLs are parsed into memory, every 100th triangle is modified to mimic gaps, overlaps, and incorrect orientations. Alternately, the respective triangle is ignored, added twice, or flipped, resulting in a mesh where 1% of all triangles are erroneous. Note that a triangle is only ignored if it does not violate the necessary condition of the proposed cell classification scheme. As discussed in Section 5.2, a gap must not be large enough for an uncut cell to fit through. To meet this requirement, the area of ignored triangles is limited by $\leq 2h^2$. An example of a faulty model is shown in Fig. 13(a), where the modified triangles are highlighted. In the following, the results from Section 6.2.1 obtained from valid geometries are used as references. Fig. 13 compares traditional ray tracing with our proposed cell classification scheme. Each bar depicts the results for one STL, measuring the number of cells classified incorrectly due to geometrical flaws. All models where there are no misclassified cells are marked in green, and all others in red. Note that only cells classified originally as interior or exterior are considered. Cut cells can change their state because the triangle that previously cut them is now missing. However, these are local errors, which do not affect the solvability of the system, and are therefore accepted in this study. The same

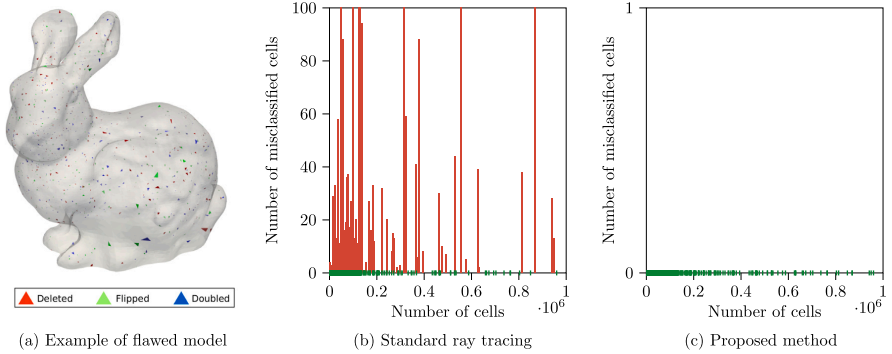


Fig. 13. Comparison between classical ray tracing and the proposed classification scheme in the presence of flawed geometries for 4948 STLs from the Thingi10K database [53]. Each bar corresponds to the number of incorrectly classified cells for one STL. The green bars indicate that not a single misclassification is observed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is true for evaluating boundary integrals, which may also suffer from gaps, overlaps, and falsely oriented triangles. These additional local errors are similarly challenging to circumvent but, again, do not necessarily lead to an unsolvable problem. Since a single exterior cell falsely classified as inside, however, can result in singular system matrices, our primary focus is on correctly classifying interior and exterior cells.

In Fig. 13(b), classical ray tracing is performed, where a majority vote of five rays decides the membership. The result of each individual ray is based on the number of intersections with the geometrical boundary. It is worth mentioning that this approach produces correct classifications for all valid STLs. However, when flaws are introduced to the examined geometries, standard ray tracing fails entirely. A significant number of misclassifications occur in many models. Fig. 13(c) shows the results for the proposed cell classification scheme. For a fair comparison, the maximum number of cast rays per cell is also set to $a_{\text{count}}^{\text{min}} = 5$. However, contrary to the classical approach, rays are exclusively cast for boundary cells within one flood fill group, as discussed in Section 5.2, leaving many cells without a single ray test. The computational cost is further reduced by shooting rays toward the local clipped mesh $\Gamma_{c,c}$ instead of the global STL Γ . As shown in Fig. 13(c), all cells in all models are classified correctly despite severely flawed geometries, demonstrating the robustness of this method.

6.3. Scaling of wall-clock times

Section 6.2 assesses the robustness of the presented embedding workflow. Since we aim not only for a reliable but also an efficient framework, the wall-clock times of the presented algorithms are studied in the following. For this purpose, the 11 STLs depicted in Fig. 11 are investigated and compared to a simple cube used as a reference. Table 1 lists the key figures of the respective surface meshes. Again, we follow the setup suggested in [46], and construct the background mesh \mathcal{P}_{bg} for $n^{\text{max}} = n_0^{\text{max}} 2^\beta$ (see Eqs. (19)–(21)) with $n_0^{\text{max}} = 14$. Algorithm 11 is executed for $\beta = \{0, \dots, 5\}$, and the execution times spent on each subtask are measured. Consequently, up to 448 cells are distributed along the largest axis of \mathcal{P}_{bg} . To reduce the influence of external disturbances, we take the average runtime of 5 simulations. Fig. 14(a)–(b) show the corresponding results for cell classification and intersection. Regardless of the model size and complexity, linear scaling of wall-clock times is observed as the background mesh is refined in Fig. 14(a). This is an expected result because the classification scheme is applied to each cell of the background mesh \mathcal{P}_{bg} . The intersection algorithm, however, is applied only to cut cells, resulting in sublinear scaling, see Fig. 14(b). Note that the plotted elapsed times in Fig. 14(b) account for all operations from Line 4 to Line 17 in Algorithm 11. Since all developments presented in this work are realized in the open-source framework QuESo, the computed intersection Γ_c can directly be used to assemble and solve the moment fitting equations. QuESo solves Eq. (3) during the iterative process of a point elimination algorithm to optimize the weights and the positions of the integration points. For more information on the used quadrature scheme, the reader is referred to [36]. Fig. 14(c) plots the corresponding simulation times for quadratic moment fitting bases. We note that the assembly and solution of the moment fitting equations are more computationally intensive than the intersection of the cut cells. Moreover, increasing the polynomial degree of the bases will only affect the simulation times in Fig. 14(c), but will not influence the computational effort required for the classifications and intersections. Fig. 14(d) shows the accumulated wall-clock times. The graphs tend towards a linear scaling, as the classification of the cells becomes more dominant with an increasing refinement of the background mesh. However, the linear range is not yet reached even with 100 million cells. Since the cell intersection and moment fitting are only applied to a fraction of all active cells in \mathcal{P}_{bg} , Fig. 15 plots their execution times over the number of cut cells. Now all simulations tend towards an expected linear scaling.

We conclude that the proposed framework enables a swift model generation for solving PDEs on unfitted background meshes. This paves the way for a direct CAD-integrated analysis pipeline.

Table 1
Key figures of the 11 STLs depicted in Fig. 11. The model ids represent the ids in the Thingi10k database [53].

Model id	Num faces	Num vertices	Volume	Surface area
37 266	29 472	14 738	68 631.4015	13 112.2210
293 137	292	148	273 280.034	29 490.7155
441 708	112 402	56 203	279 628.299	29 684.9511
551 021	348 128	174 066	19 362.2212	7282.979 87
65 904	157 726	78 869	42 564.245.2	773 637.925
47 076	1532	768	104 370.679	21 864.7523
1 458 688	7736	3844	6568.517 00	5588.699 07
252 119	49 950	24 979	28 004.8858	12 439.2719
82 536	144 024	71 946	18 354.6418	18 682.0280
37 881	3400	1700	8461.643 57	3992.616 28
550 964	6156	3072	989.761 580	1715.987 50
Cube	12	8	1.000 000 00	6.000 000 00

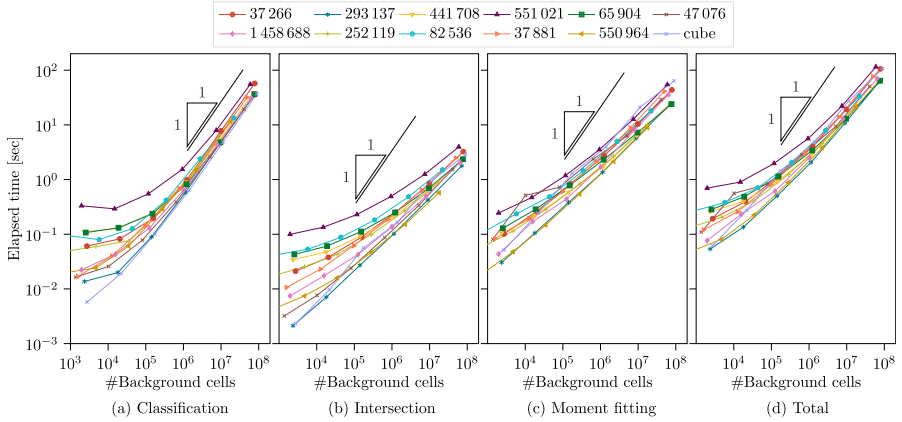


Fig. 14. Wall-clock times spent on different tasks over the number of background cells for 11 STLs from the Thingi10k database [53], see Fig. 11: (a) Classification of cells. (b) Intersection of cells (contains all operations from Line 4 to Line 17 in Algorithm 11). (c) Assembly and solution of the moment fitting equations. (d) Total time. The experiment is conducted on a local workstation with an Intel(R) Xeon(R) W-2255 CPU @ 3.70 GHz and 96 GB system memory.

6.4. Finite element analysis of valid and flawed B-Reps

Finally, the proposed embedding workflow is used to solve linear elasticity problems on complex domains defined by faceted B-Reps. We aim to demonstrate the potential of the proposed methodologies for the solution of PDEs on unfitted background discretization. Therefore, the model generator QuESo [48] is coupled with the finite element framework Kratos Multiphysics [54–56]. The underlying workflow looks the following. QuESo reads the STL file, assembles the moment fitting equations according to Section 5, and constructs highly efficient quadrature rules for each active element in \mathcal{P}_{bg} . Using the point elimination algorithm presented in [36], the number of quadrature points per cut element is bounded by $n_q \leq (p+1)^d$, where p is the polynomial degree, and d is the space dimension. Consequently, each cut element is evaluated with, at most, the same number of points as suggested by Gauss quadrature for regular elements. Moreover, all integration points have positive weights and are located within the material domain. In the next step, the analysis-ready model is passed to Kratos Multiphysics, where the finite element problem is assembled and solved. Note that the data transfer between QuESo and Kratos is limited to the background mesh \mathcal{P}_{bg} and a list of integration points. In all experiments, \mathcal{P}_{bg} is spanned by a trivariate B-Spline domain with a maximum interior knot multiplicity of $k = 1$ and quadratic basis functions, resulting in a C^1 continuous discretization field. Section 6.4.1 shows results from valid geometries, while Section 6.4.2 studies flawed geometries.

6.4.1. Valid geometries

Four different STLs from the Thingi10k database, namely the models with ids 441 708, 551 021, 252 119, and 550 964, see Fig. 11, are subjected to a body force. The material properties of the structures are Young's modulus $E = 2 \times 10^5$ N/m² and Poisson's ratio $\nu = 0.3$. Models 441 708 and 252 119 are loaded with $[0, 0, -10]$ N/m³, while 551 021 and 550 964 are subjected to $[0, 0, -100]$ N/m³.

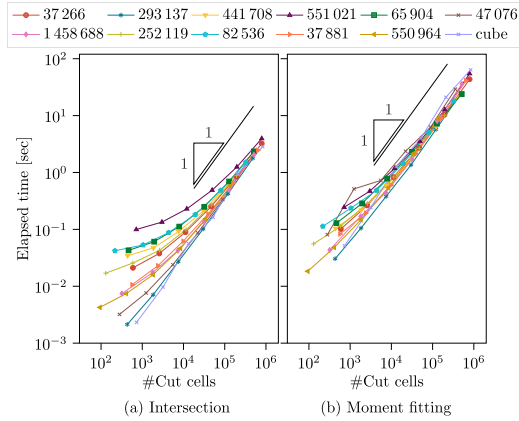


Fig. 15. Wall-clock times spent on different tasks over the number of cut background cells for 11 STLs from the Thingi10k database [53], see Fig. 11: (a) Intersection of cells (contains all operations from Line 4 to Line 17 in Algorithm 11). (b) Assembly and solution of the moment fitting equations. The experiment is conducted on a local workstation with an Intel(R) Xeon(R) W-2255 CPU @ 3.70 GHz and 96 GB system memory.

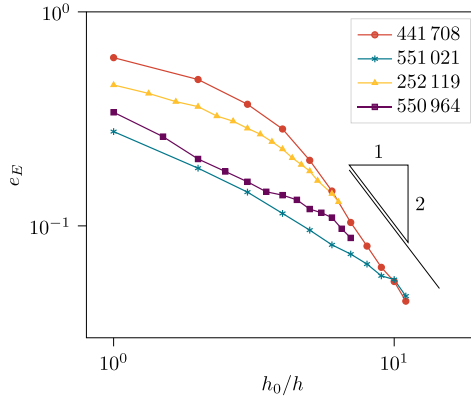


Fig. 16. FE convergence test: Relative error in strain energy for four models from the Thingi10k database [53] subjected to a body force. The unfitted background mesh \mathcal{P}_{bg} is constructed by quadratic C^1 continuous B-Spline bases.

Dirichlet boundary conditions $\bar{u} = [0, 0, 0]$ are enforced at their bases using the penalty method with a penalty factor of $\beta = 10^{12}$ N/m³ for all geometries. Fig. 16 shows the relative errors in strain energy under h -refinement. The error is defined as

$$e_E = \sqrt{\frac{|U^h - U^{ref}|}{U^{ref}}}, \tag{23}$$

where U^h is the approximated strain energy. The reference solution U^{ref} is computed from a discretization that is at least twice as fine (half the element size) as the finest mesh examined. Fig. 16 indicates that the error e_E decreases monotonically and tends towards a quadratic convergence rate for the smallest element sizes. Fig. 17 shows the respective deformation patterns and illustrates the surfaces to which Dirichlet boundary conditions are applied. In all cases, the displacement contours a very smooth.

6.4.2. Flawed geometries

As discussed in Section 5.2, CAD models are not always valid and watertight due to various sources of errors but often contain defects. These imperfections still pose major challenges for the generation of suitable FE models. In the following, we aim to

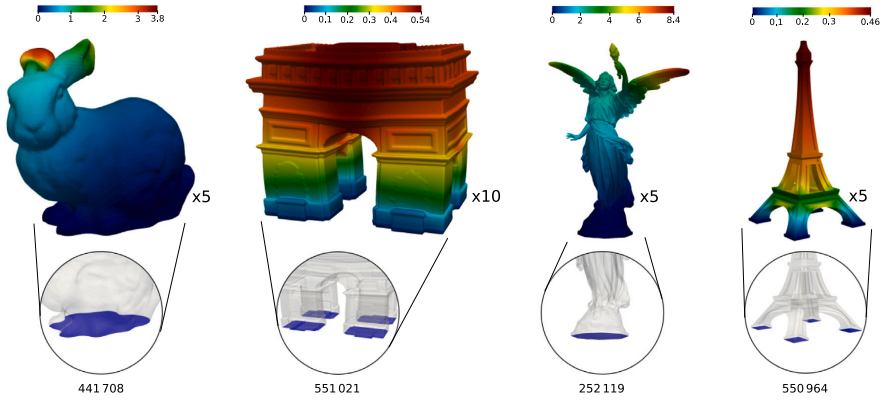


Fig. 17. Deformation contours under self-weight for four different models from the Thing10k database [53]. The blue surfaces depict the Dirichlet boundary conditions. The unfitted background mesh \mathcal{P}_{bg} is constructed by quadratic C^1 continuous B-Spline bases. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

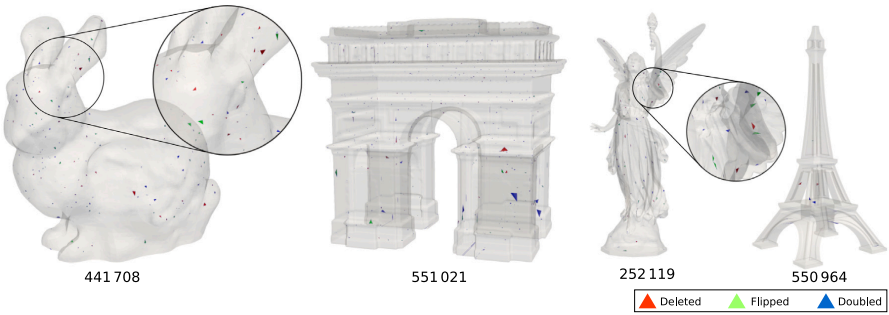


Fig. 18. Artificially introduced flaws. Every 500th triangle is either ignored, flipped, or considered twice.

investigate the potential of our proposed workflow for the direct analysis of such flawed geometries. Therefore, as in Section 6.2.2, we again introduce artificial defects into the examined STLs: 441 708, 551 021, 252 119, and 550 964. Two test series with 0.1% and 0.2% modified triangles are conducted. Every 1000th or 500th triangle in each STL is alternately ignored, flipped, or considered twice. Fig. 18 illustrates the flawed geometries highlighting the modified triangles. All four models are again subjected to the same boundary conditions as in the previous subsection. Fig. 19 compares the strain energies obtained with those calculated for the valid (watertight) geometries. The results show that some discrepancies are visible for very coarse meshes. However, when the meshes are refined, the flawed geometries produce similar results as the valid models. We note that Fig. 19 shows absolute values of the strain energy and does not aim to demonstrate optimal convergence rates. In fact, the convergence rate may still be affected by the present flaws. Nevertheless, the proposed integration scheme allows all simulations to terminate successfully and produce results with sufficient accuracy for many practical problems that cannot be easily solved with classical methods.

7. Conclusion

This publication presents a complete and fully automated methodology for the fast and robust numerical integration of embedded B-Reps. For seamless incorporation into industrial and scientific workflows, the algorithm can process all B-Reps available as oriented surface meshes, e.g., STL, and is optimized for the application of grid-based background discretizations. The realization of the framework involves two major developments. One main objective is to provide the necessary parameterization for the efficient assembly of the moment fitting equations. At its core, a novel intersection algorithm facilitates the direct application of the divergence theorem to compute the necessary moments. The result is a closed surface mesh representing the intersection between

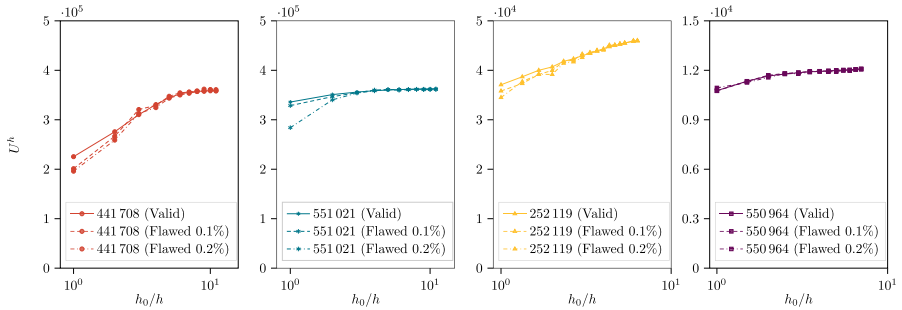


Fig. 19. Comparison of strain energies U^h obtained from valid and flawed geometries. The percentages indicate the number of modified triangles. The unfitted background mesh \mathcal{P}_{bg} is constructed by quadratic C^1 continuous B-Spline bases.

the geometrical domain and each cut element. Since this mesh is used exclusively for integration purposes, the algorithm is subject to a single quality criterion. To this end, we compare the boundary integral of the intersection with solutions of the divergence theorem, which are known for closed domains. Other standard mesh quality measures, regarding, e.g., aspect ratios, hanging vertices, duplicated nodes, etc., are not enforced, resulting in a very efficient implementation. This work's second contribution deals with partitioning the background mesh into interior, exterior, and cut elements. The developments include the extension of a robust flood fill-based classification scheme for application to all geometrical topologies, including interior cavities. Furthermore, the algorithm is adapted to run in parallel, drastically reducing execution times.

The presented methods are successfully applied to 4948 STLs from the Thingi10K database [53], ranging from simple geometries to extremely complex models with detailed features and millions of faces. All elements in the given background mesh are correctly classified and intersected for each STL with a maximum relative volume error of $<10^{-10}$. Applying higher-order quadrature rules on the elements of the computed boundary meshes allows the integration of the moment fitting bases via the divergence theorem with near-machine precision. We also show that the extended classification scheme yields identical results regardless of whether it operates on valid or severely flawed geometries. These outcomes illustrate the robustness of the proposed methods. A subsequent study demonstrates that the overall wall-clock times scale at most linearly with the number of elements in the background mesh. The computational effort for performing the geometric operations is of the same order of magnitude as for solving the moment fitting equations with quadratic bases. Finally, the proposed workflow is successfully used for the structural analysis of embedded solids defined by valid and flawed geometries. A comparison shows that the same values in strain energy are obtained when missing, duplicated, or falsely oriented faces are introduced to the initially valid models.

We provide all discussed developments as an implementation in the open-source C++ framework QuEso (Quadrature for Embedded Solids) [48], which was initially designed to solve the moment fitting equations for highly efficient quadrature rules. In conjunction with the methodologies presented in this work, QuEso can generate analysis-ready FE models from arbitrarily complex faceted B-Reps. For a given background grid, the model is output as a list of integration points and can hence be processed in any FE solver without significant adjustments. This enables a CAD-integrated simulation pipeline from the initial design in CAD (STL) to the direct solution of PDEs on unfitted background meshes. An interface to the FE framework Kratos Multiphysics [54–56] is provided in [48].

Since all required computations of cut cells are defined cell-wise, the respective algorithms are embarrassingly parallel, which is a major advantage over unstructured mesh generators. Extending the current implementation to distributed memory machines is the subject of future work. Moreover, the presented workflow may be developed further for time-dependent or highly nonlinear problems where the background mesh undergoes extreme deformations and needs to be reset.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code related to this article can be found at <https://github.com/manuelmessmer/QuEso>.

Acknowledgments

The authors gratefully acknowledge support from the Deutsche Forschungsgemeinschaft (DFG), Germany as part of the SPP 2187 with the project 423935790 “Der digitale Baukasten – Simulationsbasierte Modelle und Methoden für den Entwurf modularer Tragsysteme aus Beton” as well as the support of the DFG through the project 414265976 TRR 277 C-01 and C-02.

References

- [1] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Comput. Methods Appl. Mech. Engrg.* 194 (2005) 4135–4195, <http://dx.doi.org/10.1016/j.cma.2004.10.008>.
- [2] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, *Isogeometric Analysis: Toward Integration of CAD and FEA*, John Wiley & Sons, 2009, <http://dx.doi.org/10.1002/9780470749081>.
- [3] T. Belytschko, R. Gracie, G. Ventura, A review of extended/generalized finite element methods for material modeling, *Modelling Simul. Mater. Sci. Eng.* 17 (4) (2009) 043001, <http://dx.doi.org/10.1088/0965-0393/17/4/043001>.
- [4] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method, *Appl. Numer. Math.* 62 (4) (2012) 328–341, <http://dx.doi.org/10.1016/j.apnum.2011.01.008>.
- [5] E. Burman, S. Claus, P. Hansbo, M.G. Larson, A. Massing, CutFEM: Discretizing geometry and partial differential equations, *Internat. J. Numer. Methods Engrg.* 104 (7) (2015) 472–501, <http://dx.doi.org/10.1002/nme.4823>.
- [6] S. Badia, F. Verdugo, A.F. Martín, The aggregated unfitted finite element method for elliptic problems, *Comput. Methods Appl. Mech. Engrg.* 336 (2018) 533–553, <http://dx.doi.org/10.1016/j.cma.2018.03.022>.
- [7] S. Badia, A.F. Martín, F. Verdugo, Mixed aggregated finite element methods for the unfitted discretization of the Stokes problem, *SIAM J. Sci. Comput.* 40 (6) (2018) B1541–B1576, <http://dx.doi.org/10.1137/18M1185624>.
- [8] H.-J. Kim, Y.-D. Seo, S.-K. Youn, Isogeometric analysis for trimmed CAD surfaces, *Comput. Methods Appl. Mech. Engrg.* 198 (37) (2009) 2982–2995, <http://dx.doi.org/10.1016/j.cma.2009.05.004>.
- [9] R. Schmidt, R. Wüchner, K.-U. Bletzinger, Isogeometric analysis of trimmed NURBS geometries, *Comput. Methods Appl. Mech. Engrg.* 241–244 (2012) 93–111, <http://dx.doi.org/10.1016/j.cma.2012.05.021>.
- [10] M. Breitenberger, A. Apostolatos, B. Philipp, R. Wüchner, K.-U. Bletzinger, Analysis in computer aided design: Nonlinear isogeometric B-Rep analysis of shell structures, *Comput. Methods Appl. Mech. Engrg.* 284 (2015) 401–457, <http://dx.doi.org/10.1016/j.cma.2014.09.033>.
- [11] J. Parvizian, A. Düster, E. Rank, Finite cell method, *Comput. Mech.* 41 (1) (2007) 121–133, <http://dx.doi.org/10.1007/s00466-007-0173-y>.
- [12] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, *Comput. Methods Appl. Mech. Engrg.* 197 (45) (2008) 3768–3782, <http://dx.doi.org/10.1016/j.cma.2008.02.036>.
- [13] F. de Prenter, C.V. Verhoosel, E.H. van Brummelen, M.G. Larson, S. Badia, Stability and conditioning of immersed finite element methods: Analysis and remedies, *Arch. Comput. Methods Engrg.* 30 (6) (2023) 3617–3656, <http://dx.doi.org/10.1007/s11831-023-09913-0>.
- [14] N. Sukumar, N. Moës, B. Moran, T. Belytschko, Extended finite element method for three-dimensional crack modelling, *Internat. J. Numer. Methods Engrg.* 48 (11) (2000) 1549–1570, [http://dx.doi.org/10.1002/1097-0207\(20000820\)48:11<1549::AID-NME955>3.0.CO;2-A](http://dx.doi.org/10.1002/1097-0207(20000820)48:11<1549::AID-NME955>3.0.CO;2-A).
- [15] N. Sukumar, D. Chopp, N. Moës, T. Belytschko, Modeling holes and inclusions by level sets in the extended finite-element method, *Comput. Methods Appl. Mech. Engrg.* 190 (46) (2001) 6183–6200, [http://dx.doi.org/10.1016/S0045-7825\(01\)00215-8](http://dx.doi.org/10.1016/S0045-7825(01)00215-8).
- [16] N. Moës, M. Cloirec, P. Cartraud, J.-F. Remacle, A computational approach to handle complex microstructure geometries, *Comput. Methods Appl. Mech. Engrg.* 192 (28) (2003) 3163–3177, [http://dx.doi.org/10.1016/S0045-7825\(03\)00346-3](http://dx.doi.org/10.1016/S0045-7825(03)00346-3).
- [17] S. Loehnert, D.S. Mueller-Hoeppe, P. Wriggers, 3D corrected XFEM approach and extension to finite deformation theory, *Internat. J. Numer. Methods Engrg.* 86 (4–5) (2011) 431–452, <http://dx.doi.org/10.1002/nme.3045>.
- [18] T. Rübberg, F. Cirak, J.M. Garcia Aznar, An unstructured immersed finite element method for nonlinear solid mechanics, *Adv. Model. Simul. Eng. Sci.* 3 (22) (2016) <http://dx.doi.org/10.1186/s40323-016-0077-5>.
- [19] B. Müller, F. Kummer, M. Oberlack, Y. Wang, Simple multidimensional integration of discontinuous functions with application to level set methods, *Internat. J. Numer. Methods Engrg.* 92 (7) (2012) 637–651, <http://dx.doi.org/10.1002/nme.4353>.
- [20] A. Abedian, J. Parvizian, A. Düster, E. Rank, Finite cell method compared to h-version finite element method for elasto-plastic problems, *Appl. Math. Mech.* 35 (2014) 1239–1248, <http://dx.doi.org/10.1007/s10483-014-1861-9>.
- [21] A. Abedian, A. Düster, An extension of the finite cell method using boolean operations, *Comput. Mech.* 59 (2017) 877–886, <http://dx.doi.org/10.1007/s00466-017-1378-3>.
- [22] M. Petó, S. Eisenräger, F. Duvigneau, D. Juhre, Boolean finite cell method for multi-material problems including local enrichment of the Ansatz space, *Comput. Mech.* 72 (2023) 743–764, <http://dx.doi.org/10.1007/s00466-023-02305-y>.
- [23] M. Petó, W. Garhoun, F. Duvigneau, S. Eisenräger, A. Düster, D. Juhre, Octree-based integration scheme with merged sub-cells for the finite cell method: Application to non-linear problems in 3D, *Comput. Methods Appl. Mech. Engrg.* 401 (2022) 115565, <http://dx.doi.org/10.1016/j.cma.2022.115565>.
- [24] L. Kudela, N. Zander, S. Kollmannsberger, E. Rank, Smart octrees: Accurately integrating discontinuous functions in 3D, *Comput. Methods Appl. Mech. Engrg.* 306 (2016) 406–426, <http://dx.doi.org/10.1016/j.cma.2016.04.006>.
- [25] G. Ventura, On the elimination of quadrature subcells for discontinuous functions in the xExtended Finite-Element Method, *Internat. J. Numer. Methods Engrg.* 66 (5) (2006) 761–795, <http://dx.doi.org/10.1002/nme.1570>.
- [26] G. Ventura, E. Benvenuti, Equivalent polynomials for quadrature in Heaviside function enriched elements, *Internat. J. Numer. Methods Engrg.* 102 (3–4) (2015) 688–710, <http://dx.doi.org/10.1002/nme.4679>.
- [27] Y. Sudhakar, J.P. Moitinho de Almeida, W.A. Wall, An accurate, robust, and easy-to-implement method for integration over arbitrary polyhedra: Application to embedded interface methods, *J. Comput. Phys.* 273 (2014) 393–415, <http://dx.doi.org/10.1016/j.jcp.2014.05.019>.
- [28] S. Ducez, U. Gabbert, Efficient integration method for fictitious domain approaches, *Comput. Mech.* 56 (2015) 725–738, <http://dx.doi.org/10.1007/s00466-015-1197-3>.
- [29] S.E. Mousavi, H. Xiao, N. Sukumar, Generalized Gaussian quadrature rules on arbitrary polygons, *Internat. J. Numer. Methods Engrg.* 82 (1) (2010) 99–113, <http://dx.doi.org/10.1002/nme.2759>.
- [30] S.E. Mousavi, N. Sukumar, Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons, *Comput. Mech.* 47 (5) (2011) 535–554, <http://dx.doi.org/10.1007/s00466-010-0562-5>.
- [31] H. Xiao, Z. Gimbutas, A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions, *Comput. Math. Appl.* 59 (2) (2010) 663–676, <http://dx.doi.org/10.1016/j.camwa.2009.10.027>.
- [32] S. Hubrich, P. Di Stolfo, L. Kudela, S. Kollmannsberger, E. Rank, A. Schröder, A. Düster, Numerical integration of discontinuous functions: moment fitting and smart octree, *Comput. Mech.* 60 (5) (2017) 863–881, <http://dx.doi.org/10.1007/s00466-017-1441-0>.
- [33] A.P. Nagy, D.J. Benson, On the numerical integration of trimmed isogeometric elements, *Comput. Methods Appl. Mech. Engrg.* 284 (2015) 165–185, <http://dx.doi.org/10.1016/j.cma.2014.08.002>.
- [34] G. Legrain, Non-negative moment fitting quadrature rules for fictitious domain methods, *Comput. Math. Appl.* 99 (2021) 270–291, <http://dx.doi.org/10.1016/j.camwa.2021.07.019>.
- [35] W. Garhoun, A. Düster, Non-negative moment fitting quadrature for cut finite elements and cells undergoing large deformations, *Comput. Mech.* 70 (2022) 1059–1081, <http://dx.doi.org/10.1007/s00466-022-02203-9>.
- [36] M. Meßner, T. Teschemacher, L.F. Leidinger, R. Wüchner, K.-U. Bletzinger, Efficient CAD-integrated isogeometric analysis of trimmed solids, *Comput. Methods Appl. Mech. Engrg.* 400 (2022) 115584, <http://dx.doi.org/10.1016/j.cma.2022.115584>.

- [37] S.E. Mousavi, N. Sukumar, Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons, *Comput. Mech.* 47 (2011) 535–554, <http://dx.doi.org/10.1007/s00466-010-0562-5>.
- [38] Y. Sudhakar, W.A. Wall, Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods, *Comput. Methods Appl. Mech. Engrg.* 258 (2013) 39–54, <http://dx.doi.org/10.1016/j.cma.2013.01.007>.
- [39] B. Müller, F. Kummer, M. Oberlack, Highly accurate surface and volume integration on implicit domains by means of moment-fitting, *Internat. J. Numer. Methods Engrg.* 96 (8) (2013) 512–528, <http://dx.doi.org/10.1002/nme.4569>.
- [40] M. Joulaian, S. Hubrich, A. Düster, Numerical integration of discontinuities on arbitrary domains based on moment fitting, *Comput. Mech.* 57 (2016) 979–999, <http://dx.doi.org/10.1007/s00466-016-1273-3>.
- [41] CGAL: <https://www.cgal.org/>.
- [42] Open CASCADE: <https://www.opencascade.com/>.
- [43] GTS: <https://gts.sourceforge.net/>.
- [44] R.A.K. Sanches, P.B. Bornemann, F. Cirak, Immersed b-spline (i-spline) finite element method for geometrically complex domains, *Comput. Methods Appl. Mech. Engrg.* 200 (13) (2011) 1432–1445, <http://dx.doi.org/10.1016/j.cma.2010.12.008>.
- [45] D. Schilling, M. Ruess, N. Zander, Y. Bazilevs, A. Düster, E. Rank, Small and large deformation analysis with the p- and B-spline versions of the Finite Cell Method, *Comput. Mech.* 50 (2012) 445–478, <http://dx.doi.org/10.1007/s00466-012-0684-z>.
- [46] S. Badia, P.A. Martorell, F. Verdugo, Geometrical discretisations for unfitted finite elements on explicit boundary representations, *J. Comput. Phys.* 460 (2022) 111162, <http://dx.doi.org/10.1016/j.jcp.2022.111162>.
- [47] B. Wassermann, S. Kollmannsberger, S. Yin, L. Kudela, E. Rank, Integrating CAD and numerical analysis: ‘Dirty geometry’ handling using the Finite Cell Method, *Comput. Methods Appl. Mech. Engrg.* 351 (2019) 808–835, <http://dx.doi.org/10.1016/j.cma.2019.04.017>.
- [48] M. Meßmer, QuESo, <https://github.com/manuelmessmer/QuESo>.
- [49] S. Hubrich, M. Joulaian, A. Düster, Numerical integration in the finite cell method based on moment-fitting, in: *Proceedings of 3rd ECCOMAS Young Investigators Conference; 6th GACM Colloquium on Computational Mechanics, Aachen, Germany, 2015*.
- [50] S. Hubrich, A. Düster, Numerical integration for nonlinear problems of the finite cell method using an adaptive scheme based on moment fitting, *Comput. Math. Appl.* 77 (7) (2019) 1983–1997, <http://dx.doi.org/10.1016/j.camwa.2018.11.030>.
- [51] C.L. Lawson, R.J. Hanson, *Solving Least Squares Problems*, Vol. 15, SIAM, 1995.
- [52] A. Abedian, J. Parvizian, A. Düster, H. Khademyzadeh, E. Rank, Performance of different integration schemes in facing discontinuities in the finite cell method, *Int. J. Comput. Methods* 10 (2013) 1350002/1–24, <http://dx.doi.org/10.1142/S0219876213500023>.
- [53] Q. Zhou, A. Jacobson, Thing10K: A dataset of 10,000 3D-Printing Models, 2016, arXiv preprint [arXiv:1605.04797](https://arxiv.org/abs/1605.04797).
- [54] P. Dadvand, R. Rossi, E. Oñate, An Object-oriented Environment for Developing Finite Element Codes for Multi-disciplinary Applications, *Arch. Comput. Methods Eng.* 17 (2010) 253–297, <http://dx.doi.org/10.1007/s11831-010-9045-2>.
- [55] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S. Idelsohn, E. Oñate, Migration of a generic multi-physics framework to HPC environments, *Comput. & Fluids* 80 (2013) 301–309, <http://dx.doi.org/10.1016/j.compfluid.2012.02.004>.
- [56] V.M. Ferrándiz, P. Bucher, R. Rossi, J. Cotela, J. Carbonell, R. Zorrilla, R. Tosi, et al., KratosMultiphysics (Version 8.0), Zenodo, 2020, <http://dx.doi.org/10.5281/zenodo.3234644>.
- [57] C. Ericson, *Real-Time Collision Detection, The Morgan Kaufmann Series in Interactive 3D Technology*, 2005.
- [58] L. Hedges, AABBC, <https://github.com/lohedges/aabbc>.
- [59] A. Williams, S. Barrus, R.K. Morley, P. Shirley, An efficient and robust ray-box intersection algorithm, in: *ACM SIGGRAPH 2005 Courses*, Association for Computing Machinery, 2005, <http://dx.doi.org/10.1145/1198555.1198748>.
- [60] S. Gottschalk, M.C. Lin, D. Manocha, OBTree: a hierarchical structure for rapid interference detection, in: *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996, <http://dx.doi.org/10.1145/237170.237244>.
- [61] T. Möller, B. Trumbore, Fast, minimum storage ray-triangle intersection, *J. Graph. Tools* 2 (1) (1997) 21–28, <http://dx.doi.org/10.1080/10867651.1997.10487468>.
- [62] I.E. Sutherland, G.W. Hodgman, Reentrant polygon clipping, *Commun. ACM* 17 (1) (1974) 32–42, <http://dx.doi.org/10.1145/360767.360802>.
- [63] F.P. Preparata, M.I. Shamos, *Computational Geometry: An Introduction*, first ed., Springer, New York, 1985, <http://dx.doi.org/10.1007/978-1-4612-1098-6>.
- [64] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, R.L. Phillips, *Introduction to Computer Graphics*, Addison-Wesley Professional, ISBN: 978-0-201-60921-9, 1993.
- [65] C. Felippa, A compendium of FEM integration formulas for symbolic work, *Eng. Comput.* 21 (2004) 867–890, <http://dx.doi.org/10.1108/02644400410554362>.

Publication III

Comput. Methods Appl. Mech. Engrg. 426 (2024) 116999

Shape optimization of embedded solids using implicit Vertex-Morphing

Manuel Meßmer, Reza Najian Asl, Stefan Kollmannsberger, Roland Wüchner,
Kai-Uwe Bletzinger

Highlights

- Extension of Vertex-Morphing to embedded boundary methods.
- Rigorous elimination of mesh distortion problems.
- Discretization-independent sensitivity analysis.
- Robust enforcement of design space constraints.
- Application to solid structures with industrial complexity.

Doi: <https://doi.org/10.1016/j.cma.2024.116999>

The included publication is an open access article published by Elsevier B.V. under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).



Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Shape optimization of embedded solids using implicit Vertex-Morphing

Manuel Meßmer^{a,*}, Reza Najian Asl^a, Stefan Kollmannsberger^b, Roland Wüchner^c, Kai-Uwe Bletzinger^a

^a Chair of Structural Analysis, Technical University of Munich, Arcisstr. 21, 80333 München, Germany

^b Chair of Data Science in Civil Engineering, Bauhaus-Universität Weimar, Coudraystr. 13b, 99423 Weimar, Germany

^c Institute of Structural Analysis, Technische Universität Braunschweig, Beethovenstr. 51, 38106 Braunschweig, Germany

ARTICLE INFO

Keywords:

Embedded finite elements
Implicit Vertex-Morphing
Mesh-independent sensitivity analysis
Solids in discrete boundary representation
Constrained design space

ABSTRACT

One of the biggest challenges in optimizing the shape of complex solids is the requirement to maintain a reasonable mesh quality not only at the boundary but also for the bulk discretization of the interior. Thus, additional regularization and, in many cases, re-meshing of the structure during the iterative process is unavoidable with a Lagrangian description. By tracking the shape update using an Eulerian representation, embedded boundary methods are a promising technique for eliminating mesh distortion problems.

This work consistently combines the unique features of implicit Vertex-Morphing and embedded boundary methods, facilitating the node-based shape optimization of solids with industrial complexity. One of the crucial elements for solving the primal problem on a fixed background grid is an efficient and robust quadrature scheme. To this end, we incorporate the open-source C++ framework QuESo (<https://github.com/manuelmessmer/QuESo>) developed for the numerical integration of arbitrarily complex embedded solids defined by oriented boundary meshes, e.g., in STereoLithography (STL) format. Meanwhile, applying the Helmholtz/Sobolev-based (implicit) filter to the vertices of the embedded boundary mesh not only exploits the extensive design space of node-based optimization but also ensures robust control over the feature size. To realize the above methodology, this work introduces a novel sensitivity analysis that yields mesh-independent shape gradients with respect to the immersed boundary. Through a specific sensitivity weighting, we recover a continuous gradient field from discrete values calculated at the nodes of the background grid. In addition, the presented workflow ensures robust enforcement of challenging geometrical constraints, including minimum wall thicknesses and design space limitations.

We critically assess our approach with benchmarks and structures of industrial relevance. In all examples, trivariate B-Spline bases span the background grid, providing highly accurate finite element solutions and shape sensitivities at every iteration. Moreover, the elimination of mesh distortion problems enables a successful termination at the local optimum, even for large shape modifications.

* Corresponding author.

E-mail address: manuel.messmer@tum.de (M. Meßmer).

<https://doi.org/10.1016/j.cma.2024.116999>

Received 23 February 2024; Received in revised form 10 April 2024; Accepted 15 April 2024

Available online 23 April 2024

0045-7825/© 2024 The Author(s).

Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Traditionally, shape optimization was predominately driven by Lagrangian Finite Element (FE) discretizations. One of the major drawbacks of such approaches is the inevitable distortion of the computational mesh for large shape updates. This problem is particularly pronounced for solid structures since an accurate finite element solution requires a reasonable mesh quality not only of the boundary but also of the internal bulk discretization. Studied remedies to maintain the initial discretization quality include adaptive finite elements [1–4], mesh moving strategies [5,6], and smoothing or filtering techniques [7,8]. However, these methods increase the computational effort and often only postpone the need to completely re-mesh the domain to a later phase of the optimization.

Immersed or embedded boundary methods [9–12], on the other hand, enable the complete decoupling of geometry representation and FE discretization and are, therefore, a promising concept for eliminating mesh distortion problems stemming from large shape updates from the outset. Their main idea is to maintain a simple computational discretization, i.e., a regular grid, hereafter referred to as background mesh, whose elements are intersected by the embedded/immersed boundary. The geometry is no longer represented by an explicit boundary-fitted discretization but by modified quadrature schemes within the cut domains. Thus, the combination of shape optimization and embedded methods allows the changing boundary to move freely through the steady background mesh, potentially guaranteeing a high-quality discretization of arbitrarily complex geometries at every iteration. One class of shape optimization methods deals with embedded geometries that are implicitly defined in terms of level-set functions [13–18], where the shape updates are classically driven by the solution of an additional boundary value problem, e.g., the Hamilton–Jacobi equation. Several other approaches resort to explicit geometry descriptions with variants that employ analytical expressions [19], parametric shape designs [20–22], or CAD-based representations defined by Non-Uniform-Rational B-Splines (NURBS) [23,24]. By contrast, the authors in [25] rely on a discrete geometry description, enabling an independent manipulation of all contour nodes. In [26], the concept is further developed to leverage the merits of shape and topology optimization using an embedded domain approach. Furthermore, [27,28] extend the scope of application to aerodynamic systems.

Although many of the above approaches are conceptually applicable to solid structures, the examples shown are mostly limited to two-dimensional shapes or simple three-dimensional extrusions. A review of the existing publications indicates that the quadrature rules employed in this context are one of the remaining bottlenecks preventing application to volumetric structures with industrial complexity. Numerical integration schemes applied to optimization workflows with embedded boundaries include the area-fraction method [15,29,30], sub-triangulation [22], discontinuous shape functions [16,31], and integration point oversampling [25,26].

This publication integrates the concept of implicit Vertex-Morphing, an efficient, mesh-independent parameterization technique (or regularization) for node-based optimization, with highly efficient and robust quadrature schemes. These integration rules are designed specifically for arbitrarily complex embedded solids described in Boundary Representation (B-Rep), ensuring precise analysis. The presented workflow incorporates the recent developments of the open-source C++ framework QuESo (Quadrature for Embedded Solids) [32] presented in [33,34]. Similar to [25], we allow the individual manipulation of all vertices of a discretized geometry. In our study, we define the solid’s geometry by its outer surface as an oriented triangle mesh, allowing seamless integration into node-based shape optimization with a vast design space. Provided with such a discrete B-Rep model, e.g., in STL format, QuESo outputs a list of integration points to accurately evaluate required bulk and surface terms for the user-defined background mesh in a fully automatized process. The algorithm solves the moment fitting equations [35–37] during an iterative point elimination scheme [33,37,38]. Necessary moments are computed employing the divergence theorem over each cut element’s boundary [39,40] calculated by a novel intersection algorithm [34]. The publication at hand exploits the extraordinary robustness of this procedure, as comprehensively demonstrated in [34], to optimize the shape of embedded solids with industrial relevance.

Generally, gradient-based optimization requires the accurate evaluation of shape sensitivities of the objective functional. For a comprehensive overview of different approaches to calculating shape updates in the context of parameter-free optimization, the interested reader is referred to [41]. While sensitivities are classically computed at the nodes of a boundary-fitted discretization, embedded methods necessitate their evaluation at the immersed boundary using the primal solution computed on the background mesh. The second contribution of this work is a sensitivity analysis, where the objective functional is evaluated over the volumetric domain instead of being projected onto the boundary as in existing methods [25,26]. Embedded discretizations establish a direct relationship between each background node and a section of the immersed boundary. Our developments incorporate this association in the sensitivity analysis, yielding mesh-independent and smooth gradients. As a result, the sensitivities and, hence, the shape updates of each vertex on the immersed boundary are governed by the corresponding node of the background mesh, while each vertex remains an individual design variable. To this end, we first evaluate the sensitivities at the nodes of the background mesh by classical semi-analytical methods derived from the discrete formulation [42,43]. Subsequently, the sensitivity weighting initially introduced in the context of isogeometric shape optimization [43] is adopted for immersed boundaries to recover a continuous gradient field on the background mesh, which can be evaluated at any point on the structure. Once the sensitivities are available on the immersed boundary, i.e., immersed surface mesh, classical node-based shape optimization techniques may be employed [44–47]. For a simple example, we demonstrate that the computed gradient field is smooth enough to drive shape updates without additional regularization. Nevertheless, the proposed workflow applies implicit Vertex-Morphing [48] to control the feature size based on a user-defined filter radius. Compared to explicit filters, the implicit formulation is shown to be computationally less expensive, especially for large filter radii [48]. Furthermore, the implicit approach is unconditionally mesh-independent and allows the consideration of non-design surfaces directly as Dirichlet boundary conditions.

Lastly, we incorporate a robust methodology to enforce geometrical constraints, such as minimum wall thicknesses and design space limitations. For this purpose, a damping matrix is introduced into the geometric parameterization and the sensitivity filtering.

The presented workflow can straightforwardly be applied to higher-order and higher-continuous discretizations. All studied examples exploit this property and employ quadratic C^1 continuous B-Spline basis functions, providing highly accurate primal solutions and sensitivity fields.

This work is organized as follows.

- Section 2 introduces the employed embedded boundary method along with the underlying quadrature schemes.
- Section 3 provides the optimization algorithms and filter techniques used in this work.
- Section 4 presents the mesh-independent sensitivity analysis, which extends the sensitivity weighting proposed in [43] to embedded boundary methods.
- Section 5 describes the enforcement of geometrical constraints, including wall thickness and design space constraints.
- Section 6 summarizes the overall workflow to perform shape optimization of embedded solid B-Reps of arbitrary complexity.
- Section 7 demonstrates the potential of the presented methodology using industrial examples.

2. Embedded boundary method

This section provides a brief introduction to the realm of embedded boundary methods. However, the focus is on the central aspects that are relevant to the publication at hand. For further information, we refer the reader to the papers on the following methods: the Cut Finite Element Method (CutFEM) [10,49], the Aggregated Finite Element Method (AgFEM) [11,50], IGA of trimmed Non-Uniform Rational B-Spline (NURBS) patches [51–53], the implicit mesh discontinuous Galerkin method [54,55], and the Finite Cell Method (FCM) [12,56].

2.1. Definition

Classical finite element analysis requires the creation of a boundary-fitted mesh, which explicitly describes the material domain $\Omega \in \mathbb{R}^3$. The common philosophy of the embedded methods mentioned above is to avoid this often laborious and error-prone process from the outset. Their main idea is to use a simple discretization \mathcal{P}_{bg} , e.g., a regular grid that covers the entire geometrical domain Ω , whereby the domain's boundary $\Gamma = \partial\Omega$ is allowed to intersect elements. In the following, we refer to \mathcal{P}_{bg} as the background mesh and define it as a regular grid with hexahedral integration domains corresponding to the standard discretization used in most embedded boundary methods. Note that this definition applies to classical C^0 continuous finite elements but also includes background meshes spanned by trivariate B-Splines [57]. Due to the simplicity of \mathcal{P}_{bg} and the potentially arbitrary complexity of Ω , cut elements \mathcal{B}_{cut} are inevitable. Uncut elements are classified as interior \mathcal{B}_{in} and exterior \mathcal{B}_{out} based on their location relative to the boundary Γ . The subset $\mathcal{P} = \mathcal{P}_{\text{bg}} \setminus \mathcal{P}_{\text{out}}$ represents the active mesh, where \mathcal{P}_{out} contains all \mathcal{B}_{out} . Consequently, the necessary integration of bulk and surface terms appearing in the weak formulation can be stated as

$$\int_{\Omega} (\cdot) \, d\Omega = \sum_{\mathcal{B} \in \mathcal{P}} \int_{\mathcal{B} \cap \Omega} (\cdot) \, d\Omega, \quad (1)$$

and

$$\int_{\Gamma_{\beta}} (\cdot) \, d\Gamma = \sum_{\mathcal{B} \in \mathcal{P}} \sum_{F_{\beta} \in \Gamma_{\beta}} \int_{F_{\beta} \cap \mathcal{B}} (\cdot) \, d\Gamma, \quad \beta \in \{D, N\}, \quad (2)$$

with F_D and F_N denoting the Dirichlet and Neumann boundary conditions defined on Γ_D and Γ_N , where $\Gamma_D \cup \Gamma_N = \Gamma$ and $\Gamma_D \cap \Gamma_N = \emptyset$. In the following, we will refer to $x_r \in \Gamma$ as boundary vertices or vertices and $x_p \in \mathcal{P}$ as background nodes or nodes for clarity.

One of the biggest challenges of embedded boundary methods is the numerical integration of cut domains. Section 2.2 outlines the quadrature scheme used in this work to evaluate Eqs. (1)–(2). Another common difficulty is poorly conditioned system matrices due to little support of the basis functions within small cut elements. This problem is less critical for direct solvers but often impedes the use of iterative solvers. For huge problems, however, iterative solvers are unavoidable due to their better scalability and lower memory requirements. The authors in [58] provide a comprehensive overview of stabilization techniques and preconditioners enabling the use of iterative solvers for matrices derived from embedded finite element formulations. In the scope of this work, a PARDISO sparse direct solver from the Intel MKL library is used, which can solve the linear system of equations despite small cut elements. Furthermore, due to the implicit domain description in embedded methods, the enforcement of Dirichlet boundary conditions requires weak formulations. All examples in Section 7 use the Penalty method [59].

2.2. Numerical integration of embedded solids with QuESo

Since the present work aims at an optimization workflow for embedded geometries, a robust generation of the unfitted finite element model is essential. To this end, we employ the open-source framework QuESo (Quadrature for Embedded Solids) [32], which provides highly efficient quadrature rules for arbitrarily complex solids described by oriented boundary meshes, e.g., STereoLithography (STL) meshes. The framework reads the boundary mesh and outputs a list of integration points to evaluate Eqs. (1)–(2) for the specified background discretization. QuESo's quadrature rules, geometrical operations, and algorithmic structure are presented in great detail in [33,34]. At first, a generalized flood fill-based classification scheme robustly identifies cut, interior, and exterior

elements. Subsequently, QuESo computes the intersection of each cut element and the boundary Γ given as an STL model. The resulting surface meshes enclosing the cut domains are used to assemble the moment fitting equations. Through the application of the divergence theorem, the moments can be computed up to machine precision, as demonstrated in [34]. Finally, the moment fitting equations are solved by a point elimination algorithm employing a Non-Negative Least Squares (NNLS) solver [33]. The framework guarantees $n_q \leq (p+1)^3$ integration points per cut element, where p is the polynomial degree of the shape functions. Moreover, all integration weights are strictly positive, and point locations are restricted to the material domain.

3. Node-based shape optimization using implicit Vertex-Morphing

In the scope of the present work, we utilize the Nested Analysis and Design (NAND) technique, where design optimization is conducted in an outer loop, and analysis (such as structural analysis) is performed in an inner loop [60]. This means that the PDE solver is nested inside the optimization solver. As a result, the residuals of the system's governing equations, denoted as r , are treated as constraints in the optimization. The described problem is thus mathematically articulated as follows

$$\begin{aligned} \min \quad & f(\mathbf{x}_r, \mathbf{u}) \\ \text{subject to} \quad & \mathbf{p}(\mathbf{x}_r, s_r) = \mathbf{0}, \quad \text{on } \Gamma, \\ & \mathbf{r}(\mathbf{x}_r, \mathbf{u}) = \mathbf{0}, \quad \text{in } \Omega, \\ & h_j(\mathbf{x}_r, \mathbf{u}) = 0, \quad j = 1 \dots m_h, \\ & g_j(\mathbf{x}_r, \mathbf{u}) \leq 0, \quad j = 1 \dots m_g, \end{aligned} \quad (3)$$

where f represents the objective functional that depends on the state variables \mathbf{u} and the surface geometry. The goal is to minimize f with respect to the design variables s_r , which dictate the coordinates of the surface vertices $\mathbf{x}_r \in \Gamma$ through the parameterization operator \mathbf{p} . Section 3.1 introduces the concept of implicit Vertex-Morphing [48], which involves the specific formulation of \mathbf{p} . The functions h_j and g_j are the equality and inequality constraints, respectively.

3.1. Implicit Vertex-Morphing

Filters, whether implicit or explicit, can be used to consistently control or, in other words, regularize the design space. In addition, the associated filter radius allows to prescribe the minimum feature size of the final shape. A comprehensive comparison between implicit and explicit formulations is provided in [48]. In the following, we construct the physical shape $\mathbf{x}_r = (x_r^1, x_r^2, x_r^3)$ from the control field $s_r = (s_r^1, s_r^2, s_r^3)$ through the implicit Helmholtz/Sobolev filtering operation. The essential equations are briefly described in this section. However, for a more detailed discussion, the interested reader is referred to [48,61–63]. Generally, the Helmholtz/Sobolev filtering operation can be expressed through the components p^i of the vector-valued function \mathbf{p} , as per

$$\begin{aligned} p^i &= -(r_r^H)^2 \nabla_r \cdot \nabla_r x_r^i + x_r^i - s_r^i = 0, \quad i \in \{1, 2, 3\}, \quad \text{on } \Gamma, \\ x_r^i &= s_r^i, \quad \text{along } \Gamma_D^H \text{ (fixed boundaries)}, \end{aligned} \quad (4)$$

where $\nabla_r \cdot \nabla_r$ is the Laplace–Beltrami operator. Given a function w defined on some open neighborhood of Γ , the tangential surface gradient ∇_r reads

$$\nabla_r w = \nabla w - (\mathbf{n} \cdot \nabla w) \mathbf{n}, \quad (5)$$

with \mathbf{n} denoting the unit normal vector on Γ and ∇ being the three-dimensional Nabla operator. The filter radius r_r^H in Eq. (4) defines the maximum correlation distance of two boundary vertices. If $r_r^H \rightarrow 0$, the smoothing effect disappears, such that $\mathbf{x}_r = s_r$. In this work, linear shape functions spanned over the triangular elements of Γ discretize Eq. (4). To account for the gradients described in the three Cartesian directions $i \in \{1, 2, 3\}$, trivariate shape functions are introduced. For this purpose, we reconstruct an auxiliary tetrahedron for each triangle and use the traces of the bulk shape functions on the boundary, i.e., triangle, as surface shape functions. This concept for the discretization of surface PDEs is adopted from [64,65]. Furthermore, the boundary $\Gamma_D^H \subset \Gamma$ defines the non-design surfaces that are treated as Dirichlet conditions during optimization. Employing a standard Galerkin-based finite element formulation yields the corresponding filtering-based surface parameterization

$$\mathbf{p}(\mathbf{x}_r, s_r) = \mathbf{K}_r^H \mathbf{x}_r + \mathbf{M}_r^H \mathbf{x}_r - \mathbf{M}_r^H s_r = \mathbf{0}, \quad (6)$$

with the surface mass matrix

$$\mathbf{M}_r^H = \sum_{\Gamma_e} \mathbf{N}_r^T \mathbf{N}_r d\Gamma, \quad (7)$$

and the surface stiffness matrix

$$\mathbf{K}_r^H = (r_r^H)^2 \sum_{\Gamma_e} (\nabla_r \mathbf{N}_r)^T \nabla_r \mathbf{N}_r d\Gamma, \quad (8)$$

where \mathbf{N}_r contains the traces of the bulk shape functions on the boundary. The superscript H indicates that the respective matrices belong to the Helmholtz filtering and not to the problem of linear elasticity. Consequently, we establish the following relation

$$\mathbf{x}_r = \mathbf{A}_r^H s_r; \quad \mathbf{A}_r^H = (\mathbf{K}_r^H + \mathbf{M}_r^H)^{-1} \mathbf{M}_r^H. \quad (9)$$

Note that the matrix $\mathbf{A}_r^H \in \mathbb{R}^{m_d \times m_d}$ is positive definite, where m_d denotes the number of design variables corresponding to three times the number of vertices on the design surface.

3.2. Update rule

To tackle the constrained optimization problem in Eq. (3), we employ an adapted version of Rosen's gradient projection algorithm [66,67], which operates in the control space, as per

$${}^{k+1}s_r = {}^k s_r - \underbrace{\alpha [I - \mathbf{P}(\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T]}_{\text{projection}} \underbrace{\frac{df}{ds_r}|_c}_{\text{correction}} - \underbrace{[\mathbf{P}(\mathbf{P}^T \mathbf{P})^{-1}] g^a}_{\text{correction}}, \quad (10)$$

where α is the projection step size. The matrix $\mathbf{P} \in \mathbb{R}^{m_d \times m_k}$ contains the gradients $dg^a/ds_r|_c$ in a column-wise arrangement, with $g^a \in \mathbb{R}^{m_k}$ being the vector of active constraints. Eq. (10) includes a projection and a correction term. The former term projects the steepest descent direction onto a subspace tangent to the active constraints. Since this projection can still lead to violations in the case of nonlinear constraints, an additional correction term is added. The corresponding correction or restoring direction is multiplied by g^a , see the right term in Eq. (10).

It is clear that the update rule in Eq. (10) relies on the gradients of the objective and the constraints w.r.t. the design variables. Thus, their accurate calculation is paramount for a successful optimization. A general problem of discrete sensitivity analysis is the strong dependency on the finite element discretization used. In contrast to the design variables, which are support values of a continuous field, the discrete sensitivities are integrated values containing size effects of the associated finite elements. For the sake of clarity, we explicitly distinguish between discrete (mesh-dependent) $dJ/dx_r|_d$ sensitivities and continuous (mesh-independent) sensitivity fields $dJ/dx_r|_c$ sampled at the mesh vertices x_r . In the scope of this work, Eq. (10) processes $dJ/dx_r|_c$, resulting in mesh-independent search directions. Based on the derivations in [48], we establish the following filter operator for the continuous gradient fields of the response J

$$\frac{dJ}{ds_r}|_c = \mathbf{A}_r^H \frac{dJ}{dx_r}|_c. \quad (11)$$

Note that J can be any response function, e.g., strain energy, of the finite element problem. In Eq. (10), the responses of interest are the objective functional f , the equality constraints h_j , and the inequality constraints g_j . When the primal solution is solved on a boundary-fitted mesh, Eq. (11) can be reformulated to approximate continuous control gradients $dJ/ds_r|_c$ from discrete sensitivities $dJ/dx_r|_d$, as shown in [48]. In the scope of this work, however, continuous fields are already introduced in the physical space, i.e., $dJ/dx_r|_c$, to facilitate a consistent transfer of information between the finite element discretization (background mesh) and the surface parameterization. The next section illustrates this concept and presents the required steps for calculating $dJ/dx_r|_c$ along immersed boundaries.

4. Sensitivity analysis

The core idea of the present work is to perform gradient-based shape optimization of the immersed boundary Γ (see Section 3) while solving the primal problem

$$\mathbf{K}u = F, \quad (12)$$

with the stiffness matrix \mathbf{K} , solution vector u , and force vector F on the background mesh \mathcal{P} (see Section 2). The result is a complete decoupling of the geometry parameterization and the finite element discretization. On the one hand, this is the key feature that maintains a high mesh quality for large shape updates throughout the optimization. Simultaneously, however, the responses and design variables are no longer defined at the same discrete points, posing severe challenges for the sensitivity analysis. While \mathbf{K} , u , and F are stored on the nodes $x_p \in \mathcal{P}$, the geometric shape is defined by the vertices $x_r \in \Gamma$. Semi-analytical methods [42], which are common strategies for computing shape sensitivities, involve approximations by finite differences. Since Eq. (12) is solved on \mathcal{P} , finite differences can simply be applied to $x_p \in \mathcal{P}$ but not to $x_r \in \Gamma$. The main idea in the following is to first compute the gradients w.r.t. $x_p \in \mathcal{P}$ with semi-analytical methods and subsequently use them for a consistent evaluation of the sensitivities at the immersed boundary Γ . Sections 4.1–4.2 individually describe these two critical steps and provide several examples demonstrating the accuracy of the proposed approach.

4.1. Semi-analytical sensitivity analysis on the background mesh

This section provides a brief introduction to semi-analytical sensitivity analysis, focusing on the most important equations relevant to the publication at hand. We refer the interested reader to [42,43] for a more detailed derivation. Applying the chain rule to the response J enables the straightforward calculation of the sensitivities w.r.t. the nodes of the background mesh

$$\frac{dJ}{dx_p}|_d = \frac{\partial J}{\partial x_p} + \left(\frac{\partial J}{\partial u} \right)^T \frac{du}{dx_p}. \quad (13)$$

We note that for the evaluation of Eq. (13) at a specific node x_p , the response must be integrated over the associated finite elements, resulting in mesh-dependent discrete sensitivity values. Calculating the partial derivatives $\partial J/\partial x_p$ is usually simple and, in many

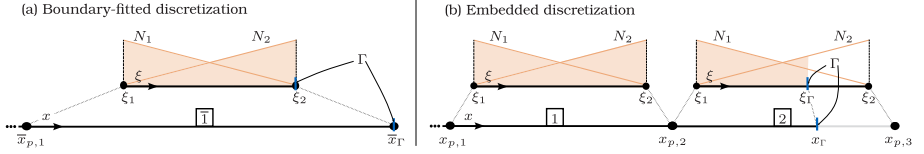


Fig. 1. Boundary-fitted and embedded discretization with one-dimensional linear truss elements.

cases, leads to a zero vector. Given the discrete primal problem in Eq. (12), the derivative of the state variable u w.r.t. the nodal coordinate x_p can be formulated as

$$\frac{du}{dx_p} = \mathbf{K}^{-1} \left(\frac{dF}{dx_p} - \frac{d\mathbf{K}}{dx_p} u \right) \quad (14)$$

The substitution of Eq. (14) into Eq. (13) results in the equation for analytical sensitivity analysis

$$\frac{dJ}{dx_p} \Big|_d = \frac{\partial J}{\partial x_p} + \left(\frac{\partial J}{\partial u} \right)^T \mathbf{K}^{-1} \left(\frac{dF}{dx_p} - \frac{d\mathbf{K}}{dx_p} u \right) = \frac{\partial J}{\partial x_p} + \left(\frac{\partial J}{\partial u} \right)^T \mathbf{K}^{-1} F^*, \quad (15)$$

with F^* being the so-called pseudo load vector. We follow a semi-analytical approach, where the pseudo load vector is evaluated by finite differences, as per

$$F^* = \frac{F(x_p + \Delta x_p) - F(x_p)}{\Delta x_p} - \frac{\mathbf{K}(x_p + \Delta x_p) - \mathbf{K}(x_p)}{\Delta x_p} u. \quad (16)$$

Considering the strain energy $U = \frac{1}{2} u^T \mathbf{K} u$ as the response and $\frac{\partial U}{\partial x_p} = 0$, Eq. (15) simplifies to

$$\frac{dU}{dx_p} \Big|_d = u^T F^*. \quad (17)$$

The crucial aspect of the proposed sensitivity analysis is that Eq. (15) is computed with respect to the nodes of the background mesh $x_p \in \mathcal{P}$, yielding the gradients $dJ/dx_{p|d}$. Eqs. (16)–(17) reveal that, in the case of $J = U$, the respective calculation only requires evaluating the perturbed system, i.e., $F(x_p + \Delta x_p)$ and $\mathbf{K}(x_p + \Delta x_p)$. Nevertheless, for subsequent shape optimization, the gradients must be computed w.r.t. the immersed boundary, i.e., $x_f \in \Gamma$, and not w.r.t. the nodes of the background mesh. Furthermore, these gradients should be independent of the finite element discretization. To this end, the next subsection establishes a consistent relation between $dJ/dx_{p|d}$ and $dJ/dx_{f|c}$.

4.2. Sensitivity weighting and interpolation at the embedded boundary

As discussed above, the gradients $dJ/dx_{p|d}$ can be computed with the same methods used for classical boundary-fitted discretizations. However, standard response functions, such as strain energy, volume, etc., are integrated values, which means that their discrete gradients depend on the size of the associated finite elements [43,48]. In embedded methods, the intersection between the boundary Γ and the background element $B \in \mathcal{P}$ determines the area of integration. As a result, the cut domain's location, shape, and topology affect the sensitivities when evaluating Eqs. (15)–(16). In the following, we recover a continuous gradient field that is independent of the location of the boundary Γ within \mathcal{P} . This allows the gradients to be calculated w.r.t. any vertex x_f located on Γ . Three examples in one- two- and three dimensions discuss the mentioned phenomenon and present the required gradient scaling and interpolation procedure in detail. Our approach adopts the sensitivity weighting introduced in the scope of isogeometric shape optimization of shells [43] and extends it for application to embedded boundary methods.

4.2.1. One-dimensional example

Fig. 1 depicts a simple one-dimensional truss model in two different configurations. In both cases, x_f denotes the location of the boundary Γ . Given the response J , we aim to find the derivative $dJ/dx_{f|c}$. For illustration purposes, the structure's volume V serves as the response function below.

(a) Boundary-fitted discretization

Firstly, let us consider the boundary-fitted discretization in Fig. 1(a), where the nodes of the FE mesh explicitly describe the boundary Γ . Note that the coordinates associated with the boundary-fitted mesh are marked by an *overline*, e.g., \bar{x} . The volume of the structure is given as

$$\bar{V} = A (\bar{x}_\Gamma - \bar{x}_{p,1}), \quad (18)$$

where A denotes the surface area of the cross-section. Consequently, the gradient of \bar{V} with respect to the position \bar{x}_r of the boundary Γ yields

$$\left. \frac{d\bar{V}}{d\bar{x}_r} \right|_d = A. \quad (19)$$

It becomes evident that the sensitivity value contains information about the element's cross-section. Due to the simplicity of the given example, the result in Eq. (19) does not necessarily cause problems. However, for demonstration purposes, we may normalize Eq. (19) with the surface area A to obtain a mesh-independent unit gradient, as per

$$\left. \frac{1}{A} \frac{d\bar{V}}{d\bar{x}_r} \right|_d = 1. \quad (20)$$

Eq. (20) shall serve as a reference solution for the discussion in the next paragraph.

(b) Embedded discretization

We now investigate the embedded discretization, as depicted in Fig. 1(b). The volumes of elements 1 and 2 read

$$V^1 = A(x_{p,2} - x_{p,1}), \text{ and } V^2 = A(x_r - x_{p,2}). \quad (21)$$

Since the internal boundary's position is described by $x_r = N_1(\xi_r)x_{p,2} + N_2(\xi_r)x_{p,3}$, we rewrite Eq. (21) as

$$V^1 = A(x_{p,2} - x_{p,1}), \text{ and } V^2 = A(N_1(\xi_r)x_{p,2} + N_2(\xi_r)x_{p,3} - x_{p,2}). \quad (22)$$

Given that the entire volume V of the structure is the accumulated volume $V = V^1 + V^2$, the discrete gradients with respect to the coordinates $x_{p,2}$ and $x_{p,3}$ are

$$\left. \frac{dV}{dx_{p,2}} \right|_d = \left. \frac{dV^1}{dx_{p,2}} \right|_d + \left. \frac{dV^2}{dx_{p,2}} \right|_d = AN_1(\xi_r), \quad (23)$$

$$\left. \frac{dV}{dx_{p,3}} \right|_d = \left. \frac{dV^1}{dx_{p,3}} \right|_d + \left. \frac{dV^2}{dx_{p,3}} \right|_d = AN_2(\xi_r). \quad (24)$$

Eqs. (23) and (24) reveal that the gradients are a function of ξ_r and hence depend on where element 2 is cut. Note that our final goal is to interpolate the gradients of the background nodes at the embedded boundary Γ . However, a direct interpolation of the gradients in Eqs. (23)–(24) at x_r will inevitably propagate the dependency on ξ_r . This clearly contradicts the result in Eq. (19) obtained from the boundary-fitted discretization, which also depends on the surface area A but not on the element's length. Furthermore, the reference value is a complete mesh-independent unit gradient with respect to x_r , see Eq. (20). The only possibility of always obtaining $dV/dx_r = 1$ by interpolating $dV/x_{p,2}$ and $dV/x_{p,3}$ is that both gradients at the background nodes are themselves equal to one. Therefore, in this particular example, a consistent sensitivity analysis must ensure that $dV/dx_{p,2} = 1$ and $dV/dx_{p,3} = 1$ for any value of ξ_r . In line with the gradient weighting proposed in [43] to eliminate the influence of the control points on the geometry of a NURBS-based shell surface, we introduce the following weighting factors at nodes $x_{p,2}$ and $x_{p,3}$

$$\mathcal{N}_{x_{p,2}} = \int N_1 d\Gamma = AN_1(\xi_r), \quad (25)$$

$$\mathcal{N}_{x_{p,3}} = \int N_2 d\Gamma = AN_2(\xi_r). \quad (26)$$

Note that in this one-dimensional example, the boundary Γ , which defines the domain of integration in Eqs. (25)–(26), is the cross-section of the truss. Eqs. (25)–(26) can be interpreted as a measure of how much a perturbation of the coordinates $x_{p,2}$ or $x_{p,3}$ affects the boundary Γ . The respective sensitivity weighting yields

$$\frac{1}{\mathcal{N}_{x_{p,2}}} \left. \frac{dV}{dx_{p,2}} \right|_d = 1, \quad (27)$$

$$\frac{1}{\mathcal{N}_{x_{p,3}}} \left. \frac{dV}{dx_{p,3}} \right|_d = 1. \quad (28)$$

Finally, we can interpolate Eqs. (27) and (28) via the shape functions N_1 and N_2 at the internal boundary ξ_r

$$\left. \frac{dV}{dx_r} \right|_c = N_1(\xi_r) \frac{1}{\mathcal{N}_{x_{p,2}}} \left. \frac{dV}{dx_{p,2}} \right|_d + N_2(\xi_r) \frac{1}{\mathcal{N}_{x_{p,3}}} \left. \frac{dV}{dx_{p,3}} \right|_d = 1, \quad (29)$$

which results in an expected constant unit gradient. The next subsections generalize the presented weighting and interpolation procedure to two- and three-dimensional problems.

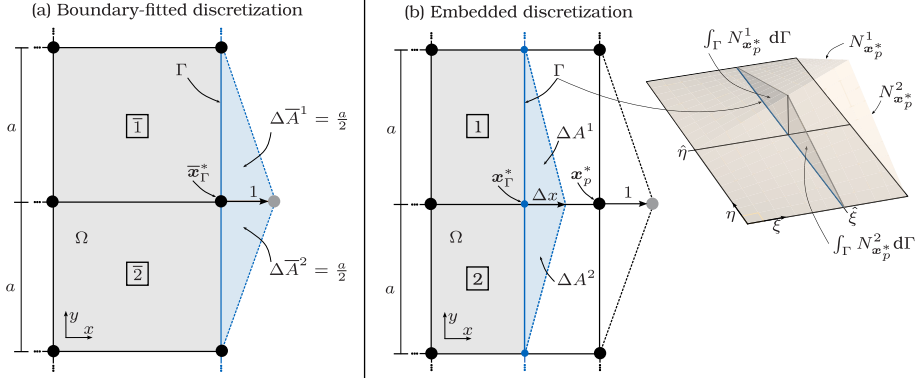


Fig. 2. Boundary-fitted and embedded discretization with bilinear quadrilateral elements.

4.2.2. Two-dimensional example

Similar to the previous example, this section studies a classical boundary-fitted mesh before discussing the procedure for embedded discretizations. In both cases, we assume bilinear quadrilateral finite elements and aim to calculate the gradients of the volume V with respect to the vertices $\mathbf{x}_r \in \Gamma$.

(a) Boundary-fitted discretization

Fig. 2(a) shows a section of a standard finite element mesh composed of quadrilateral elements whose right edges explicitly describe the boundary Γ . This example quantifies the influence of the x -component of node $\bar{\mathbf{x}}_r^* = \{\bar{x}_r^*, \bar{y}_r^*\}$ on the structure’s volume, see Fig. 2(a). Therefore, a unit displacement in x -direction is applied on the corresponding node. This perturbation can be interpreted as the finite difference Δx_p in Eq. (16). Based on the observed change in area, the gradients are

$$\left. \frac{dV}{d\bar{x}_r^*} \right|_d = t (\Delta A^1 + \Delta A^2) = ta, \tag{30}$$

where t denotes the thickness of the two-dimensional elements. As in the one-dimensional example, the sensitivity value in Eq. (30) contains mesh-dependent information. Consequently, the gradient may again be normalized by the respective element’s dimensions

$$\frac{1}{ta} \left. \frac{dV}{d\bar{x}_r^*} \right|_d = 1. \tag{31}$$

(b) Embedded discretization

Fig. 2(b) perturbs node x_p^* of the background mesh \mathcal{P}_{bg} in the embedded configuration by a unit displacement, which again imitates the finite difference Δx_p in Eq. (16). This example illustrates that the maximum perturbation Δx of the boundary Γ is no longer 1 but $N_{x_p^*}^1(\xi, \eta)$. Note that both shape function values, $N_{x_p^*}^1(\xi, \eta)$ and $N_{x_p^*}^2(\xi, \eta)$, are equal at this particular point. Consequently, the gradient with respect to the x -coordinate of node x_p^* is

$$\left. \frac{dV}{dx_p^*} \right|_d = t (\Delta A^1 + \Delta A^2) = t \left(N_{x_p^*}^1(\xi, \eta) \frac{a}{2} + N_{x_p^*}^2(\xi, \eta) \frac{a}{2} \right). \tag{32}$$

A closer look at Fig. 2(b) reveals that Eq. (32) equals the integral of all shape functions associated with node x_p^* evaluated over the boundary Γ

$$\mathcal{N}_{x_p^*} = \int_{\Gamma} N_{x_p^*}^1 d\Gamma + \int_{\Gamma} N_{x_p^*}^2 d\Gamma. \tag{33}$$

Let us remember that the reference solution in Eq. (31) is a constant unit gradient. To achieve the same result, we apply the weighting analogous to the one-dimensional case in Section 4.2.1

$$\frac{1}{\mathcal{N}_{x_p^*}} \left. \frac{dV}{dx_p^*} \right|_d = 1. \tag{34}$$

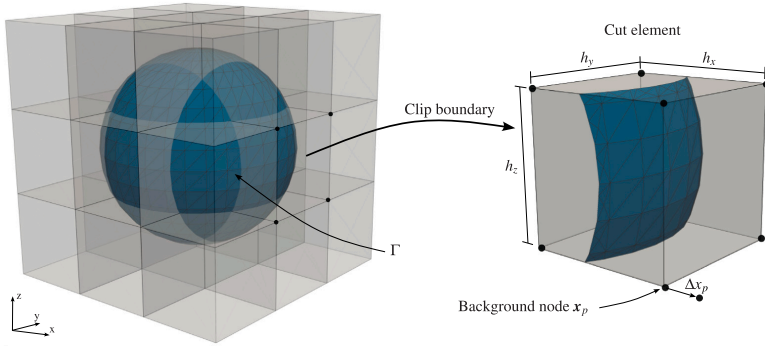


Fig. 3. Clipped boundary of an exemplary cut element. The clipping operation is performed by QuESo [32]. The resulting triangle mesh is used to calculate the weighting factors in Eq. (37).

In fact, by following this procedure, we obtain unit gradients for all nodes x_p depicted in Fig. 2(b). Consequently, they can again be used to consistently calculate the shape sensitivity w.r.t. the vertex coordinate x_r^* on the immersed boundary Γ

$$\frac{dV}{dx_r^*} \Big|_c = \sum_{i=1}^{n=4} N_{x_{p,i}}(\xi, \eta) \left(\frac{1}{\mathcal{N}_{x_{p,i}}} \frac{dV}{dx_{p,i}} \Big|_d \right) = 1, \tag{35}$$

where i are the indices of the nodes connected to the element containing x_r^* . Since, in this particular case, x_r^* is located on the boundary between elements 1 and 2, both can be used to evaluate Eq. (35). In any case, the result is a mesh-independent unit gradient.

4.2.3. Generalization and application to three-dimensional problems

The following paragraph provides the generalized equations for the sensitivity analysis introduced in Sections 4.2.1–4.2.2, along with a consistency check for three-dimensional solid geometries. In accordance with the above derivations, the continuous gradient field $dJ/dx_r|_c$ sampled at the boundary vertices $x_r \in \Gamma$ can be expressed as

$$\frac{dJ}{dx_r} \Big|_c = \sum_{i \in I} N_{x_{p,i}} \left(\frac{1}{\mathcal{N}_{x_{p,i}}} \frac{dJ}{dx_{p,i}} \Big|_d \right), \tag{36}$$

where I is the set of indices of all background nodes $x_{p,i}$ with support at x_r . The corresponding weighting factors read

$$\mathcal{N}_{x_{p,i}} = \sum_{j \in \mathcal{J}} \int_{\Gamma} N_{x_{p,i}}^j d\Gamma, \tag{37}$$

with \mathcal{J} denoting the indices of all basis functions associated with node $x_{p,i}$. In the scope of this work, Eq. (37) is evaluated by quadrature rules on the triangles that discretize the boundary Γ . Generally, shape functions are piece-wise polynomials defined on element level. Thus, for an accurate evaluation by means of numerical integration, we decompose Γ into subsections that conform to the background mesh. This means that no triangle crosses the element boundaries of \mathcal{P} . The open-source code QuESo [32] performs the necessary clipping operations. For more information on the respective algorithms, the interested reader is kindly referred to [34]. Fig. 3 shows the clipped mesh that serves as surface parameterization to evaluate all active shape functions of the associated background nodes, see Eq. (37).

The consistency of the proposed sensitivity analysis is demonstrated in the following. For this purpose, we investigate a sphere and compute the gradients $dV/dx_r|_c$ of its volume V w.r.t. $x_r \in \Gamma$. Due to the symmetrical nature of the sphere, the gradients $dV/dx_r|_c$ must be uniformly distributed in the direction of the surface normals. In order to illustrate the influence of the presented weighting scheme, Eq. (36) is additionally evaluated by omitting the weighting factors, such that $\forall i \in I : (\mathcal{N}_{x_{p,i}} = 1)$. The resulting gradients are denoted as $dV/dx_r|_d$. Fig. 4 shows the magnitudes of $dV/dx_r|_d$ and $dV/dx_r|_c$, along with the used background mesh. Note that the gradients $dV/dx_r|_d$ w.r.t. the background nodes $x_p \in \mathcal{P}$ are computed by finite differences in analogy to the formulation introduced in Eq. (16). Fig. 4(b) reveals that the gradients $dV/dx_r|_d$ without weighting exhibit severe disturbances, which depend on the intersections between the boundary Γ and the cut elements $\mathcal{B}_{cut} \in \mathcal{P}$. By contrast, the results in Fig. 4(c), calculated with the appropriate weighting factors (Eq. (37)), show the expected uniform distribution. Moreover, since $dV/dx_r|_c$ is obtained through interpolation of gradients evaluated at the background nodes x_p , the respective field is entirely independent of the surface mesh representing Γ . It is also worth emphasizing that the proposed sensitivity analysis is not restricted to classical trilinear

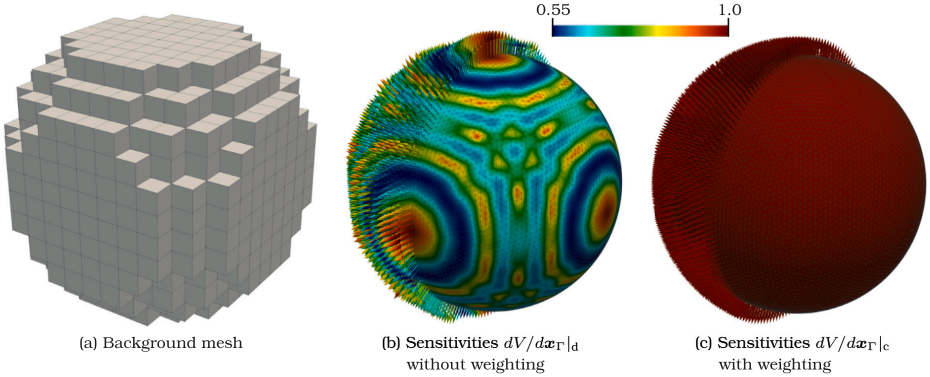


Fig. 4. Influence of the proposed weighting procedure (see Eqs. (36)–(37)) on the volume sensitivities evaluated at the embedded boundary of a sphere. The gradients $dV/x_r|_d$ are calculated by omitting the weighting factors in Eq. (36), such that $\forall i \in I : (\mathcal{N}_{x_r}^i = 1)$, while the gradients $dV/x_r|_c$ are evaluated via Eq. (36) using the weighting factors stated in Eq. (37). The color scheme indicates the magnitude of the gradients. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

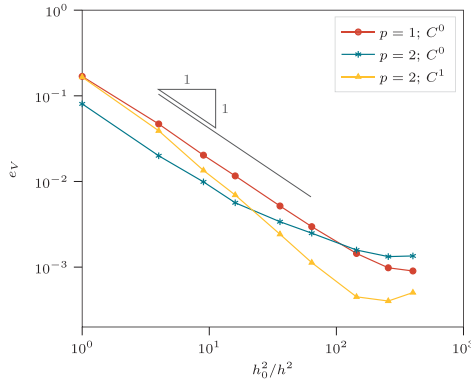


Fig. 5. Relative error e_V (see Eq. (38)) under h -refinement of the background mesh for different polynomial degrees p and continuities C^l .

hexahedral finite elements but can straightforwardly be applied to discretizations of higher order or higher continuity. Fig. 5 plots the relative error of $dV/dx_r|_c$ under h -refinement of three different background meshes. The error is computed as follows

$$e_V = \frac{\|V_{\text{grad}} - N\|_{L2}}{\|N\|_{L2}}, \tag{38}$$

where V_{grad} is a vector containing the gradients $dV/dx_r|_c$ evaluated at each vertex $x_r \in \Gamma$. Every gradient is compared to the respective unit normal $n \in N$ of the sphere, which serves as a reference solution. Fig. 5 shows the results using linear finite elements and quadratic C^0 and C^1 continuous B-spline bases for the background discretization \mathcal{P} . In all cases, e_V monotonically decreases as \mathcal{P} is refined. The finite differences employed to compute $dV/dx_r|_d$ suggest linear convergence rates. However, we observe that the graphs tend towards $O(h^2)$, where $h = h_x = h_y = h_z$ is the element’s edge length. To understand this phenomenon, let us consider the perturbation Δx_p of $x_p \in \mathcal{P}$ depicted in Fig. 3. If x_p moves along the x -axis, the affected section of the boundary Γ is proportional to $A \sim h_y h_z$. Halving the element size, therefore, leads to a quartering of the region approximated by the finite differences. As a result, the gradients $dV/dx_r|_c$ follow a convergence rate of $O(h^2)$.

Finally, we investigate the strain energy gradients $dU/dx_r|_c$ w.r.t. the boundary vertices x_r of a cantilever beam with a rectangular cross-section. Fig. 6(a) shows the problem setup, including the geometry, the applied boundary conditions, and the resulting deformation pattern. Initially, the problem is solved on an axis-aligned background mesh \mathcal{P} , as depicted in Fig. 6(b),

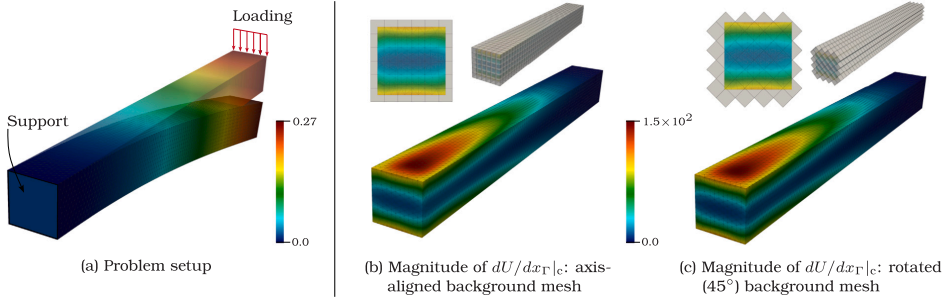


Fig. 6. Strain energy sensitivities $dU/dx_{\Gamma}|_c$ evaluated at the embedded boundary of a cantilever with rectangular cross-section: Comparison between axis-aligned and 45° rotated background mesh. In both examples, the background mesh is spanned by quadratic C^1 continuous B-Splines.

yielding a smooth sensitivity field. Subsequently, \mathcal{P} is rotated by 45° to investigate the influence of the background mesh's orientation on the gradients $dU/dx_{\Gamma}|_c$. Fig. 6(c) reveals that the previous gradient field remains unchanged despite irregular intersections between the cut elements and the boundary Γ .

In summary, the proposed sensitivity analysis stated in Eqs. (36)–(37) guarantees reproducible and consistent results that are independent of the surface mesh Γ and the background discretization \mathcal{P} .

5. Geometrical constraints

A common requirement in shape optimization is compliance with geometric constraints, which arise, e.g., from the manufacturing or assembly process of the individual part. In this context, minimum wall thicknesses and design space restrictions are particularly important, posing major challenges for robust optimization workflows. A direct approach involves formulating the constraints mathematically and incorporating them individually in the optimization problem [68]. However, given that these geometrical constraints are inherently localized or point-wise, the solution must typically consider a significant number of constraints. This potentially leads to as many constraints as there are design variables, placing a substantial burden on the optimizer. Alternative approaches combine the restrictions in a single function [69], which yields a more efficient implementation but also a weaker enforcement. A systematic and promising solution to address these constraints is to incorporate them explicitly into the parameterization and hence prevent the generation of non-feasible geometries by definition. The following two subsections discuss how minimum wall thickness and design space constraints are detected and enforced in our workflow. Finally, Section 5.3 deals with the smooth transition from design to non-design surfaces.

5.1. Active constraint detection

The formulations for compliance with specified minimum wall thicknesses and feasible design spaces are fundamentally similar. In both cases, we shoot a ray [70] for each vertex $x_r \in \Gamma$ in normal direction n and test for intersection with a constraining boundary Γ_c . Thereby, the vertex normals n are computed by the averaged orientation of their adjacent triangles and normalized to unit length. The distinction between the different constraint types manifests in the specific boundary Γ_c used to detect the active vertices. For the minimum wall thickness constraint, Γ_c is defined as the actual boundary Γ of the structure, while the design space constraint requires the introduction of an auxiliary boundary delimiting the feasible domain. In the latter case, we read Γ_c from an additional STL model. Fig. 7 schematically demonstrates the detection of the two constraint types and illustrates the definition of Γ_c . A vertex $x_r \in \Gamma$ is considered to be active if the distance d between x_r and Γ_c in direction n is smaller than

$$\begin{aligned} d &< 2\Delta x_{\max} + t_{\min}; && \text{for wall thickness constraints,} \\ d &< 2\Delta x_{\max}; && \text{for design space constraints,} \end{aligned} \tag{39}$$

where t_{\min} is the enforced minimum wall thickness and Δx_{\max} is the maximum vertex shape update of the previous iteration. Thus, the term $2\Delta x_{\max}$ serves as a buffer to prevent the unnoticed violation of the constraints. As depicted in Fig. 7, the direction of the associated normal vectors, i.e., inward-pointing or outward-pointing, depends on the constraint type. QuESo [32], introduced in Section 2.2, performs the necessary ray triangle intersection tests. The underlying algorithmic structure, which allows efficient processing of large surface meshes, is presented in [34].

5.2. Constraint enforcement

Interpolated or fixed non-design areas can readily be included in the implicit filtering by applying Dirichlet conditions on the corresponding boundaries, see Eq. (4). However, enforcing additional geometrical constraints, i.e., prescribed wall thicknesses or

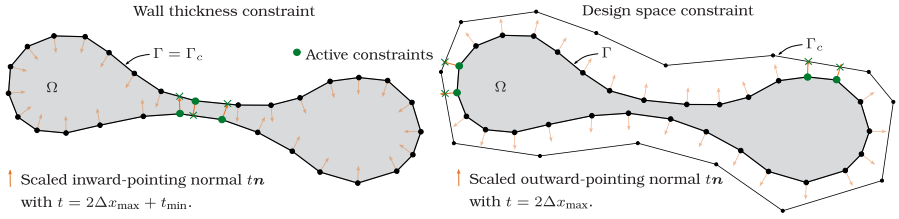


Fig. 7. Detection of active geometrical constraints.

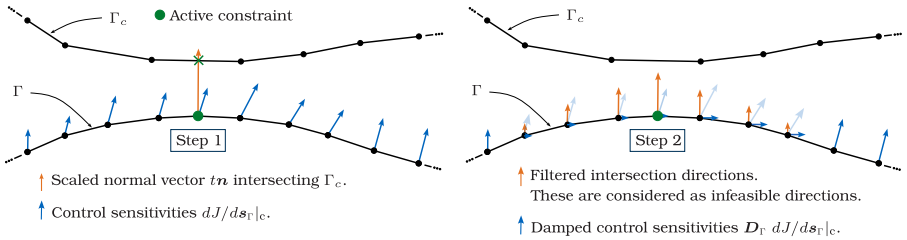


Fig. 8. Damping of control sensitivities. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

feasible design spaces, is more challenging. In the scope of this work, we systematically dampen the control gradients $dJ/ds_r|_c$ that point toward an infeasible direction by projecting them onto a tangential subspace, as per

$$\frac{dJ}{ds_r|_c}^{\text{proj}} = D_r \left. \frac{dJ}{ds_r|_c} \right|_c, \text{ with } D_r = [I - N(N^T N)^{-1} N^T]. \quad (40)$$

Here, D_r is referred to as the damping matrix, and $N \in \mathbb{R}^{3m_f \times m_c}$ denotes a block diagonal matrix, where each $3 \times m_{c,i}$ block contains one ($m_{c,i} = 1$) or multiple ($m_{c,i} > 1$) infeasible directions enforced at vertex i . All other entries of N are zero. Furthermore, m_f represents the number of all vertices $x_r \in \Gamma$, and m_c is the total number of active constraints, i.e., the total number of infeasible directions.

The main idea to enforce minimum wall thickness and design space constraints is to use the active normal vectors n , which satisfy the condition in Eq. (39), as the infeasible directions in Eq. (40). However, since the filtering generally smooths the gradients over a certain distance, we must attenuate the entire surrounding of an actively constrained vertex that lies within the range of the filter radius. In order to find the respective neighbors, the filter function A_r^H is applied to the normal vectors intersecting Γ_c , as indicated by the orange arrows in Fig. 8. Subsequently, all filtered intersection directions are assembled in matrix N in Eq. (40). To numerically limit the affected neighborhood to a reasonable size, only filtered directions are considered where the length of the vector is larger than a certain threshold defined as $\delta \approx 0.01 - 0.05$. Fig. 8 illustrates the overall damping process. Consequently, we modify the parameterization formulation (Eq. (9)) and respectively the sensitivity filtering (Eq. (11)) as follows

$$x_r = A_r^H D_r s_r, \quad (41)$$

and

$$\left. \frac{dJ}{ds_r|_c} \right|_c = D_r A_r^H \left. \frac{dJ}{dx_r|_c} \right|_c. \quad (42)$$

It is important to recognize that while the modified sensitivity and filtering equations are formulated in the global matrix–vector form, the damping matrix can be constructed and applied locally to the gradients at each vertex.

5.3. Smooth transition from design to non-design surfaces

As discussed above, the implicit filter function inherently allows the introduction of non-design surfaces by applying Dirichlet boundary conditions in Eq. (4). However, since the vertices adjacent to the non-design surface are free to move, kinks can potentially occur at the transition zone. While this is a desired feature in some cases, other problems may prefer a generally smooth design. Therefore, our workflow provides the additional option to dampen the gradients in the vicinity of non-design surfaces. Similar to the approach introduced in Section 5.2, the filter function is employed to find the respective neighborhood. For this purpose, an

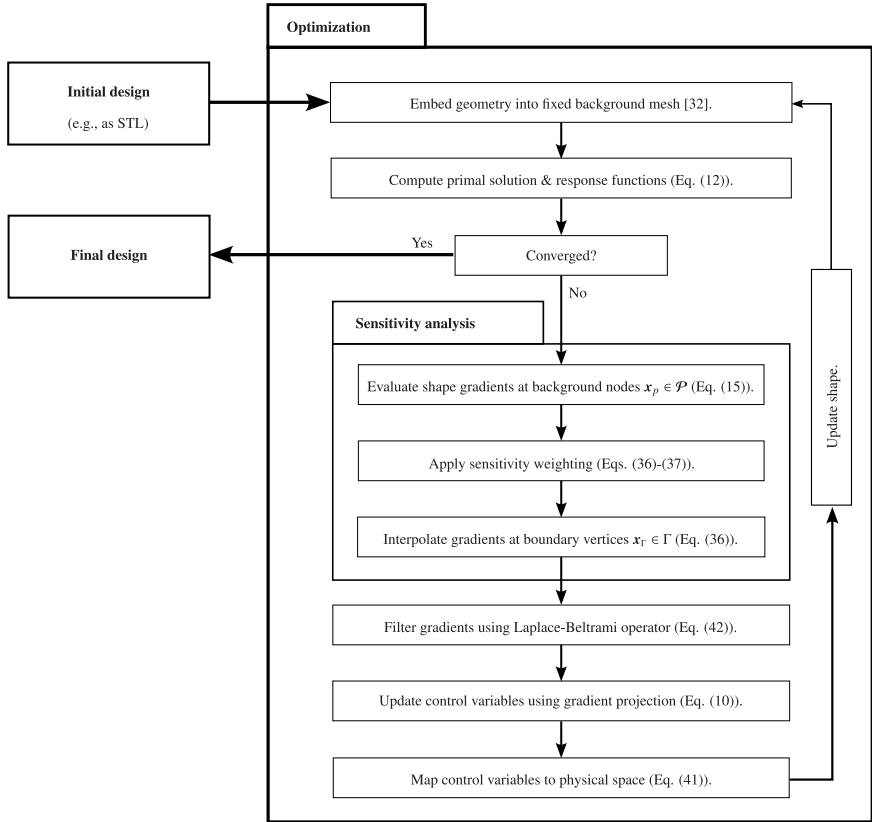


Fig. 9. Workflow of the node-based shape optimization of embedded solids.

auxiliary field, e.g., unit vector field, is introduced on the non-design surfaces and propagated over the actual design surface through the filter function. Subsequently, we eliminate, i.e., set to zero, all control gradients $dJ/ds_r|_c$ wherever the magnitude of the filtered unit gradient field is larger than the previously introduced numerical limit $\delta \approx 0.01 - 0.05$.

6. Optimization workflow

The main feature of the presented optimization strategy is that the geometry representation and the finite element discretization are independent of each other. This allows the structure's boundary Γ to move freely through a fixed background mesh during each shape update. As a result, mesh distortion problems are eliminated from the outset. For clarity, Fig. 9 combines the necessary steps discussed in Sections 2–5 into a complete optimization workflow.

7. Numerical experiments

This section will demonstrate the potential of the proposed workflow for optimizing the shape of complex solid structures described in boundary representation. The first subsection examines the accuracy of the sensitivity analysis proposed in Section 4 using the example of a simple cube. Subsequently, Sections 7.2–7.3 present examples of industrial complexity, including mass and strain energy minimization problems under multiple constraints. All numerical experiments are conducted with the open-source FE framework Kratos Multiphysics [71–73], which is fed with the quadrature rules of QuESo [32–34].

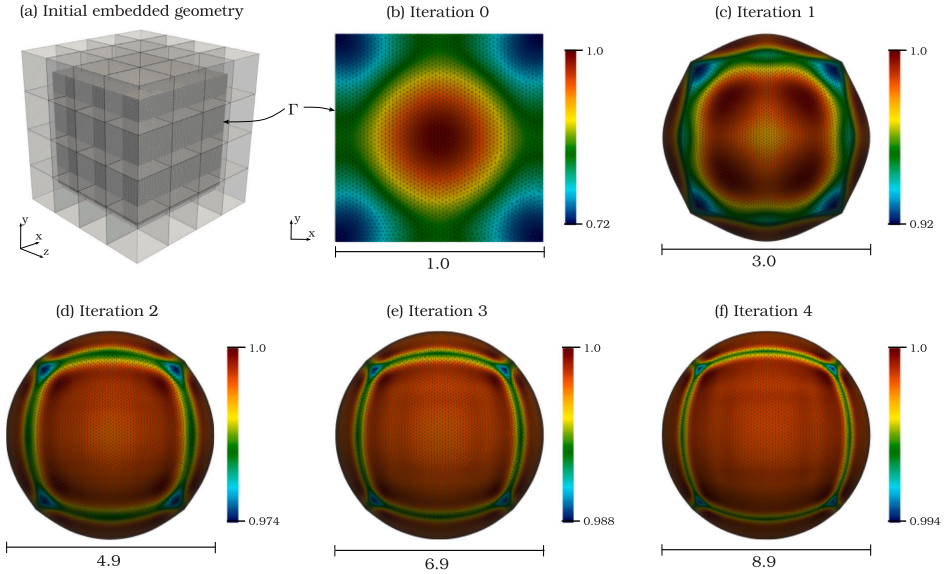


Fig. 10. First four iterations of a volume maximization problem of an embedded cube. The color scheme shows the magnitude of the volume sensitivities $dV/dx_r|_c$. The gradients are directly used to update the positions of the boundary vertices x_r , and no additional filtering is applied, see Eq. (43). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.1. Volume maximization of a cube

To study the quality of the computed gradient field presented in Section 4, we investigate a simple cube with an edge length of $l = 1$, as shown in Fig. 10(a). Its geometrical boundary is embedded into a background mesh spanned by quadratic C^1 continuous B-Splines, where each knot span's length is $h = 1/3$. The objective is to maximize the volume of the initial cube. Note that considering the negative objective functional transforms the formulation into a minimization problem, as stated in Eq. (3). In order to demonstrate the consistency and smoothness of the computed gradient field, this example omits the filtering in Eqs. (9) and (11) and directly manipulates the boundary vertices x_r in the physical space using the unfiltered sensitivities $dJ/dx_r|_c$. The respective shape update straightforwardly reads

$$x_r^{k+1} = x_r^k + \alpha \left. \frac{dJ}{dx_r} \right|_c, \tag{43}$$

where α denotes a constant step size, which is set to $\alpha = 1$.

Fig. 10(b)–(f) show that the initial cube evolves to a nearly perfect sphere after very few iterations. This is an expected result because the sphere has the largest volume compared to all shapes with the same surface area. The color scheme visualizes the magnitudes of the sensitivity gradients $dJ/dx_r|_c$, which approach the value one as the geometry converges to a sphere. Since the gradients on the boundary surface $dJ/dx_r|_c$ are governed by the gradients of the nodes of the background mesh, the corresponding shape functions guarantee a smooth field, as can be seen in Fig. 10. Moreover, the sensitivities are completely independent of the discretization of the surface mesh. Fig. 11(a) plots the relative error of the radius computed as

$$e_R = \frac{\max\{R\} - \min\{R\}}{\text{avg}\{R\}}, \text{ with } \text{avg}\{R\} = \frac{\sum_{R \in \mathbf{R}} R}{|\mathbf{R}|} \tag{44}$$

where $R \in \mathbf{R}$ is the distance between each vertex on the boundary to the center of the initial cube. Eventually, all values $R \in \mathbf{R}$ are expected to be equal, representing the radius of the sphere. Fig. 11(a) reveals a monotonic reduction of e_R during the first nine optimization steps. In fact, after only four iterations, the maximum relative error drops below one percent. Next to it, Fig. 11(b) depicts the current volume over the average radius $\text{avg}\{R\}$, which grows at an expected cubic rate during the optimization.

We conclude that the sensitivity analysis presented in Section 4 enables the computation of shape gradients that are independent of the discretization of the boundary mesh. In addition, the direct use of these gradients as shape updates yields smooth geometries.

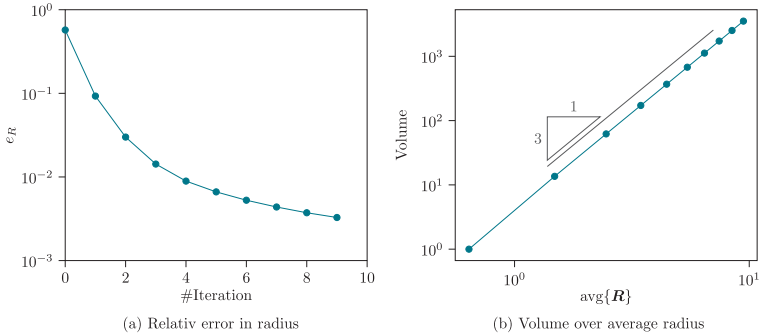


Fig. 11. Volume maximization of an embedded cube. (a) Relative error in radius as defined in Eq. (44) and (b) volume over the average radius $\text{avg}\{R\}$ four the first nine iterations. The background mesh is spanned by quadratic C^1 continuous B-Splines and a knot span size of $h = 1/3$.

7.2. Strain energy minimization of a hook

The second study investigates the performance of the entire workflow proposed in Section 6 using the example of a three-dimensional hook. Fig. 12 shows the problem setup, including geometrical description, boundary conditions, and design space constraint. Due to the deliberately simple initial geometry, large shape updates are to be expected, which generally pose severe challenges for the shape optimization of solid structures. This example is intended to show that the presented methodology provides accurate primal solutions despite significant changes in geometry.

We employ C^1 continuous B-Spline bases on the background mesh with a uniform knot span size of $h = 3$ mm. The Dirichlet conditions $\bar{u} = [0, 0, 0]$ are enforced on Γ_D with the Penalty method [59] and a penalty factor of $\beta = 10^{12}$ N/mm³. Two forces, $F_1 = 10$ kN and $F_2 = 5$ kN, as shown in Fig. 12, load the structure. The Young’s modulus $E = 2.1 \times 10^5$ N/mm², Poisson’s ratio $\nu = 0.3$, and density 7.85×10^{-6} kg/mm³ define the linear elastic material. This example aims to minimize the strain energy U while retaining the initial mass m^{init} . In analogy to Eq. (3), the optimization problem reads

$$\begin{aligned}
 &\min && U, \\
 &\text{subject to} && p = 0, && \text{on } \Gamma, \\
 &&& r = 0, && \text{in } \Omega, \\
 &&& m - m^{\text{init}} = 0,
 \end{aligned} \tag{45}$$

where m is the current mass. Additionally, the geometrical constraints introduced in Section 5 enforce a minimum material thickness of 8 mm, compliance with the design space, and a smooth transition from design to non-design surface with $\delta = 0.04$.

Fig. 13(a)–(b) compare the shape of the initial and optimized geometry, which is obtained with a filter radius of $r_f^H = 5$ mm. It is shown that the cross-section undergoes significant thinning orthogonal to the z -axis. However, during the optimization process, the material thickness constraint becomes active and limits the minimum distance between the two opposite boundaries to 8.13 mm in the final geometry. The geometry also approaches the boundary of the design space but does not violate it, as shown in Fig. 13(b). Fig. 14(a)–(b) plot the graphs of the objective and the constraint for each iteration. The strain energy follows a monotonic reduction and approaches a plateau, indicating that a local minimum has been found. In the last iteration, the relative change in the objective is less than 0.1%, which is defined as the convergence criterion. Over the entire course of the optimization, the maximum violation of the mass constraint is 1.05%. Compared to the original design, the final geometry is characterized by a 33.0% reduction in strain energy with only a 0.17% increase in mass.

Fig. 14(c) shows the convergence in strain energy under h -refinement of the background mesh for the initial and the final geometry. Each marker represents the results of an individual FE analysis, whereby the discretization is successively refined. The element size of the initial and coarsest mesh is referred to as h_0 . The dashed lines indicate the discretization used for this optimization process ($h = 3$ mm). It becomes evident that a highly accurate finite element solution is maintained until the last iteration. Thus, the discretization quality of the background mesh is shown to be completely unaffected by the shape updates.

7.3. Mass minimization of a jet engine bracket

This section shall demonstrate the robustness of our workflow for optimizing structures of industrial complexity. In the following, we examine the second place winning geometry of the General Electric (GE) jet engine bracket challenge.¹ The corresponding design

¹ <https://grabcad.com/challenges/ge-jet-engine-bracket-challenge/results>.

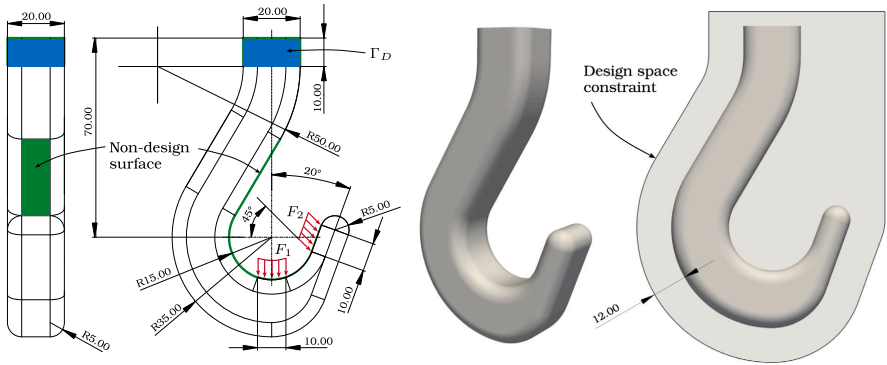


Fig. 12. Initial design, boundary conditions, and design space constraint of a three-dimensional hook. Dimensions are given in [mm].

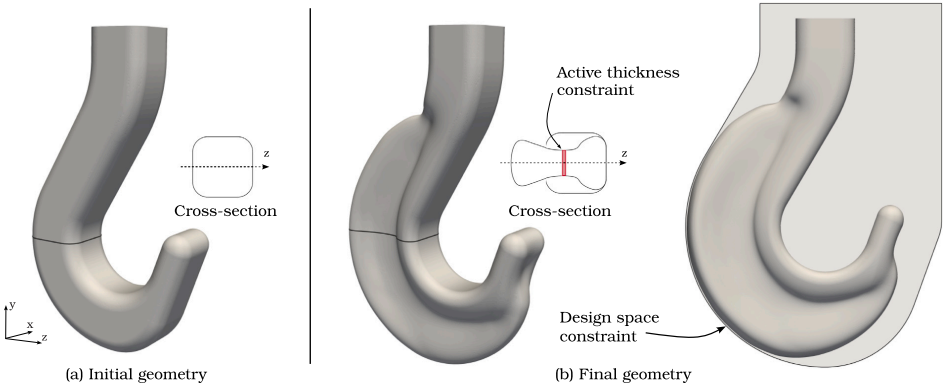


Fig. 13. Hook example: Comparison of initial and final geometry.

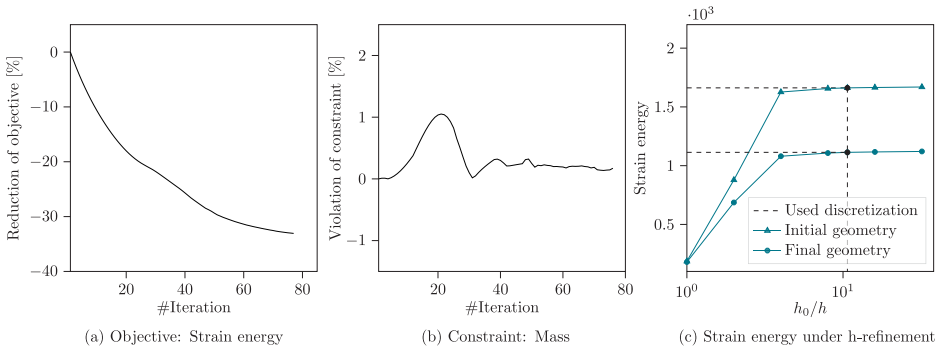


Fig. 14. Hook example: Relevant graphs for the optimization problem, including the objective function (a), the constraint (b), and a convergence study of the background mesh for the initial and final geometry (c).

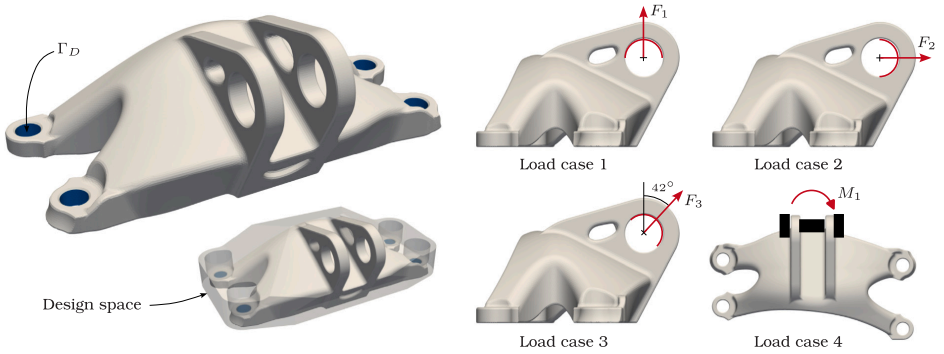


Fig. 15. Jet engine bracket example: Initial geometry (designer: Thomas Johansson; <https://grabcad.com/library/tj2-1>) and problem setup.

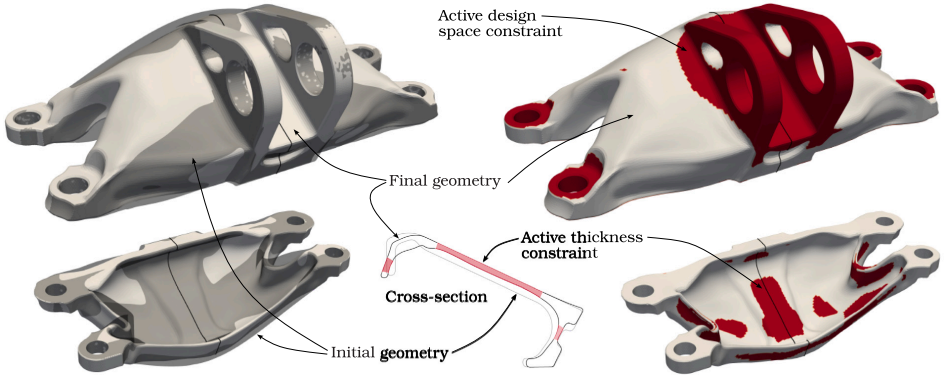


Fig. 16. Jet engine bracket example: Comparison between initial and final geometry. The red areas indicate the active geometrical constraints. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

shown in Fig. 15 is used as the initial geometry of a mass minimization problem. According to the challenge regulations, the structure is subjected to four different load cases. These are defined by $F_1 = 36$ kN, $F_2 = 38.25$ kN, $F_3 = 42.75$ kN, and $M = 546$ N m. For each of them, an individual constraint is formulated to maintain the strain energy value of the initial geometry. In analogy to Eq. (3), the constrained minimization problem of the mass m reads

$$\begin{aligned}
 &\min && m, \\
 &\text{subject to} && \rho = 0, && \text{on } \Gamma, \\
 &&& r = 0, && \text{in } \Omega, \\
 &&& U_j - U_j^{\text{init}} = 0, && j \in \{1, 2, 3, 4\},
 \end{aligned} \tag{46}$$

where U_j and U_j^{init} are the current and initial strain energy values associated with the four load cases. We consider the official design space constraint, as depicted in Fig. 15. In addition, a minimum thickness constraint of 1.0 mm is enforced with $\delta = 0.01$ mm, see Section 5. The surface filter radius is set to $r_f^H = 5$ mm. Again, quadratic C^1 continuous B-Splines span the background mesh using a uniform knot span size of $h = 1.5$ mm. The Young's modulus $E = 1.138 \times 10^5$ N/mm², Poisson's ratio $\nu = 0.342$, and density 4.43×10^{-6} kg/mm³ define the linear elastic material.

Fig. 16 compares the initial and optimized geometry. The red areas highlight the vertices with active design space and material thickness constraints. Table 1 lists the key properties of the final geometry. Although the original shape is already a highly optimized design, a further reduction in mass of $\approx 15\%$ is achieved. Fig. 17(a)–(b) depict the evolution of the objective and the constraints. Analogous to the previous example in Section 7.2, the objective decreases monotonically until the local minimum is reached. The

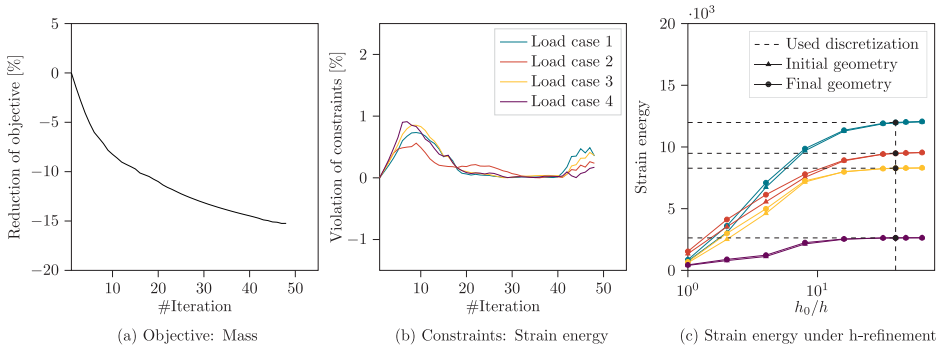


Fig. 17. Jet engine bracket example: Relevant graphs for the optimization problem, including the objective function (a), the constraints (b), and a convergence study of the background mesh for the initial and final geometry (c). The color scheme for the different load cases used in (b) also applies to (c). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Jet engine bracket example: key figures of final geometry.

Objective	Constraints					Min. material thickness	Max. nodal design space violation
Mass	Strain energy load case 1	Strain energy load case 2	Strain energy load case 3	Strain energy load case 4			
-14.96%	+0.35%	+0.23%	+0.36%	+0.17%	1.009 mm	1.048 mm	

design does not exceed a violation of the four strain energy constraints by more than 1% in any optimization phase. Fig. 17(c) again illustrates the quality of the background discretization used in this optimization process. A convergence study of the strain energy under h -refinement is conducted. The graphs show that for all four load conditions, the discretization employed delivers highly accurate primal solutions for both the initial and the final geometry. These results prove that the underlying shape updates do not affect the discretization quality.

8. Conclusion

This publication presents a complete workflow to optimize the shape of embedded solid structures described in boundary representation using implicit Vertex-Morphing. Its main feature is the decoupling of the geometry description from the finite element discretization, eliminating mesh distortion problems even for extensive shape updates. Following the concept of classical embedded boundary methods allows the computation of the primal solution and response functions on a fixed background mesh, i.e., a regular grid, which can be reset in each iteration. A gradient-based constrained optimizer calculates the shape updates of the immersed boundary, representing the structure's skin. The geometric model is defined by an oriented triangle mesh, e.g., in STL format, which offers a seamless interface to additive manufacturing processes. One of the crucial elements for a robust embedding workflow is the reliable construction of integration points for cut domains. To this end, the present work incorporates the open-source framework QuESo [32–34], which provides efficient and highly accurate quadrature rules for arbitrarily shaped geometries. This paves the way for the optimization of solid structures with industrial complexity using embedded boundary methods. The second contribution of the publication at hand is a novel sensitivity analysis delivering shape gradients that are independent of the surface mesh and the background discretization. This procedure includes three main steps.

- Calculation of shape sensitivities at the nodes of the background mesh.
- Weighting of gradients to eliminate size effects of the cut elements.
- Interpolation of gradients at the immersed boundary through the shape functions of the background elements.

Several examples demonstrate the consistency of the proposed sensitivity analysis. The computed gradients are shown to yield smooth shape updates even without additional regularization. Nevertheless, the presented workflow adopts a Helmholtz/Sobolev-based filter to control the minimum feature size of the final geometry and to account for non-design surfaces. Furthermore, a sensitivity damping scheme is introduced that effectively enforces geometric constraints, such as minimum wall thicknesses or design space limits. We demonstrate the potential of our approach by optimizing several models, including an industrial component. In all examples, the initial mesh quality is proven to be maintained throughout the entire simulation, allowing a successful termination at the local minimum. Future work may enrich the current workflow by enhanced gradient-based optimization algorithms, as proposed in [74].

CRediT authorship contribution statement

Manuel Meßmer: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation. **Reza Najian Asl:** Writing – review & editing, Software, Methodology, Investigation. **Stefan Kollmannsberger:** Writing – review & editing, Conceptualization. **Roland Wüchner:** Writing – review & editing, Conceptualization. **Kai-Uwe Bletzinger:** Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors gratefully acknowledge support from the Deutsche Forschungsgemeinschaft (DFG), Germany as part of the SPP 2187 with the project 423935790 “Der digitale Baukasten – Simulationsbasierte Modelle und Methoden für den Entwurf modularer Tragsysteme aus Beton” as well as the support of the DFG, Germany through the project 414265976 TRR 277 C-01 and C-02.

References

- [1] N. Kikuchi, K.Y. Chung, T. Torigaki, J.E. Taylor, Adaptive finite element methods for shape optimization of linearly elastic structures, *Comput. Methods Appl. Mech. Engrg.* 57 (1) (1986) 67–89, [http://dx.doi.org/10.1016/0045-7825\(86\)90071-X](http://dx.doi.org/10.1016/0045-7825(86)90071-X).
- [2] T.-M. Yao, K.K. Choi, 3-D shape optimal design and automatic finite element regriding, *Internat. J. Numer. Methods Engrg.* 28 (2) (1989) 369–384, <http://dx.doi.org/10.1002/nme.1620280209>.
- [3] P. Morin, R.H. Nochetto, M.S. Pauletti, M. Verani, Adaptive finite element method for shape optimization, *ESAIM: Control Optim. Calc. Var.* 18 (4) (2012) 1122–1149, <http://dx.doi.org/10.1051/cocv/2011192>.
- [4] S. Riehl, P. Steinmann, An integrated approach to shape optimization and mesh adaptivity based on material residual forces, *Comput. Methods Appl. Mech. Engrg.* 278 (2014) 640–663, <http://dx.doi.org/10.1016/j.cma.2014.06.010>.
- [5] K. Stein, T. Tezduyar, R. Benney, Mesh moving techniques for fluid-structure interactions with large displacements, *J. Appl. Mech.* 70 (1) (2003) 58–63, <http://dx.doi.org/10.1115/1.1530635>.
- [6] P. Toton, R.A.K. Sanches, K. Takizawa, T.E. Tezduyar, A linear-elasticity-based mesh moving method with no cycle-to-cycle accumulated distortion, *Comput. Mech.* 67 (2021) 413–434, <http://dx.doi.org/10.1007/s00466-020-01941-y>.
- [7] H. Azeгами, K. Takeuchi, A smoothing method for shape optimization: traction method using the robin condition, *Int. J. Comput. Methods* 3 (1) (2006) 21–33, <http://dx.doi.org/10.1142/S0219876206000709>.
- [8] K.E. Swartz, K. Mittal, M. Schmidt, J.-L. Barrera, S. Watts, D.A. Tortorelli, Yet another parameter-free shape optimization method, *Struct. Multidiscip. Optim.* 66 (2023) 245, <http://dx.doi.org/10.1007/s00158-023-03684-9>.
- [9] T. Belytschko, R. Gracie, G. Ventura, A review of extended/generalized finite element methods for material modeling, *Modelling Simul. Mater. Sci. Eng.* 17 (4) (2009) 043001, <http://dx.doi.org/10.1088/0965-0393/17/4/043001>.
- [10] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method, *Appl. Numer. Math.* 62 (4) (2012) 328–341, <http://dx.doi.org/10.1016/j.apnum.2011.01.008>.
- [11] S. Badia, F. Verdugo, A.F. Martín, The aggregated unfitted finite element method for elliptic problems, *Comput. Methods Appl. Mech. Engrg.* 336 (2018) 533–553, <http://dx.doi.org/10.1016/j.cma.2018.03.022>.
- [12] J. Parvizian, A. Düster, E. Rank, Finite cell method, *Comput. Mech.* 41 (2007) 121–133, <http://dx.doi.org/10.1007/s00466-007-0173-y>.
- [13] J. Sethian, A. Wiegmann, Structural boundary design via level set and immersed interface methods, *J. Comput. Phys.* 163 (2) (2000) 489–528, <http://dx.doi.org/10.1006/jcph.2000.6581>.
- [14] X. Wang, M.Y. Wang, D. Guo, Structural shape and topology optimization in a level-set-based framework of region representation, *Struct. Multidiscip. Optim.* 27 (2004) 1–19, <http://dx.doi.org/10.1007/s00158-003-0363-y>.
- [15] G. Allaire, F. Jouve, A.-M. Toader, Structural optimization using sensitivity analysis and a level-set method, *J. Comput. Phys.* 194 (1) (2004) 363–393, <http://dx.doi.org/10.1016/j.jcp.2003.09.032>.
- [16] P. Duysinx, L. Van Miegroet, T. Jacobs, C. Fleury, Generalized shape optimization using X-FEM and level set methods, in: *IUTAM Symposium on Topological Design Optimization of Structures, Machines and Materials*, Springer, 2006, pp. 23–32, <http://dx.doi.org/10.1007/1-4020-4752-5.3>.
- [17] S.-H. Ha, S. Cho, Level set based topological shape optimization of geometrically nonlinear structures using unstructured mesh, *Comput. Struct.* 86 (13) (2008) 1447–1455, <http://dx.doi.org/10.1016/j.comptruc.2007.05.025>.
- [18] P. Wei, M.Y. Wang, X. Xing, A study on X-FEM in continuum structural optimization using a level set model, *Comput. Aided Des.* 42 (8) (2010) 708–719, <http://dx.doi.org/10.1016/j.cad.2009.12.001>.
- [19] J. Norato, R. Haber, D. Tortorelli, M. Bendsoe, A geometry projection method for shape optimization, *Internat. J. Numer. Methods Engrg.* 60 (14) (2004) 2289–2312, <http://dx.doi.org/10.1002/nme.1044>.
- [20] N.H. Kim, Y. Chang, Eulerian shape design sensitivity analysis and optimization with a fixed grid, *Comput. Methods Appl. Mech. Engrg.* 194 (30) (2005) 3291–3314, <http://dx.doi.org/10.1016/j.cma.2004.12.019>.
- [21] A.R. Najafi, M. Safdari, D.A. Tortorelli, P.H. Geubelle, A gradient-based shape optimization scheme using an interface-enriched generalized FEM, *Comput. Methods Appl. Mech. Engrg.* 296 (2015) 1–17, <http://dx.doi.org/10.1016/j.cma.2015.07.024>.
- [22] E. Nadal, J.J. Ródenas, J. Albelda, M. Tur, J.E. Tarancón, F.J. Fuenmayor, Efficient finite element methodology based on cartesian grids: Application to structural shape optimization, *Abstr. Appl. Anal.* 2013 (953786) (2013) <http://dx.doi.org/10.1155/2013/953786>.
- [23] O. Marco, J.J. Ródenas, F.J. Fuenmayor, M. Tur, An extension of shape sensitivity analysis to an immersed boundary method based on cartesian grids, *Comput. Mech.* 62 (2018) 701–723, <http://dx.doi.org/10.1007/s00466-017-1522-0>.

- [24] O. Marco, J.J. Ródenas, J. Albelda, E. Nadal, M. Tur, Structural shape optimization using cartesian grids and automatic h-adaptive mesh projection, *Struct. Multidiscip. Optim.* 58 (2018) 61–81, <http://dx.doi.org/10.1007/s00158-017-1875-1>.
- [25] S. Riehl, P. Steinmann, On structural shape optimization using an embedding domain discretization technique, *Internat. J. Numer. Methods Engrg.* 109 (9) (2017) 1315–1343, <http://dx.doi.org/10.1002/nme.5326>.
- [26] G. Stankiewicz, C. Dev, P. Steinmann, Coupled topology and shape optimization using an embedding domain discretization method, *Struct. Multidiscip. Optim.* 64 (2021) 2687–2707, <http://dx.doi.org/10.1007/s00158-021-03024-9>.
- [27] D.D. Santis, M.J. Zaher, C. Farhat, Gradient based aerodynamic shape optimization using the FIVER embedded boundary method, in: 54th AIAA Aerospace Sciences Meeting, 2016, <http://dx.doi.org/10.2514/6.2016-0807>.
- [28] J.B. Ho, C. Farhat, Aerodynamic shape optimization using an embedded boundary method with smoothness guarantees, in: AIAA Scitech 2021 Forum, 2021, <http://dx.doi.org/10.2514/6.2021-0280>.
- [29] M.P. Bendsoe, O. Sigmund, Material interpolation schemes in topology optimization, *Arch. Appl. Mech.* 69 (1999) 635–654, <http://dx.doi.org/10.1007/s004190050248>.
- [30] P.D. Dunning, H.A. Kim, G. Mullineux, Investigation and improvement of sensitivity computation using the area-fraction weighted fixed grid FEM and structural optimization, *Finite Elem. Anal. Des.* 47 (8) (2011) 933–941, <http://dx.doi.org/10.1016/j.finel.2011.03.006>.
- [31] V. Topa, M. Purcar, C. Munteanu, L. Grindei, C. Pacurar, O. Garvasiuc, Shape optimization approach based on the extended finite element method, *COMPEL - Int. J. Comput. Math. Electr. Electron. Eng.* 31 (2) (2012) 477–497, <http://dx.doi.org/10.1108/03321641211200545>.
- [32] M. Meßner, QuEso, <https://github.com/manuelmessner/QuEso>.
- [33] M. Meßner, T. Teschemacher, L.F. Leidinger, R. Wüchner, K.-U. Bletzinger, Efficient CAD-integrated isogeometric analysis of trimmed solids, *Comput. Methods Appl. Mech. Engrg.* 400 (2022) 115584, <http://dx.doi.org/10.1016/j.cma.2022.115584>.
- [34] M. Meßner, S. Kollmannsberger, R. Wüchner, K.-U. Bletzinger, Robust numerical integration of embedded solids described in boundary representation, *Comput. Methods Appl. Mech. Engrg.* 419 (2024) 116670, <http://dx.doi.org/10.1016/j.cma.2023.116670>.
- [35] S.E. Mousavi, H. Xiao, N. Sukumar, Generalized Gaussian quadrature rules on arbitrary polygons, *Internat. J. Numer. Methods Engrg.* 82 (1) (2010) 99–113, <http://dx.doi.org/10.1002/nme.2759>.
- [36] S.E. Mousavi, N. Sukumar, Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons, *Comput. Mech.* 47 (2011) 535–554, <http://dx.doi.org/10.1007/s00466-010-0562-5>.
- [37] H. Xiao, Z. Gimbutas, A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions, *Comput. Math. Appl.* 59 (2) (2010) 663–676, <http://dx.doi.org/10.1016/j.camwa.2009.10.027>.
- [38] A.P. Nagy, D.J. Benson, On the numerical integration of trimmed isogeometric elements, *Comput. Methods Appl. Mech. Engrg.* 284 (2015) 165–185, <http://dx.doi.org/10.1016/j.cma.2014.08.002>.
- [39] Y. Sudhakar, W.A. Wall, Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods, *Comput. Methods Appl. Mech. Engrg.* 258 (2013) 39–54, <http://dx.doi.org/10.1016/j.cma.2013.01.007>.
- [40] B. Müller, F. Kummer, M. Oberlack, Highly accurate surface and volume integration on implicit domains by means of moment-fitting, *Internat. J. Numer. Methods Engrg.* 96 (8) (2013) 512–528, <http://dx.doi.org/10.1002/nme.4569>.
- [41] L. Radtke, G. Bletsos, N. Kühn, T. Suchan, T. Rung, A. Düster, K. Welker, Parameter-free shape optimization: Various shape updates for engineering applications, *Aerospace* 10 (9) (2023) <http://dx.doi.org/10.3390/aerospace10090751>.
- [42] K.-U. Bletzinger, M. Firl, F. Daoud, Approximation of derivatives in semi-analytical structural optimization, *Comput. Struct.* 86 (13) (2008) 1404–1416, <http://dx.doi.org/10.1016/j.compstruc.2007.04.014>.
- [43] J. Kiendl, R. Schmidt, R. Wüchner, K.-U. Bletzinger, Isogeometric shape optimization of shells using semi-analytical sensitivity analysis and sensitivity weighting, *Comput. Methods Appl. Mech. Engrg.* 274 (2014) 148–167, <http://dx.doi.org/10.1016/j.cma.2014.02.001>.
- [44] K.-U. Bletzinger, R. Wüchner, F. Daoud, N. Camprubi, Computational methods for form finding and optimization of shells and membranes, *Comput. Methods Appl. Mech. Engrg.* 194 (30) (2005) 3438–3452, <http://dx.doi.org/10.1016/j.cma.2004.12.026>.
- [45] K.-U. Bletzinger, M. Firl, J. Linhard, R. Wüchner, Optimal shapes of mechanically motivated surfaces, *Comput. Methods Appl. Mech. Engrg.* 199 (5) (2010) 324–333, <http://dx.doi.org/10.1016/j.cma.2008.09.009>.
- [46] C. Le, T. Bruns, D. Tortorelli, A gradient-based, parameter-free approach to shape optimization, *Comput. Methods Appl. Mech. Engrg.* 200 (9) (2011) 985–996, <http://dx.doi.org/10.1016/j.cma.2010.10.004>.
- [47] M. Firl, K.-U. Bletzinger, Shape optimization of thin walled structures governed by geometrically nonlinear mechanics, *Comput. Methods Appl. Mech. Engrg.* 237–240 (2012) 107–117, <http://dx.doi.org/10.1016/j.cma.2012.05.016>.
- [48] R. Najian Asl, K.-U. Bletzinger, The implicit bulk-surface filtering method for node-based shape optimization and a comparison of explicit and implicit filtering techniques, *Struct. Multidiscip. Optim.* 66 (2023) 111, <http://dx.doi.org/10.1007/s00158-023-03548-2>.
- [49] E. Burman, S. Claus, P. Hansbo, M.G. Larson, A. Massing, CutFEM: Discretizing geometry and partial differential equations, *Internat. J. Numer. Methods Engrg.* 104 (7) (2015) 472–501, <http://dx.doi.org/10.1002/nme.4823>.
- [50] S. Badia, A.F. Martin, F. Verdugo, Mixed aggregated finite element methods for the unfitted discretization of the Stokes problem, *SIAM J. Sci. Comput.* 40 (6) (2018) B1541–B1576, <http://dx.doi.org/10.1137/18M1185624>.
- [51] H.-J. Kim, Y.-D. Seo, S.-K. Youn, Isogeometric analysis for trimmed CAD surfaces, *Comput. Methods Appl. Mech. Engrg.* 198 (37) (2009) 2982–2995, <http://dx.doi.org/10.1016/j.cma.2009.05.004>.
- [52] R. Schmidt, R. Wüchner, K.-U. Bletzinger, Isogeometric analysis of trimmed NURBS geometries, *Comput. Methods Appl. Mech. Engrg.* 241–244 (2012) 93–111, <http://dx.doi.org/10.1016/j.cma.2012.05.021>.
- [53] M. Breitenberger, A. Apostolatos, B. Philipp, R. Wüchner, K.-U. Bletzinger, Analysis in computer aided design: Nonlinear isogeometric B-rep analysis of shell structures, *Comput. Methods Appl. Mech. Engrg.* 284 (2015) 401–457, <http://dx.doi.org/10.1016/j.cma.2014.09.033>.
- [54] R. Saye, Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part I, *J. Comput. Phys.* 344 (2017) 647–682, <http://dx.doi.org/10.1016/j.jcp.2017.04.076>.
- [55] R. Saye, Implicit mesh discontinuous Galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part II, *J. Comput. Phys.* 344 (2017) 683–723, <http://dx.doi.org/10.1016/j.jcp.2017.05.003>.
- [56] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, *Comput. Methods Appl. Mech. Engrg.* 197 (45) (2008) 3768–3782, <http://dx.doi.org/10.1016/j.cma.2008.02.036>.
- [57] D. Schillingner, M. Ruess, N. Zander, Y. Bazilevs, A. Düster, E. Rank, Small and large deformation analysis with the p- and B-spline versions of the finite cell method, *Comput. Mech.* 50 (2012) 445–478, <http://dx.doi.org/10.1007/s00466-012-0684-z>.
- [58] F. de Prenter, C.V. Verhoosel, E.H. van Brummelen, M.G. Larson, S. Badia, Stability and conditioning of immersed finite element methods: analysis and remedies, *Arch. Comput. Methods Engrg.* 30 (2023) 3617–3656, <http://dx.doi.org/10.1007/s11831-023-09913-0>.
- [59] I. Babuska, The finite element method with penalty, *Math. Comp.* 27 (122) (1973) 221–228, <http://dx.doi.org/10.2307/2005611>.
- [60] P.A. Newman, G.J.-W. Hou, A.C. Taylor, Observations Regarding Use of Advanced CFD Analysis, Sensitivity Analysis, and Design Codes in MDO, Institute for Computer Applications in Science and Engineering Report No. 96-16, NASA CR-198293, 1996.

- [61] B.S. Lazarov, O. Sigmund, Filters in topology optimization based on Helmholtz-type differential equations, *Internat. J. Numer. Methods Engrg.* 86 (6) (2011) 765–781, <http://dx.doi.org/10.1002/nme.3072>.
- [62] A. Kawamoto, T. Matsumori, S. Yamasaki, T. Nomura, T. Kondoh, S. Nishiwaki, Heaviside projection based topology optimization by a PDE-filtered scalar function, *Struct. Multidiscip. Optim.* 44 (2011) 19–24, <http://dx.doi.org/10.1007/s00158-010-0562-2>.
- [63] T. Dick, N.R. Gauger, S. Schmidt, Combining Sobolev smoothing with parameterized shape optimization, *Comput. Fluids* 244 (2022) 105568, <http://dx.doi.org/10.1016/j.compfluid.2022.105568>.
- [64] G. Dziuk, C.M. Elliott, Finite element methods for surface PDEs, *Acta Numer.* 22 (2013) 289–396, <http://dx.doi.org/10.1017/S0962492913000056>.
- [65] D. Edelmann, Isoparametric finite element analysis of a generalized robin boundary value problem on curved domains, *SMAI J. Comput. Math.* 7 (2021) 57–73, <http://dx.doi.org/10.5802/smai-jcm.71>.
- [66] J.B. Rosen, The gradient projection method for nonlinear programming. Part I. linear constraints, *J. Soc. Ind. Appl. Math.* 8 (1) (1960) 181–217.
- [67] J.B. Rosen, The gradient projection method for nonlinear programming. Part II. nonlinear constraints, *J. Soc. Ind. Appl. Math.* 9 (4) (1961) 514–532.
- [68] R. Najian Asl, S. Shayegan, A. Geiser, M. Hojjat, K.-U. Bletzinger, A consistent formulation for imposing packaging constraints in shape optimization using vertex morphing parametrization, *Struct. Multidiscip. Optim.* 56 (2017) 1507–1519, <http://dx.doi.org/10.1007/s00158-017-1819-9>.
- [69] A. Geiser, I. Antonau, K.-U. Bletzinger, Aggregated formulation of geometric constraints for node-based shape optimization with vertex morphing, in: 14th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control, *Eccomas Proceedia*, 2021, pp. 80–94, <http://dx.doi.org/10.7712/140121.7952.18383>.
- [70] F.P. Preparata, M.I. Shamos, *Computational Geometry: An Introduction*, first ed., Springer, New York, 1985, <http://dx.doi.org/10.1007/978-1-4612-1098-6>.
- [71] P. Dadvand, R. Rossi, E. Oñate, An object-oriented environment for developing finite element codes for multi-disciplinary applications, *Arch. Comput. Methods Eng.* 17 (2010) 253–297, <http://dx.doi.org/10.1007/s11831-010-9045-2>.
- [72] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S. Idelsohn, E. Oñate, Migration of a generic multi-physics framework to HPC environments, *Comput. & Fluids* 80 (2013) 301–309, <http://dx.doi.org/10.1016/j.compfluid.2012.02.004>.
- [73] V.M. Ferrández, P. Bucher, R. Rossi, J. Cotela, J. Carbonell, R. Zorrilla, R. Tosi, et al., *KratosMultiphysics (Version 8.0)*, Zenodo, 2020, <http://dx.doi.org/10.5281/zenodo.3234644>.
- [74] I. Antonau, M. Hojjat, K.-U. Bletzinger, Relaxed gradient projection algorithm for constrained node-based shape optimization, *Struct. Multidiscip. Optim.* 63 (2021) 1633–1651, <http://dx.doi.org/10.1007/s00158-020-02821-y>.

Bibliography

- [1] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, 2009. DOI: <https://doi.org/10.1002/9780470749081>.
- [2] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005. DOI: <https://doi.org/10.1016/j.cma.2004.10.008>.
- [3] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff–Love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49):3902–3914, 2009. DOI: <https://doi.org/10.1016/j.cma.2009.08.013>.
- [4] M. Breitenberger, A. Apostolatos, B. Philipp, R. Wüchner, and K.-U. Bletzinger. Analysis in computer aided design: Nonlinear isogeometric B-Rep analysis of shell structures. *Computer Methods in Applied Mechanics and Engineering*, 284:401–457, 2015. DOI: <https://doi.org/10.1016/j.cma.2014.09.033>.
- [5] A. Apostolatos, R. Schmidt, R. Wüchner, and K.-U. Bletzinger. A Nitsche-type formulation and comparison of the most common domain decomposition methods in isogeometric analysis. *International Journal for Numerical Methods in Engineering*, 97(7):473–504, 2014. DOI: <https://doi.org/10.1002/nme.4568>.
- [6] H.-J. Kim, Y.-D. Seo, and S.-K. Youn. Isogeometric analysis for trimmed CAD surfaces. *Computer Methods in Applied Mechanics and Engineering*, 198(37):2982–2995, 2009. DOI: <https://doi.org/10.1016/j.cma.2009.05.004>.
- [7] T. Teschemacher, A. M. Bauer, R. Aristio, M. Meßmer, R. Wüchner, and K.-U. Bletzinger. Concepts of data collection for the CAD-integrated isogeometric analysis. *Engineering with Computers*, 38:5675–5693, 2022. DOI: <https://doi.org/10.1007/s00366-022-01732-4>.
- [8] G. Xu, B. Mourrain, A. Galligo, and T. Rabczuk. High-quality construction of analysis-suitable trivariate NURBS solids by reparameterization

- methods. *Computational Mechanics*, 54:1303–1313, 2014. DOI: <https://doi.org/10.1007/s00466-014-1060-y>.
- [9] H. Al Akhras, T. Elguedj, A. Gravouil, and M. Rochette. Isogeometric analysis-suitable trivariate NURBS models from standard B-Rep models. *Computer Methods in Applied Mechanics and Engineering*, 307:256–274, 2016. DOI: <https://doi.org/10.1016/j.cma.2016.04.028>.
- [10] E. Burman and P. Hansbo. Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method. *Applied Numerical Mathematics*, 62(4):328–341, 2012. DOI: <https://doi.org/10.1016/j.apnum.2011.01.008>.
- [11] E. Burman, S. Claus, P. Hansbo, M. G. Larson, and A. Massing. CutFEM: discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 104(7):472–501, 2015. DOI: <https://doi.org/10.1002/nme.4823>.
- [12] S. Badia, F. Verdugo, and A. F. Martín. The aggregated unfitted finite element method for elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 336:533–553, 2018. DOI: <https://doi.org/10.1016/j.cma.2018.03.022>.
- [13] S. Badia, A. F. Martín, and F. Verdugo. Mixed Aggregated Finite Element Methods for the Unfitted Discretization of the Stokes Problem. *SIAM Journal on Scientific Computing*, 40(6):B1541–B1576, 2018. DOI: <https://doi.org/10.1137/18M1185624>.
- [14] E. Nadal, J. J. Ródenas, J. Albelda, M. Tur, J. E. Tarancón, and F. J. Fuenmayor. Efficient Finite Element Methodology Based on Cartesian Grids: Application to Structural Shape Optimization. *Abstract and Applied Analysis*, 2013(953786), 2013. DOI: <https://doi.org/10.1155/2013/953786>.
- [15] O. Marco, R. Sevilla, Y. Zhang, J. J. Ródenas, and M. Tur. Exact 3D boundary representation in finite element analysis based on Cartesian grids independent of the geometry. *International Journal for Numerical Methods in Engineering*, 103(6):445–468, 2015. DOI: <https://doi.org/10.1002/nme.4914>.
- [16] J. Parvzian, A. Düster, and E. Rank. Finite cell method. *Computational Mechanics*, 41:121–133, 2007. DOI: <https://doi.org/10.1007/s00466-007-0173-y>.
- [17] A. Düster, J. Parvzian, Z. Yang, and E. Rank. The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 197(45):3768–3782, 2008. DOI: <https://doi.org/10.1016/j.cma.2008.02.036>.
- [18] D. Schillinger, M. Ruess, N. Zander, Y. Bazilevs, A. Düster, and E. Rank. Small and large deformation analysis with the p- and B-spline versions of the Finite Cell Method. *Computational Mechanics*, 50:445–478, 2012. DOI: <https://doi.org/10.1007/s00466-012-0684-z>.
- [19] P. Dadvand, R. Rossi, and E. Oñate. An Object-oriented Environment for Developing Finite Element Codes for Multi-disciplinary Applications.

- Archives of Computational Methods in Engineering*, 17:253–297, 2010. DOI: <https://doi.org/10.1007/s11831-010-9045-2>.
- [20] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S. Idelsohn, and E. Oñate. Migration of a generic multi-physics framework to HPC environments. *Computers & Fluids*, 80:301–309, 2013. DOI: <https://doi.org/10.1016/j.compfluid.2012.02.004>.
- [21] V. M. Ferrándiz, P. Bucher, R. Rossi, J. Cotela, J. Carbonell, R. Zorrilla, R. Tosi, et al. KratosMultiphysics (Version 8.0). *Zenodo*, 2020. DOI: <https://doi.org/10.5281/zenodo.3234644>.
- [22] FreeCAD: <https://www.freecad.org/>.
- [23] I. Babuska. The Finite Element Method with Penalty. *Mathematics of Computation*, 27(122):221–228, 1973. DOI: <https://doi.org/10.2307/2005611>.
- [24] F. de Prenter, C. V. Verhoosel, E. H. van Brummelen, M. G. Larson, and S. Badia. Stability and Conditioning of Immersed Finite Element Methods: Analysis and Remedies. *Archives of Computational Methods in Engineering*, 30:3617–3656, 2023. DOI: <https://doi.org/10.1007/s11831-023-09913-0>.
- [25] J. B. Rosen. The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints. *Journal of the Society for Industrial and Applied Mathematics*, 8(1):181–217, 1960. (Visited on 02/12/2024).
- [26] J. B. Rosen. The Gradient Projection Method for Nonlinear Programming. Part II. Nonlinear Constraints. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):514–532, 1961. (Visited on 02/12/2024).
- [27] M. Meßmer, L. F. Leidinger, S. Hartmann, F. Bauer, F. Duddeck, R. Wüchner, and K.-U. Bletzinger. Isogeometric Analysis on Trimmed Solids: A B-Spline-Based Approach Focusing on Explicit Dynamics. In *Proceedings of 13th European LS-DYNA Conference*, Ulm, Germany, 2021.
- [28] J. O. Hallquist: LS-DYNA Theory Manual, <https://www.dynasupport.com/manuals/additional/ls-dyna-theory-manual-2005-beta> (2006).
- [29] S. E. Mousavi, H. Xiao, and N. Sukumar. Generalized Gaussian quadrature rules on arbitrary polygons. *International Journal for Numerical Methods in Engineering*, 82(1):99–113, 2010. DOI: <https://doi.org/10.1002/nme.2759>.
- [30] S. E. Mousavi and N. Sukumar. Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons. *Computational Mechanics*, 47:535–554, 2011. DOI: <https://doi.org/10.1007/s00466-010-0562-5>.
- [31] H. Xiao and Z. Gimbutas. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Computers & Mathematics with Applications*, 59(2):663–676, 2010. DOI: <https://doi.org/10.1016/j.camwa.2009.10.027>.

- [32] A. P. Nagy and D. J. Benson. On the numerical integration of trimmed isogeometric elements. *Computer Methods in Applied Mechanics and Engineering*, 284:165–185, 2015. DOI: <https://doi.org/10.1016/j.cma.2014.08.002>.
- [33] G. Legrain. Non-negative moment fitting quadrature rules for fictitious domain methods. *Computers & Mathematics with Applications*, 99:270–291, 2021. DOI: <https://doi.org/10.1016/j.camwa.2021.07.019>.
- [34] W. Garhuom and A. Düster. Non-negative moment fitting quadrature for cut finite elements and cells undergoing large deformations. *Computational Mechanics*, 70:1059–1081, 2022. DOI: <https://doi.org/10.1007/s00466-022-02203-9>.
- [35] T. J. R. 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. DOI: <https://doi.org/10.1016/j.cma.2008.12.004>.
- [36] R. R. Hiemstra, F. Calabrò, D. Schillinger, and T. J. R. 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. DOI: <https://doi.org/10.1016/j.cma.2016.10.049>.
- [37] L. F. Leidinger. *Explicit isogeometric B-Rep analysis for nonlinear dynamic crash simulations*. Dissertation, Technische Universität München, München, 2020. URL: <https://mediatum.ub.tum.de/1542623>.
- [38] Y. Sudhakar and W. A. Wall. Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods. *Computer Methods in Applied Mechanics and Engineering*, 258:39–54, 2013. DOI: <https://doi.org/10.1016/j.cma.2013.01.007>.
- [39] B. Müller, F. Kummer, and M. Oberlack. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *International Journal for Numerical Methods in Engineering*, 96(8):512–528, 2013. DOI: <https://doi.org/10.1002/nme.4569>.
- [40] M. Joulaiian, S. Hubrich, and A. Düster. Numerical integration of discontinuities on arbitrary domains based on moment fitting. *Computational Mechanics*, 57:979–999, 2016. DOI: <https://doi.org/10.1007/s00466-016-1273-3>.
- [41] CGAL: <https://www.cgal.org/>.
- [42] B. Wassermann, S. Kollmannsberger, S. Yin, L. Kudela, and E. Rank. Integrating CAD and numerical analysis: ‘Dirty geometry’ handling using the Finite Cell Method. *Computer Methods in Applied Mechanics and Engineering*, 351:808–835, 2019. DOI: <https://doi.org/10.1016/j.cma.2019.04.017>.
- [43] Q. Zhou and A. Jacobson. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797*, 2016. DOI: <https://doi.org/10.48550/arXiv.1605.04797>.

- [44] K.-U. Bletzinger, M. Firl, and F. Daoud. Approximation of derivatives in semi-analytical structural optimization. *Computers & Structures*, 86(13):1404–1416, 2008. DOI: <https://doi.org/10.1016/j.compstruc.2007.04.014>.
- [45] J. Kiendl, R. Schmidt, R. Wüchner, and K.-U. Bletzinger. Isogeometric shape optimization of shells using semi-analytical sensitivity analysis and sensitivity weighting. *Computer Methods in Applied Mechanics and Engineering*, 274:148–167, 2014. DOI: <https://doi.org/10.1016/j.cma.2014.02.001>.
- [46] K.-U. Bletzinger, M. Firl, J. Linhard, and R. Wüchner. Optimal shapes of mechanically motivated surfaces. *Computer Methods in Applied Mechanics and Engineering*, 199(5):324–333, 2010. DOI: <https://doi.org/10.1016/j.cma.2008.09.009>.
- [47] M. Firl and K.-U. Bletzinger. Shape optimization of thin walled structures governed by geometrically nonlinear mechanics. *Computer Methods in Applied Mechanics and Engineering*, 237-240:107–117, 2012. DOI: <https://doi.org/10.1016/j.cma.2012.05.016>.
- [48] S. Riehl and P. Steinmann. On structural shape optimization using an embedding domain discretization technique. *International Journal for Numerical Methods in Engineering*, 109(9):1315–1343, 2017. DOI: <https://doi.org/10.1002/nme.5326>.
- [49] R. Najian Asl and K.-U. Bletzinger. The implicit bulk-surface filtering method for node-based shape optimization and a comparison of explicit and implicit filtering techniques. *Structural and Multidisciplinary Optimization*, 66:111, 2023. DOI: <https://doi.org/10.1007/s00158-023-03548-2>.
- [50] J. A. Cottrell, A. Reali, Y. Bazilevs, and T. J. R. Hughes. Isogeometric analysis of structural vibrations. *Computer Methods in Applied Mechanics and Engineering*, 195(41):5257–5296, 2006. DOI: <https://doi.org/10.1016/j.cma.2005.09.027>.
- [51] T. J. R. Hughes. *The Finite Element Method : Linear Static and Dynamic Finite Element Analysis*. Dover Publications, 2000.
- [52] F. Auricchio, L. Beirão da Veiga, T. J. R. Hughes, A. Reali, and G. Sangalli. Isogeometric collocation for elastostatics and explicit dynamics. *Computer Methods in Applied Mechanics and Engineering*, 249-252:2–14, 2012. DOI: <https://doi.org/10.1016/j.cma.2012.03.026>.
- [53] E. Rank. Immersed analysis: principles, achievements, challenges, Virtual International Conference on Isogeometric Analysis, 2021.
- [54] Z. Lian, L. Leidinger, S. Hartmann, F. Bauer, and R. Wüchner. Towards the solution of cross-talk in explicit isogeometric B-Rep analysis. In *Proceedings of the 14th European LS-DYNA Conference, Baden-Baden, Germany, October 2023*.
- [55] Z. Lian, L. F. Leidinger, S. Hartmann, F. Bauer, M. Pabst, C. Krisadawat, and R. Wüchner. Cross-Talk Effects in Trimmed Isogeometric Shells and the

- Control Point Duplication Approach. *Submitted to: Computer Methods in Applied Mechanics and Engineering*, 2024.
- [56] L. Coradello, D. D'Angella, M. Carraturo, J. Kiendl, S. Kollmannsberger, E. Rank, and A. Reali. Hierarchically refined isogeometric analysis of trimmed shells. *Computational Mechanics*, 66:431–447, 2020. DOI: <https://doi.org/10.1007/s00466-020-01858-6>.
- [57] D. D'Angella. *The multi-level Bézier extraction for hierarchical local refinement of trimmed isogeometric finite cell analyses*. PhD thesis, Technische Universität München, 2022. URL: <https://mediatum.ub.tum.de/1616380>.

Bisherige Titel der Schriftenreihe

Band	Titel
1	Frank Koschnick, <i>Geometrische Lockingeffekte bei Finiten Elementen und ein allgemeines Konzept zu ihrer Vermeidung</i> , 2004.
2	Natalia Camprubi, <i>Design and Analysis in Shape Optimization of Shells</i> , 2004.
3	Bernhard Thomee, <i>Physikalisch nichtlineare Berechnung von Stahlfaserbetonkonstruktionen</i> , 2005.
4	Fernaß Daoud, <i>Formoptimierung von Freiformschalen - Mathematische Algorithmen und Filtertechniken</i> , 2005.
5	Manfred Bischoff, <i>Models and Finite Elements for Thin-walled Structures</i> , 2005.
6	Alexander Hörmann, <i>Ermittlung optimierter Stabwerkmodelle auf Basis des Kraftflusses als Anwendung plattformunabhängiger Prozesskopplung</i> , 2006.
7	Roland Wüchener, <i>Mechanik und Numerik der Formfindung und Fluid-Struktur-Interaktion von Membrantragwerken</i> , 2006.
8	Florian Jurecka, <i>Robust Design Optimization Based on Metamodeling Techniques</i> , 2007.
9	Johannes Linhard, <i>Numerisch-mechanische Betrachtung des Entwurfsprozesses von Membrantragwerken</i> , 2009.
10	Alexander Kupzok, <i>Modeling the Interaction of Wind and Membrane Structures by Numerical Simulation</i> , 2009.
11	Bin Yang, <i>Modified Particle Swarm Optimizers and their Application to Robust Design and Structural Optimization</i> , 2009.
12	Michael Fleischer, <i>Absicherung der virtuellen Prozesskette für Folgeoperationen in der Umformtechnik</i> , 2009.
13	Amphon Jrusjrunkiat, <i>Nonlinear Analysis of Pneumatic Membranes - From Subgrid to Interface</i> , 2009.
14	Alexander Michalski, <i>Simulation leichter Flächentragwerke in einer numerisch generierten atmosphärischen Grenzschicht</i> , 2010.
15	Matthias Firl, <i>Optimal Shape Design of Shell Structures</i> , 2010.
16	Thomas Gallinger, <i>Effiziente Algorithmen zur partitionierten Lösung stark gekoppelter Probleme der Fluid-Struktur-Wechselwirkung</i> , 2011.
17	Josef Kiendl, <i>Isogeometric Analysis and Shape Optimal Design of Shell Structures</i> , 2011.
18	Joseph Jordan, <i>Effiziente Simulation großer Mauerwerksstrukturen mit diskreten Rissmodellen</i> , 2011.

- | Band | Titel |
|-------------|--|
| 19 | Albrecht von Boetticher, <i>Flexible Hangmurenbarrieren: Eine numerische Modellierung des Tragwerks, der Hangmure und der Fluid-Struktur-Interaktion</i> , 2012. |
| 20 | Robert Schmidt, <i>Trimming, Mapping, and Optimization in Isogeometric Analysis of Shell Structures</i> , 2013. |
| 21 | Michael Fischer, <i>Finite Element Based Simulation, Design and Control of Piezoelectric and Lightweight Smart Structures</i> , 2013. |
| 22 | Falko Hartmut Dieringer, <i>Numerical Methods for the Design and Analysis for Tensile Structures</i> , 2014. |
| 23 | Rupert Fisch, <i>Code Verification of Partitioned FSI Environments for Lightweight Structures</i> , 2014. |
| 24 | Stefan Sicklinger, <i>Stabilized Co-Simulation of Coupled Problems Including Fields and Signals</i> , 2014. |
| 25 | Madjid Hojjat, <i>Node-based parametrization for shape optimal design</i> , 2015. |
| 26 | Ute Israel, <i>Optimierung in der Fluid-Struktur-Interaktion - Sensitivitätsanalyse für die Formoptimierung auf Grundlage des partitionierten Verfahrens</i> , 2015. |
| 27 | Electra Stavropoulou, <i>Sensitivity analysis and regularization for shape optimization of coupled problems</i> , 2015. |
| 28 | Daniel Markus, <i>Numerical and Experimental Modeling for Shape Optimization of Offshore Structures</i> , 2015. |
| 29 | Pablo Suárez, <i>Design Process for the Shape Optimization of Pressurized Bulkheads as Components of Aircraft Structures</i> , 2015. |
| 30 | Armin Widhammer, <i>Variation of Reference Strategy - Generation of Optimized Cutting Patterns for Textile Fabrics</i> , 2015. |
| 31 | Helmut Masching, <i>Parameter Free Optimization of Shape Adaptive Shell Structures</i> , 2016. |
| 32 | Hao Zhang, <i>A General Approach for Solving Inverse Problems in Geophysical Systems by Applying Finite Element Method and Metamodel Techniques</i> , 2016. |
| 33 | Tianyang Wang, <i>Development of Co-Simulation Environment and Mapping Algorithms</i> , 2016. |
| 34 | Michael Breitenberger, <i>CAD-integrated Design and Analysis of Shell Structures</i> , 2016. |
| 35 | Önay Can, <i>Functional Adaptation with Hyperkinematics using Natural Element Method: Application for Articular Cartilage</i> , 2016. |
| 36 | Benedikt Philipp, <i>Methodological Treatment of Non-linear Structural Behavior in the Design, Analysis and Verification of Lightweight Structures</i> , 2017. |

Band	Titel
37	Michael Andre, <i>Aeroelastic Modeling and Simulation for the Assessment of Wind Effects on a Parabolic Trough Solar Collector</i> , 2018.
38	Andreas Apostolatos, <i>Isogeometric Analysis of Thin-Walled Structures on Multipatch Surfaces in Fluid-Structure Interaction</i> , 2018.
39	Altuğ Emiroğlu, <i>Multiphysics Simulation and CAD-Integrated Shape Optimization in Fluid-Structure Interaction</i> , 2019.
40	Mehran Saeedi, <i>Multi-Fidelity Aeroelastic Analysis of Flexible Membrane Wind Turbine Blades</i> , 2017.
41	Reza Najian Asl, <i>Shape optimization and sensitivity analysis of fluids, structures, and their interaction using Vertex Morphing Parametrization</i> , 2019.
42	Ahmed Abodonya, <i>Verification Methodology for Computational Wind Engineering Prediction of Wind Loads on Structures</i> , 2020.
43	Anna Maria Bauer, <i>CAD-integrated Isogeometric Analysis and Design of Lightweight Structures</i> , 2020.
44	Andreas Winterstein, <i>Modeling and Simulation of Wind-Structure Interaction of Slender Civil Engineering Structures Including Vibration Mitigation Systems</i> , 2020.
45	Franz-Josef Ertl, <i>Vertex Morphing for Constrained Shape Optimization of Three-dimensional Solid Structures</i> , 2020.
46	Daniel Baumgärtner, <i>On the Grid-based Shape Optimization of Structures with Internal Flow and the Feedback of Shape Changes into a CAD Model</i> , 2020.
47	Mohamed Khalil, <i>Combining Physics-based models and machine learning for an Enhanced Structural Health Monitoring</i> , 2021.
48	Long Chen, <i>Gradient Descent Akin Method</i> , 2021.
49	Aditya Ghantasala, <i>Coupling Procedures for Fluid-Fluid and Fluid-Structure Interaction Problems Based on Domain Decomposition Methods</i> , 2021.
50	Ann-Kathrin Goldbach, <i>The CAD-Integrated Design Cycle for Structural Membranes</i> , 2021.
51	Iñigo Pablo López Canalejo, <i>A Finite-Element Transonic Potential Flow Solver with an Embedded Wake Approach for Aircraft Conceptual Design</i> , 2022.
52	Mayu Sakuma, <i>An Application of Multi-Fidelity Uncertainty Quantification for Computational Wind Engineering</i> , 2022.
53	Suneth Warnakulasuriya, <i>Development of Methods for Finite Element-Based Sensitivity Analysis and Goal-Directed Mesh Refinement Using the Adjoint Approach for Steady and Transient Flows</i> , 2022.

Band	Titel
54	Klaus Bernd Sautter, <i>Modeling and Simulation of Flexible Protective Structures by Coupling Particle and Finite Element Methods</i> , 2022.
55	Efthymios Papoutsis, <i>On the incorporation of industrial constraints in node-based optimization for car body design</i> , 2023.
56	Thomas Josef Oberbichler, <i>A modular and efficient implementation of isogeometric analysis for the interactive CAD-integrated design of lightweight structures</i> , 2023.
57	Tobias Christoph Teschemacher, <i>CAD-integrated constitutive modeling, analysis, and design of masonry structures</i> , 2023.
58	Shahrokh Shayegan, <i>Enhanced Algorithms for Fluid-Structure Interaction Simulations: Accurate Temporal Discretization and Robust Convergence Acceleration</i> , 2023.
59	Ihar Antonau, <i>Enhanced computational design methods for large industrial node-based shape optimization problems</i> , 2023.
60	Rishith Ellath Meethal, <i>Hybrid modelling and simulation approaches for the solution of forward and inverse problems in engineering by combining finite element methods and neural networks</i> , 2023.
61	Máté Péntek, <i>Method Development for the Numerical Wind Tunnel in Applied Structural Engineering</i> , 2023.
62	Anoop Kodakkal, <i>High Fidelity Modeling and Simulations for Uncertainty Quantification and Risk-averse Optimization of Structures Under Natural Wind Conditions</i> , 2024.
63	Philipp Bucher, <i>CoSimulation and Mapping for large scale engineering applications</i> , 2024.
64	Martin Fußeder, <i>Methodological and Application-Oriented Advances in Sensitivity Analysis with a Focus on Structural Engineering</i> , 2024.
65	Wenjia Wang, <i>Adjoint Sensitivity Analysis for Non-parametric Shape Optimization with Geometric Nonlinearity and Elastoplasticity</i> , 2024.
66	Manuel Meßmer, <i>Efficient and robust quadrature for embedded solids: Application to isogeometric analysis and shape optimization</i> , 2024.