Higher-Order Finite Elements for Embedded Simulation

ANDREAS LONGVA, RWTH Aachen University
FABIAN LÖSCHNER, RWTH Aachen University
TASSILO KUGELSTADT, RWTH Aachen University
JOSÉ ANTONIO FERNÁNDEZ-FERNÁNDEZ, RWTH Aachen University
JAN BENDER, RWTH Aachen University

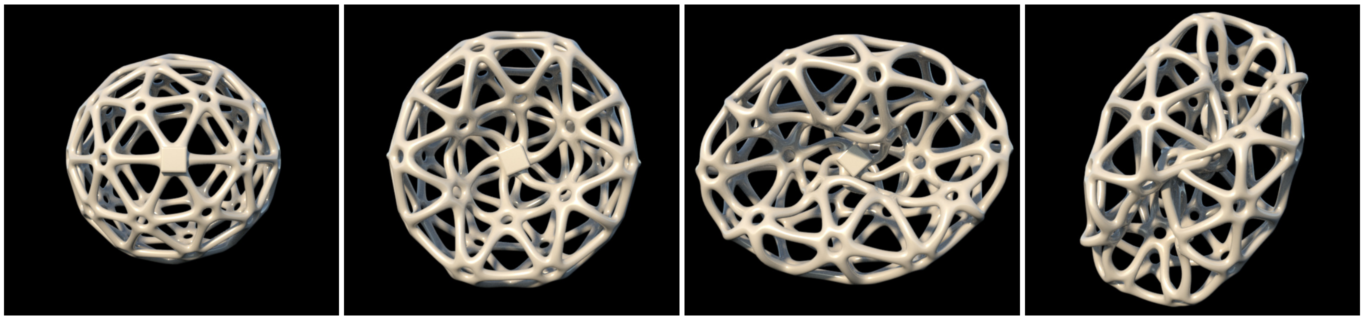


Fig. 1. Sequence of four simulation steps of a deformable hollow ball with holes which is compressed and twisted. Our embedded simulation method with higher-order finite elements is able to capture the complex deformation behavior of the model without the requirement of a high-resolution boundary-conforming discretization. Note that self-collisions are not handled in this simulation.

As demands for high-fidelity physics-based animations increase, the need for accurate methods for simulating deformable solids grows. While higher-order finite elements are commonplace in engineering due to their superior approximation properties for many problems, they have gained little traction in the computer graphics community. This may partially be explained by the need for finite element meshes to approximate the highly complex geometry of models used in graphics applications. Due to the additional per-element computational expense of higher-order elements, larger elements are needed, and the error incurred due to the geometry mismatch eradicates the benefits of higher-order discretizations. One solution to this problem is the embedding of the geometry into a coarser finite element mesh. However, to date there is no adequate, practical computational framework that permits the accurate embedding into higher-order elements.

We develop a novel, robust quadrature generation method that generates theoretically guaranteed high-quality sub-cell integration rules of arbitrary polynomial accuracy. The number of quadrature points generated is bounded only by the desired degree of the polynomial, independent of the embedded geometry. Additionally, we build on recent work in the Finite Cell Method (FCM) community so as to tackle the severe ill-conditioning caused by partially filled elements by adapting an Additive-Schwarz-based preconditioner so that it is suitable for use with state-of-the-art non-linear material models from the graphics literature. Together these two contributions constitute a general-purpose framework for embedded simulation with higher-order finite elements.

We finally demonstrate the benefits of our framework in several scenarios, in which second-order hexahedra and tetrahedra clearly outperform their first-order counterparts.

Authors' addresses: Andreas Longva, longva@cs.rwth-aachen.de, RWTH Aachen University; Fabian Löschner, loeschner@cs.rwth-aachen.de, RWTH Aachen University; Tassilo Kugelstadt, kugelstadt@cs.rwth-aachen.de, RWTH Aachen University; José Antonio Fernández-Fernández, fernandez@cs.rwth-aachen.de, RWTH Aachen University; Jan Bender, bender@cs.rwth-aachen.de, RWTH Aachen University.

CCS Concepts: • **Computing methodologies → Physical simulation**.

Additional Key Words and Phrases: deformable solids, finite cell method, higher-order finite elements, embedded simulation

ACM Reference Format:
Andreas Longva, Fabian Löschner, Tassilo Kugelstadt, José Antonio Fernández-Fernández, and Jan Bender. 2020. Higher-Order Finite Elements for Embedded Simulation. *ACM Trans. Graph.* 39, 6, Article 181 (December 2020), 14 pages. https://doi.org/10.1145/3414685.3417853

1 INTRODUCTION

The Finite Element Method (FEM) is a tried and tested solution to many engineering problems, and also has a long tradition in computer graphics, in particular for the dynamic simulation of deformable bodies. Over the past few decades, a vast amount of research on first-order discretizations has pushed the first-order elements — linear tetrahedra and trilinear hexahedra — close to their computational limits. Whereas traditionally the goal of computer graphics researchers has been *plausibility* at modest computational costs, there is a growing trend towards, and demand for, high-fidelity *predictive* simulations that faithfully reproduce real-life behavior. This is particularly important for virtual prototyping applications, as well as for pushing the boundaries of animation and visual effects. In order to meet this phenomenal challenge, it is our opinion that more effective algorithmic strategies that exhibit faster convergence towards high-fidelity solutions are needed.

Under the right conditions, higher-order elements offer significantly higher convergence rates than first-order elements, and by more faithfully capturing variations in the underlying stress field, are less prone to locking effects. However, due to the increased computational expense of the higher-order formulation, it is generally necessary to use larger elements. This in turns leads to a less accurate approximation of the often highly complex geometrical models used in computer graphics applications. As a result, the errors incurred due to the geometrical mismatch essentially negate the benefits of the higher-order discretization.

A solution to this issue is to accurately embed the high-resolution geometry into a lower resolution *background mesh*, so that the correct distribution of mass and stiffness can be accommodated by the discretization. There exists a vast body of research in various finite element communities on methods that accomplish this task in subtly different ways. For this publication, we have been particularly inspired by the Finite Cell Method (FCM) [Parvizian et al. 2007], which in its simplest form is just the finite element method combined with a procedure for sub-cell integration of the *cut cells*, i.e. the cells in the background mesh that intersect the boundary of the embedded geometry. For non-linear problems in particular, such as the simulation of non-linear deformable solids, it is of crucial importance to obtain high-quality, efficient numerical quadrature rules for the elements used. Higher-order elements generally require more accurate integration, putting additional demand on the quality of the numerical quadrature.

Part of the success of the FEM is arguably owed to its standardized integration procedure. Since elements of the same type use the same quadrature rules regardless of their shape, high-quality tabulated rules with strong theoretical properties are always in reach. This is no longer the case for the FCM or other embedded methods, where individual cut elements require custom rules. However, to ensure robust simulations in all cases, including fully automated pipelines, it is crucial that the quadrature rules for individual elements satisfy certain *theoretically guaranteed* properties.

Our main contribution is a novel robust quadrature generation algorithm that guarantees that all quadrature rules produced for cut elements satisfy the following quality criteria:

- *Accuracy.* Constructed quadrature rules are accurate up to the required polynomial degree m.
- *Boundedness.* The number of quadrature points is bounded by a reasonably small number depending only on the polynomial degree m, and is independent of the complexity of the embedded geometry.
- *Authenticity.* Quadrature points are always inside the embedded geometry.
- *Positivity.* Quadrature weights are always positive.
- *Efficiency.* Quadrature rules can be computed efficiently.

Accuracy is necessary to ensure that all finite element quantities (mass matrix, stiffness matrix, etc.) can be computed to sufficient accuracy in order to ensure theoretically predicted convergence rates. For non-linear problems, numerical quadrature is typically repeatedly re-used. Since assembly scales linearly with the number of quadrature points, it is of crucial importance that the number of quadrature points is *bounded independently of the embedded geometry*, so that the computational complexity of assembly depends only on the complexity of the *background mesh*. The requirement on *authenticity*, i.e. that the quadrature points do not lie in a fictitious domain outside the embedded geometry, ensures that we can assign meaningful material parameters to the quadrature points. Moreover, the evaluation of non-linear basis functions sufficiently far outside the embedded domain is tantamount to polynomial extrapolation, which could lead to unpredictable effects on the solution. *Positivity* prevents negative weights from contributing negative eigenvalues to the coefficient matrices. Negative weights lead to non-physical phenomena associated with negative volume for some physical processes. Finally, our method computes the quadrature rules *efficiently*. This facilitates fast iteration times or even on-the-fly adaptations to changing topology. To our knowledge, our quadrature generation algorithm is the only algorithm that theoretically guarantees all the above properties, in graphics or elsewhere in literature.

De Prenter et al. [2019] recently proposed an Additive-Schwarz-type preconditioner for the FCM that effectively resolves the severe ill-conditioning that arises when cut cells with small volume overlaps are present. However, it is not directly applicable to state-of-the-art non-linear material models found in the computer graphics literature. We extend the preconditioner so that it is applicable also in this setting. Together with a stabilization strategy, we demonstrate that we are able to overcome the severe ill-conditioning issues that plague embedded methods.

Previous works in graphics have exclusively considered embedded simulation with first-order elements. The aforementioned contributions constitute a general-purpose framework for higher-order finite element simulation of embedded deformables. We show in Section 6 the computational advantages of our framework by demonstrating that second-order hexahedra and tetrahedra clearly outperform their first-order counterparts in several scenarios.

2 RELATED WORK

The simulation of deformable bodies has a long history in computer graphics and there is a vast amount of related work on this topic. Here we mainly focus on publications which are closely related to our work, like methods for embedding high-resolution solids in a simpler background mesh and higher-order finite element methods. For a general overview we refer the reader to the survey of Nealan et al. [2006] and a good introduction to finite element methods can be found in the Siggraph course of Sifakis and Barbic [2012].

2.1 Embedded simulation in graphics

A simple way of embedding the simulated geometry into a coarser background mesh is to treat all elements of the background mesh as completely filled. The embedded geometry is deformed by interpolating the displacements from the background mesh. This approach has been used by several authors to simulate embedded geometry [Faloutsos et al. 1997; James et al. 2004; Müller et al. 2004; Rivers and James 2007] or to achieve spatially adaptive FEM simulations [Debunne et al. 2001]. In a similar way the virtual node algorithm [Molino et al. 2004; Sifakis et al. 2007] which was originally designed for cutting and fracturing applications can be used to simulate embedded geometry. However, treating the partially filled elements as completely filled means that the mass and stiffness properties of the embedded geometry are not accurately captured, which can lead to visual artifacts.

To handle partially filled elements of the background mesh accurately, Kaufmann et al. [2008] employ a discontinuous Galerkin (DG) FEM. It is problematic that their method relies on analytic integrals of polynomials to assemble the system matrices because this does not extend to non-linear material models. Later, Kaufmann et al. [2009] also apply the DG-FEM to cutting of shells. Patterson et al. [2012] propose a FEM method that embeds detailed geometry into a regular lattice. They reach sub-voxel accuracy by introducing a specialized integration rule for partially filled boundary cells. However, this only works for hexahedra with trilinear shape functions and it is not obvious whether it can be generalized to work with higher-order elements. Their quadrature rule construction cannot guarantee that all quadrature points are inside the embedded geometry which is problematic when higher-order elements are used. In summary, none of these previous works on embedded simulation present an accurate sub-cell integration method that satisfies all of the five quality criteria that we outlined in Section 1. However, these properties are important for achieving efficient and robust embedded simulations with higher-order finite elements.

Some authors propose methods based on multi-scale techniques, in which the basis of a coarse background mesh is adapted to better capture the underlying behavior of an embedded, fine-resolution solid [Budninskiy et al. 2019; Chen et al. 2018, 2019; Kharevych et al. 2009; Nesme et al. 2006]. Nesme et al. [2009] propose an embedded FEM approach that uses a projection between different levels of a hierarchical hexahedral grid that allows for relatively accurate handling of empty spaces and different material parameters within each cell. We remark that such basis adaptation is largely orthogonal to our method: we make no attempt to modify the finite element basis to be better suited for a particular deformable solid, and such techniques could in principle be built on top of our framework.

2.2 Higher-Order FEM in Graphics

In contrast to engineering, where higher-order methods are often preferred because of their superior accuracy and convergence properties, most works in computer graphics use FEM with linear shape functions because of their simplicity and computation speed [Kugelstadt et al. 2018]. However, there are a few works that show that higher-order methods are useful also in graphics. Roth et al. [1998] use quadratic tetrahedral finite elements with Bézier basis functions in surgery simulations. Mezger et al. [2009] employ quadratic tetrahedral elements to simulate elastoplastic materials for the purpose of shape editing. Bargteil and Cohen [2014] present a simulation of deformable bodies in which they can adaptively switch from linear to quadratic elements in regions of large deformations where more accuracy is required. Weber et al. [2013; 2015] present an efficient GPU implementation of a simulation method with quadratic elements and a method with cubic elements and p-multigrid solver. These previous works show that higher-order FEM can produce better simulation quality or save computation time because fewer elements are needed in comparison to linear ones. However, in all of these methods the benefit from using fewer elements is limited by the need for conforming discretizations. In contrast, Rémillard and Kry [2013] use quadratic B-spline elements on a regular lattice to model volumetric deformation of an embedded solid coupled with thin shell surface deformations. However, they do not accurately account for the material distribution in the partially filled elements near the surface.

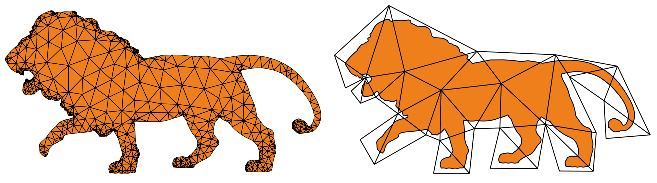

(a) FEM - 1348 elements (b) FCM - 32 cells

Fig. 2. Since the FCM does not need to fit the boundary, one can generate substantially simpler finite element meshes that still capture the main topological features of the embedded geometry. We refer to the finite element mesh used in the FCM as the *background mesh*.

2.3 Finite Cell Method

We discuss relevant connections to the Finite Cell Method [Düster et al. 2008; Parvizian et al. 2007] in Section 4 for topics related to sub-cell integration and Section 5 for topics related to the treatment of ill-conditioning. We recommend the review paper by Schillinger and Ruess [2015] for an overview.

3 EMBEDDED FINITE ELEMENTS FOR DEFORMABLE SOLIDS

First, we briefly review finite element methods for deformable solids, thereby introducing our notation and establishing the mathematical framework we need for the following sections.

3.1 Background

We consider the equation of motion for a deformable solid, posed in its reference configuration:

$$\rho \ddot{\mathbf{u}} = \operatorname{div} \mathbf{P}(\mathbf{F}) + \mathbf{f}^{\text{ext}} \qquad \text{in } \Omega. \tag{1}$$

Here $\Omega \subseteq \mathbb{R}^d$ is the d-dimensional geometry of the solid, $\rho = \rho(\mathbf{X})$ is the density at point $\mathbf{X}$ in the configuration. Similarly, $\mathbf{u} = \mathbf{u}(t, \mathbf{X})$ is the displacement at time t, $\mathbf{F}$ the deformation gradient, $\mathbf{P}$ the first Piola-Kirchhoff stress tensor and $\mathbf{f}^{\text{ext}} = \mathbf{f}^{\text{ext}}(t, \mathbf{X})$ represents the external forces per unit volume. We have here omitted the inclusion of damping forces in order to simplify the presentation, but e.g. Rayleigh damping may be incorporated as in any other FEM framework. The resulting weak form associated with the finite-dimensional Galerkin problem is

$$\begin{aligned}\int_{\Omega_h} \rho\, \ddot{\mathbf{u}}_h \cdot \mathbf{w}_h \, d\mathbf{X} = -\int_{\Omega_h} \mathbf{P}_h : \nabla \mathbf{w}_h \, d\mathbf{X} + \int_{\delta\Omega_h} \boldsymbol{\tau} \cdot \mathbf{w}_h \, dA \\ + \int_{\Omega_h} \mathbf{f}^{\text{ext}} \cdot \mathbf{w}_h \, d\mathbf{X} \qquad \forall \mathbf{w}_h \in V_h^{\text{FEM}},\end{aligned} \tag{2}$$

where $\mathbf{w}_h$ is a test function in the finite-dimensional approximation space V_h^{FEM}, Ω_h is the domain associated with the finite element mesh and $\delta\Omega_h$ its boundary. $\boldsymbol{\tau}$ represents the surface traction. We do not consider Neumann boundary conditions, and we remark that the surface traction terms on the Dirichlet boundary may be ignored when the Dirichlet nodes are constrained directly to the boundary. For traditional finite element methods, it is assumed that $\Omega \approx \Omega_h$, i.e. that the finite element mesh closely approximates the real geometry Ω.

3.2 Embedded Finite Elements

In computer graphics applications, it is usually of interest to simulate high-resolution geometry at comparatively low cost. It is therefore common practice to *embed* high-resolution geometry into a lower-resolution simulation mesh. In the context of finite elements, the simplest form of embedding is realized simply by simulating an associated lower resolution, enclosing finite element mesh, which we refer to as the *background mesh* (Figure 2b). Then the displacement of the high-resolution embedded geometry is interpolated from the coarser background mesh. Although straightforward, this method artificially adds mass and stiffness in regions devoid of material, because the integral is computed over a larger domain $\Omega_h \supseteq \Omega$. We show in Section 6 that a poor approximation of the geometry may lead to a severe decrease in convergence rate of the FEM.

Conceptually, the remedy is simple: Simply replace the integration domain Ω_h in the weak form (2) with the embedded geometry Ω. This means that the typical per-element integrals in the FEM are replaced by integrals over partially filled elements. If these integrals are computed exactly, then the resulting method is mathematically equivalent to the Finite Cell Method (FCM) (and other similar immersed methods). In order to distinguish between the standard FEM and our embedded method, we will henceforth refer to standard discretizations on Ω_h as FEM, and the method that arises when embedding Ω into Ω_h and integrating over Ω as FCM.

Alas, this remedy immediately poses two major computational challenges. First is the realization of an effective integration method for the intersection of embedded geometry and individual finite elements. Second, the embedding almost invariably leads to very small supports of some basis functions, wreaking havoc on the conditioning of the finite element equations. We present solutions to both these problems in the following sections.

4 INTEGRATION OF CUT ELEMENTS

All matrix and vector components associated with the FEM can be decomposed as a sum of integrals over individual elements. These per-element integrals are then computed by tabulated quadrature. This is also the case for the FCM, except that the integrals for elements that intersect the boundary of Ω must take into account only the part of the element that intersects Ω. Thus, the FEM and FCM differ only in the quadrature rules used for boundary elements.

A question that arises is how accurate the quadrature rules need to be. In general, higher-order elements require quadrature rules that are accurate to higher polynomial accuracy. Bathe [2006] recommends to use exact quadratures when possible to ensure reliability. For linear elasticity, the integrands are polynomials and can be exactly evaluated, but for non-linear problems, the best choice may be very problem-dependent. Since our method allows us to construct quadrature rules that are exact up to any desired polynomial order, we leave the choice of accuracy to the user.

We will proceed to make the discussion more mathematically precise. Basis functions are defined on a *reference element* (sometimes called *master element*) and transformed for each element K by an invertible mapping $T_K : \hat{K} \to K$ from the reference element $\hat{K}$ to K. This construction makes it convenient to transform integrals from material space back to integrals over the reference element. For a given quantity $f(\mathbf{X})$, we have

$$\int_{K\cap\Omega} f(\mathbf{X})\, d\mathbf{X} = \int_{T_K^{-1}(K\cap\Omega)} f(T_K(\boldsymbol{\xi}))\, |J_K(\boldsymbol{\xi})| \;\mathrm{d}\,\boldsymbol{\xi} \tag{3}$$

$$\approx \sum_{j=1}^{N} w_j f(T_K(\boldsymbol{\xi}_j))\, |J_K(\boldsymbol{\xi}_j)|, \tag{4}$$

where $w_j, \boldsymbol{\xi}_j$ for $j = 1, \ldots, N$ represent quadrature weights and points in the reference element $\hat{K}$, and J_K denotes the Jacobian of the transformation T_K. If $K = K \cap \Omega$ (as it is assumed for the traditional FEM), then $T_K^{-1}(K \cap \Omega) = \hat{K}$ and we may use standard quadrature rules. Thus, we focus only on the construction of specialized per-element integration rules for elements K that intersect the boundary of the embedded geometry, often referred to as *cut* or *partially filled* elements. Roughly speaking, quadrature construction algorithms rely on one (or both) of the following paradigms:

- *Geometric methods.* The embedded domain is somehow (approximately) decomposed into smaller regions for which quadrature rules are known.
- *Moment-fitting methods.* Quadrature rules are obtained as the solution to an algebraic problem involving the moments of polynomials.

In the following sections, we will briefly describe some common approaches to integration of partially filled cells. Building on these approaches, we describe our proposed quadrature simplification method in Section 4.4. Finally, in Section 4.5 we propose a geometric method that, when combined with our simplification scheme is suitable for quadrature generation for polyhedral embedded geometry. Figure 3 demonstrates our proposed procedure for computing quadrature rules for a given pair of background and embedded meshes.

4.1 Geometric subdivision

Methods based on geometric subdivision are conceptually simple and can produce high-quality quadrature rules, though at the expense of a large number of quadrature points. Given a background cell K and embedded geometry Ω, the cell is left unchanged if it is entirely contained inside the embedded geometry. Otherwise, if the cell intersects the boundary of the embedded geometry, an approximate geometry $\tilde{K}$ is constructed by recursively subdividing the cell K. For each subdivision level, the sub-cells entirely outside the embedded geometry are discarded. Then only cells that intersect Ω are recursively subdivided. The recursive process stops either at a predefined maximum depth or when a sufficiently good approximation of the embedded geometry has been obtained. Once the octree

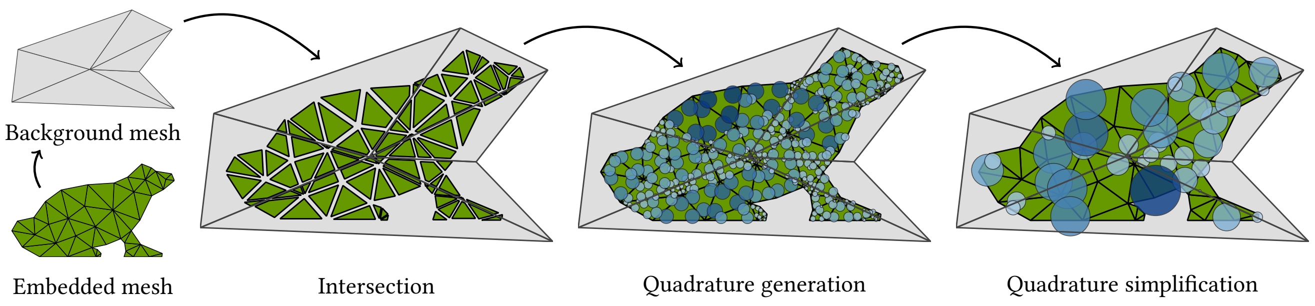


Fig. 3. We embed a mesh of a frog with 55 triangles into a background mesh consisting of 6 triangles and compute quadrature rules that are exact for polynomials of order 2 for each background element. *Intersection (Section 4.5)*: For each background cell, we compute the intersection of the embedded mesh with the background cell, which gives a collection of convex polygons. *Quadrature generation (Section 4.5)*: For each background cell, each polygon is triangulated. For each resulting triangle, we take a standard 3-point triangle quadrature rule (accurate up to order 2 polynomials) and map the weights and points to material space, such that the collection of points from all triangles form an initial quadrature rule for each background element, with a total of 429 points in all background elements. The quadrature is accurate to polynomial order 2. Larger area and darker color corresponds to larger weights. *Quadrature simplification (Section 4.4)*: The quadrature rule in each background elements is simplified by picking at most $M = 6$ weights from the initial quadrature and a new set of positive weights. The resulting quadrature consists of 36 points in total, and is also accurate to polynomial order 2. Note that in practice, the quadratures are stored (and simplified) in coordinates of the reference element, not in world coordinates as depicted here.

has been sufficiently refined, a quadrature rule can be constructed by combining the Gauss quadratures of the individual sub-cells.

Subdivision methods have been extensively researched within the FCM community. Traditionally the FCM has been associated with regularly shaped hexahedral elements (see e.g. [Düster et al. 2008; Parvizian et al. 2007]), for which an octree subdivision scheme is straightforward to implement, but there also exist similar subdivision methods for tetrahedral elements, in the form of TetFCM [Duczek et al. 2016]. We mention *smart octrees* [Kudela et al. 2016] as an important variation of this paradigm, in which cells are subdivided according to locations of sharp features of the embedded geometry. Nevertheless, the number of quadrature points obtained from any purely geometrical method generally increases with the complexity of the embedded geometry, unless accuracy is compromised by limiting the number of quadrature points generated.

4.2 Moment fitting

Moment fitting is a fundamentally different approach to quadrature generation [Müller et al. 2013]. We limit our discussion to moments associated with total-order polynomials. Let $\mathbb{P}_m^d$ denote the space of polynomials in $\mathbb{R}^d$ of total order at most m. Then if f is a polynomial function in $\mathbb{P}_m^d$, it can be expanded in an appropriate basis such that

$$f(\xi) = \sum_{i=1}^{M} \kappa_i p_i(\xi) \tag{5}$$

for basis polynomials p_i and weights κ_i. Even when f is not a polynomial, it is widely recognized that quadrature rules based on polynomial approximation are well-suited for general purpose integration of smooth functions. The simplest choice of basis, the monomial basis, would give basis polynomials of the form $x^\alpha y^\beta z^\gamma$ for $\alpha + \beta + \gamma \le m$. The number of basis polynomials $M = \dim \mathbb{P}_m^d$ is given by (see e.g. [Xu 1997])

$$M = \binom{m+d}{m}. \tag{6}$$

Given N quadrature points $\xi_j \in \mathbb{R}^d$ and weights w_j for integrating over a domain D, the quadrature rule is able to exactly integrate f in D if the rule is able to integrate every basis polynomial exactly. This leads to the set of algebraic equations

$$\begin{pmatrix} w_1 p_1(\xi_1) + w_2 p_1(\xi_2) + \cdots + w_N p_1(\xi_N) \\ \vdots \\ w_1 p_M(\xi_1) + w_2 p_M(\xi_2) + \cdots + w_N p_M(\xi_N) \end{pmatrix} = \begin{pmatrix} \int_D p_1(\xi)\,\mathrm{d}\xi \\ \vdots \\ \int_D p_M(\xi)\,\mathrm{d}\xi \end{pmatrix},$$

which is compactly represented by the $M \times N$ system of non-linear equations, in which $P = P(\xi_1, \ldots, \xi_N) \in \mathbb{R}^{M \times N}$

$$P\mathbf{w} = \mathbf{b}. \tag{7}$$

Computing good quadrature points from the above relation is a major computational challenge. For this reason, most practitioners simplify the problem by a-priori determining a set of suitable points based on application-specific heuristics. With the points fixed, the problem reduces to the solution of a linear system for the weights. However, the conditioning of the matrix P often makes it very difficult to obtain highly accurate solutions, especially for higher-order polynomials. Moreover, the resulting weights are in general not positive. We remark that there exist sophisticated and very computationally expensive methods that may be able to solve the full non-linear problem (see e.g. the work of Keshavarzzadeh et al. [2018] and the references therein), but to our knowledge, the robustness of these procedures is not guaranteed.

Additionally, some geometric procedure is still required for computing the integrals in the vector $\mathbf{b}$. Popular choices are Monte Carlo integration [Patterson et al. 2012] or transforming volume integrals to surface integrals by way of the divergence theorem [Hafner et al. 2019; Koschier et al. 2017]. In the first case, it might prove difficult to obtain sufficient accuracy for polynomials of higher order. In the second case, self-intersections or non-manifold surface geometry might reduce the robustness of the procedure.

4.3 LP-based moment fitting

Ryu and Boyd [2014] demonstrated that Gauss quadratures may be formulated as solutions to an infinite-dimensional Linear Program (LP). From this, they developed a method in which a finite-dimensional LP based on moment-fitting of a fine sampling of points in the domain of interest is solved to obtain an initial quadrature rule, which is further improved through non-linear optimization.

However, as remarked by other authors [Jakeman and Narayan 2018], the solvability of this LP is closely tied to the placement of points, and may fail altogether, which matches our own observations while working with the closely related LP formulation that we will present in the next section. As a result, additional complex safeguards are necessary to ensure convergence, and the theoretical guarantees provided by the original infinite-dimensional LP cannot be maintained in practice.

4.4 Simplification of high-quality rules

We have seen that while geometric methods can produce high-quality rules with positive weights and points (approximately) interior to the embedded geometry, the number of points produced may easily make simulations intractably expensive for complex geometry. Moment-fitting methods use an algebraic procedure to potentially produce a much smaller amount of points, but it may be difficult or expensive to obtain high accuracy, and often there are no guarantees that the weights are positive, which may be of crucial importance for non-linear problems.

We propose an algorithm that takes a high-quality quadrature rule satisfying all of the quality criteria outlined in Section 1 except for *Boundedness* and produces a *simplified* quadrature consisting of at most M quadrature points for a given polynomial degree m, independent of the embedded geometry. This initial quadrature rule could for example come from a geometric subdivision method (Section 4.1) or from our intersection-based method (Section 4.5).

4.4.1 Admissible quadrature rules. For a given polynomial degree m, we seek a quadrature rule that satisfies the moment fitting equations (7), with the additional constraint that the weights must be non-negative. That is, we seek points and weights $\mathbf{w}$ that satisfy

$$P\mathbf{w} = \mathbf{b}, \qquad \mathbf{w} \geq 0. \tag{8}$$

We denote by $\mathbf{w}^0 \in \mathbb{R}^N$ and ξ_j^0 for $j = 1, \dots N$ a set of N *initial* quadrature weights and points that satisfy (8). We assume that $N > M$, otherwise we leave the quadrature rule untouched, as we can not usually expect to improve it further in this case.

In practice, it is not important that the given quadrature weights $\mathbf{w}_0$ satisfy $P\mathbf{w} = \mathbf{b}$ exactly. In fact, we define $\mathbf{b}^0 = P(\xi^0)\mathbf{w}^0$ and instead relax the admissible set of weights to those that satisfy $P^0\mathbf{w} = \mathbf{b}^0$, where $P^0 = P(\xi^0)$. Thus, our simplification method will produce weights that are *as good as the initial quadrature rule.* If the initial quadrature rule is exact, our simplified rule will also be exact (up to numerical error). To simplify the remaining presentation, we assume that $P = P^0$ and $\mathbf{b} = P\mathbf{w}^0$.

4.4.2 Basic feasible points. For a fixed set of quadrature points (and therefore P), the admissible weights are represented exactly by the *feasible set* of a Linear Program (LP), and it is very possible that the set of admissible weights is empty, i.e. that there exist no admissible weights for the given set of points. However, by assumption, $\mathbf{w}^0$ and ξ^0 are admissible, and so we know that the feasible set is non-empty. Without loss of generality, we assume that $P = P^0$ has full row rank. Otherwise, since the linear system is consistent, it is possible to remove linearly dependent rows from P and $\mathbf{b}$ and produce a linear system $P'\mathbf{w} = \mathbf{b}'$ with a smaller number of equations. From Linear Programming theory (see e.g. [Nocedal and Wright 2006]), these prerequisites imply the existence of one or more *basic feasible points* $\mathbf{w}$. In this context, a basic feasible point $\mathbf{w}$ is an admissible set of weights with at most M non-zeros. If w_j is zero, we may discard ξ_j, and so in other words it is always possible to select M points from ξ^0 for which there are admissible weights. In practice, such a basic feasible point $\mathbf{w}$ can be obtained directly by a simplex solver for solving LPs to very high precision at reasonable cost, provided M is not too large (polynomials of $m = 10$ and higher can be accommodated, but costs increase quickly).

4.4.3 Choice of polynomial basis. The simplest choice of polynomial basis is the set of monomials, i.e. $x^\alpha y^\beta z^\gamma$ for $\alpha+\beta+\gamma \leq m$. However, monomials are notorious for producing severely ill-conditioned systems due to their near-linear dependence in large subregions of the domain. In practice, tensor product constructions of Chebyshev or Legendre polynomials are sensible alternatives. We remark that the choice of polynomial basis for the quadrature construction is unrelated to the choice of polynomials for the finite element basis. In our implementation, we use a tensor product basis of one-dimensional α-order Chebyshev polynomials $p_\alpha(x)$ of the first kind, defined as

$$p_{\alpha\beta\gamma}(x, y, z) = p_\alpha(x)\, p_\beta(y)\, p_\gamma(z), \tag{9}$$

with $\alpha+\beta+\gamma \leq m$. These polynomials are defined in such a way that they behave well in the unit cube domain $[-1, 1]^3$. For the moment, we assume without loss of generality that $\xi_j^0 \in [-1, 1]^d$ for all j. In the context of cut cells and embedded geometry, it is quite likely that some cells will contain only a small cluster of points in $[-1, 1]^d$. To be more precise, let us assume that the points are contained inside a subdomain $U \subset [-1, 1]^d$ with diameter $|U| \ll 1$. In this case, the values of the basis polynomials barely change inside of the subdomain, and the basis functions $p_{\alpha\beta\gamma}$ become close to linearly dependent. To improve this situation, we scale and translate the polynomials to the bounding box of our quadrature points, which significantly improves conditioning.

4.4.4 Robust simplification. In practice, the LP (8) suffers from some of the same problems as other moment-fitting methods. The conditioning of the matrix may still be poor, despite adapting the basis polynomials to the point set, which might cause simplex solvers to fail in finding a good solution. However, all is not lost. The key ingredient in our simplification scheme is the realization that we may take advantage of the initial quadrature $\mathbf{w}^0$ to transform the LP into an equivalent, much better conditioned LP.

The key property we exploit is that the right-hand side $\mathbf{b}$ is evaluated by our original quadrature rule, and so we may write $\mathbf{b} = P\mathbf{w}^0$. Thus, we may reformulate the equality constraint $P\mathbf{w} = \mathbf{b}$ as $(\mathbf{w} - \mathbf{w}^0) \in \ker P$, where $\ker P$ denotes the kernel (null space) of

P. Let now $P = U\Sigma V^T$ denote the SVD of P, $r = \operatorname{rank} P$ and V_r the first r columns of V. The kernel constraint may then be formulated $V_r^T(\mathbf{w} - \mathbf{w}^0) = 0$, and we obtain the transformed LP feasible set

$$V_r^T \mathbf{w} = V_r^T \mathbf{w}^0, \qquad \mathbf{w} \geq 0. \tag{10}$$

Since V_r^T has orthogonal rows, this modified but equivalent formulation is perfectly conditioned in terms of its singular values, and may in principle be solved by any robust LP solver. Whereas we experienced frequent, spurious failures with the original LP formulation, we have only found the transformed formulation to fail to reach high accuracy when faced with severely degenerate geometry, and in all cases simply filtering out the near-degenerate geometry would resolve the issue.

We remark that our LP formulation has no objective function to optimize for. One might attempt to further minimize some function that represents the sparsity of $\mathbf{w}$, in the hope that this might give further reduction in the number of points. The sparsity-minimization problem can be formulated exactly as a Mixed Integer Linear Program (MILP), and although we found this to be prohibitively expensive, our tests showed that little was to be gained from trying to find sparser solutions $\mathbf{w}$. This seems to suggest that in general simply picking a *subset of existing points* is not sufficient to obtain a sparser quadrature, and more complex and expensive non-linear optimization techniques (e.g. [Jakeman and Narayan 2018; Keshavarzzadeh et al. 2018]) for more optimal point locations may be suitable if high offline precomputation costs are permitted.

4.4.5 Summary. When combined with a high-quality initial quadrature, we conclude that:

- Our simplification algorithm picks a subset of points from the initial quadrature. Thus, the points are inside the embedded geometry if the initial points are.
- The new weights are positive and preserve the polynomial approximation properties of the initial quadrature, up to any reasonable polynomial order m.
- Our algorithm never produces more than M points, where M is the dimension of the polynomial basis.
- Our algorithm is reliable and is suitable for use in a fully automated pipeline.

For $m \leq 10$, M is only a small constant factor (at most $\approx 3 - 4$) greater than the number of points found in state-of-the-art quadrature rules for interior cells [Witherden and Vincent 2015]. Although we have restricted our discussion to total-order polynomial rules, our algorithm is not restricted to total-order moments, and can easily be adapted to any other type of moments (such as product rule for hex elements) by replacing the polynomial basis. For relatively low order polynomials (say, $m \leq 6$ or so), our experience suggests that the computational costs are modest. In fact, we have not attempted to make any kind of optimizations from our initial implementation, because our precomputation costs were insignificant compared to the simulation. The computation is dominated by the SVD and the simplex solver. The former has a complexity of $O(M^2N)$, and so scales linearly with the complexity of the geometry. The simplex solver technically does not have a polynomial complexity bound, but in practice it works well. We conclude that our proposed algorithm fulfills the desiderata set out in the introduction, provided that the initial quadrature rule can be obtained at acceptable cost.

4.5 Quadrature rules from intersections of convex polyhedra

When the embedded geometry is described by a level-set function, subdivision-based methods work very well for constructing an initial quadrature rule as input to our simplification algorithm. However, for many tasks in computer graphics, the geometry is initially described by a surface mesh that may not even be designed for simulation, and so might be non-manifold, have self-intersections or otherwise exhibit artifacts that would complicate the definition of a consistent level-set function or directly computing the intersection between the surface mesh and a given background mesh element. State-of-the-art mesh generators are able to reliably and quickly produce high-quality tetrahedral meshes even from severely flawed input [Hu et al. 2020, 2018]. By offloading these complex geometry processing tasks to such external specialized software, we instead propose a method that, given a background finite element mesh and a polyhedral mesh consisting of convex polyhedral cells, directly produces high-quality quadrature rules of arbitrarily high polynomial degree m, and therefore serves as a natural companion to our simplification algorithm. Figure 3 gives a high-level overview of the procedure when used together with our simplification algorithm.

Let the embedded geometry Ω be represented by a mesh $\mathcal{T}$ that consists of convex polyhedral cells. For a given finite element K whose geometry is a convex polyhedron, we can easily obtain a tetrahedral decomposition of the intersection $K \cap \Omega$ by intersecting each (candidate) polyhedral cell with K and decompose the result into a set of tetrahedra — a trivial operation given that the polyhedron is convex. We remark that there is no need to preserve the connectivity of the polyhedral mesh. Let $\mathcal{T}_K$ denote such a tetrahedralization for element K, and let $\boldsymbol{\eta}_j$ and l_j denote quadrature points and weights for the reference tetrahedron $\hat{Q}$. Employing the Inverse Function Theorem, we can reformulate the integral (3) as

$$\int_{K\cap\Omega} f(\mathbf{X})\, d\mathbf{X} = \sum_{Q\in\mathcal{T}_K} \int_Q f\, d\mathbf{X} = \sum_{Q\in\mathcal{T}_K} \int_{\hat{Q}} f \circ T_Q\, |J_Q|\, d\mathbf{X}$$
$$\approx \sum_{Q\in\mathcal{T}_K} \sum_j l_j\, f(T_Q(\boldsymbol{\eta}_j))\, |J_Q|,$$

and defining $\mathbf{X}_j^Q = T_Q(\boldsymbol{\eta}_j) \in Q$ as the quadrature points in physical space, we can map them back to $\hat{K}$ as $\boldsymbol{\xi}_j^Q := T_K^{-1}(\mathbf{X}_j^Q)$ to obtain the quadrature rule

$$\int_{K\cap\Omega} f(\mathbf{X})\, d\mathbf{X} \approx \sum_{Q,j} l_j \frac{|J_Q|}{|J_K(\boldsymbol{\xi}_j^Q)|}\, f(T_K(\boldsymbol{\xi}_j^Q))\, |J_K(\boldsymbol{\xi}_j^Q)| \tag{11}$$
$$= \sum_{Q,j} w_j^Q f(T_K(\boldsymbol{\xi}_j^Q))\, |J_K(\boldsymbol{\xi}_j^Q)|. \tag{12}$$

Clearly, the weights w_j^Q are positive for non-degenerate elements. Moreover, if T_K is a linear map (e.g. tetrahedra, parallelogram-shaped hexahedra), f a polynomial of degree m and the tetrahedron quadrature rule $l_j, \boldsymbol{\eta}_j$ integrates polynomials of degree m exactly, then the resulting quadrature rule for $K \cap \Omega$ integrates f exactly. If T_K is not a linear map, then the application of T_K^{-1} is straightforward to compute with Newton's method.

Since it is possible to construct quadrature rules for tetrahedra with arbitrarily high polynomial order, the proposed intersection method can generate quadrature rules of arbitrarily high order. This is particularly useful because the mass matrix integrals for some higher-order elements must be evaluated to relatively high order to avoid singular mass matrices.

Unlike subdivision methods, the proposed intersection method *directly* constructs a quadrature rule of the desired degree, and no stopping criterion is needed for an iterative process. Given a robust implementation for convex polyhedra intersection, it is straightforward to implement, and for high accuracy quadrature rules, it can be expected to generate substantially fewer quadrature points due to the explicit geometry representation. The intersection method is therefore a good candidate for generating initial quadrature rules for our simplification algorithm.

4.6 Numerical verification

To verify the accuracy of the simplified quadratures, we perform a numerical experiment. For a quadrature rule of strength m, all three-dimensional monomials of up to and including order m (i.e. $x^\alpha y^\beta z^\gamma$ where $\alpha + \beta + \gamma \leq m$) are integrated over a cuboid geometry embedded inside the reference hexahedron $[-1, 1]^3$. A rigid body transformation with a non-zero rotation is applied to the embedded geometry to verify that a non-trivial intersection of embedding and background element is handled correctly. Then, the monomials are defined in the local coordinate system of the embedded geometry with an additional translation such that the integrals do not vanish due to anti-symmetry of monomials with odd exponents. In this local coordinate system, the exact integrals can be computed analytically as volume integrals of polynomials over a cuboid domain. As all integrals are non-zero, relative errors between the integrals approximated using quadratures and the exact integrals can be computed. The results are shown in Figure 4, where each point represents the relative error for a single monomial. In our implementation only quadrature rules of strengths 1, 2, 3, 5 and 10 are implemented, although a simplification to any strength is possible. For the results with initial rules in between which are shown in the figure (indicated by greyish blue), the closest rule with a higher strength was used. This rule was also used as a source for the corresponding simplified rule. The results show that the relative error using the simplified quadratures is of the same order of magnitude (close to machine epsilon) as the relative error using the initial non-simplified quadratures.

5 CONDITIONING OF SYSTEM MATRICES

A well-known problem of fictitious domain, immersed, or embedded simulation methods, is that of the poor conditioning of system matrices when the intersection of a background cell and the embedded geometry is small compared to the size of the background cell. We discuss the cause of this ill-conditioning in Section 5.1. We adapt an Additive-Schwarz preconditioner that was recently introduced in the FCM community in Section 5.2, and finally discuss stabilization in Section 5.3 to ensure robustness in all cases.

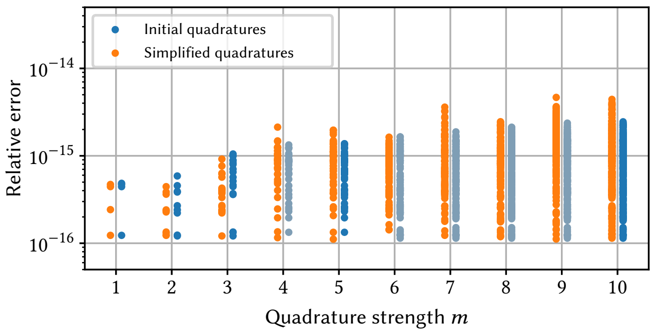


Fig. 4. Relative error of numerically approximated integrals for a range of quadrature strengths. For each quadrature strength m, all three-dimensional monomials of up to and including order m were integrated over a cuboid geometry embedded inside the reference hexahedron.

5.1 The cause of ill-conditioning

Ill-conditioning of immersed methods was recently studied in detail in the context of the FCM by de Prenter et al. [2017]. The problem can be illustrated in the 1D context. When a 1D linear element is cut on one end, the portion inside the embedded geometry consists of two linear functions that take radically different values. Scaling the functions appropriately (Jacobi preconditioning) thus effectively improves the conditioning for first-order elements, also in 2 and 3 dimensions. However, when considering a quadratic element, the portion inside the embedded geometry now contains three almost linear functions. Scaling does not resolve the problem in this case, because the three functions are still nearly linearly dependent after scaling. In summary, the ill-conditioning is a result of a combination of poorly scaled and nearly linearly dependent basis functions.

5.2 Additive-Schwarz preconditioning

De Prenter et al. [2017] proposed an algebraic preconditioner based on local orthogonalization of basis functions. The similarity of this preconditioner to Additive-Schwarz preconditioners was recognized in a subsequent publication [de Prenter et al. 2019]. We refer to these publications for in-depth details on the preconditioning scheme, and only summarize the key features below.

The preconditioner is purely algebraic, and so is applicable to any linear combination of the mass matrix $M \in \mathbb{R}^{n \times n}$ and stiffness matrix $K \in \mathbb{R}^{n \times n}$. We let C be such a linear combination. For a given element K, we denote by R_K the restriction to the element, i.e. when applied to a vector it selects the entries corresponding to nodes in K. The preconditioner S is then given by

$$S := \sum_K R_K^T C_K^{-1} R_K = \sum_K S_K, \tag{13}$$

where $C_K = R_K C R_K^T$ is the submatrix of C obtained when selecting only the rows and columns corresponding to the nodes in K. Note that this is *not* the same as the element matrices involved in the assembly process, as C_K already includes contributions from neighboring elements.

It is not necessary to include every element K in the summation. It is enough only to consider cut elements. The remaining nodes that do not belong to any cut element K can instead be preconditioned with a diagonal preconditioner. Further cost savings can be obtained by only considering cut elements with a *volume fraction* (ratio of embedded intersection volume to background cell volume) lower than a certain threshold. We have found $0.5 - 0.7$ to work well, but the optimal value is rather application-dependent. Including fewer elements in the summation makes the construction of the preconditioner less expensive, but may increase the number of solver iterations. We remark that the number of non-zeros in S is at most the same as C, and for applications where only a small fraction of elements are cut by the embedded geometry, the preconditioner is similar to a diagonal matrix plus some additional dense blocks.

Although in principle C_K should be invertible, small intersections with the geometry may lead to inaccurate inverses. Jomo et al. [2019] suggested to use a pseudo-inverse to improve robustness in these cases. However, they only considered positive definite matrices, whereas the system matrices encountered with non-linear material models may be indefinite and contain negative eigenvalues. Their truncation of small eigenvalues would therefore lead to a singular preconditioner, while indefinite solvers like MINRES [Paige and Saunders 1975] require a positive definite preconditioner. To ensure a positive definite preconditioner, we instead *reflect* the eigenvalues of C_K, so that if λ_K is an eigenvalue, we rebuild a reflected pseudo-inverse by setting the kth eigenvalue to

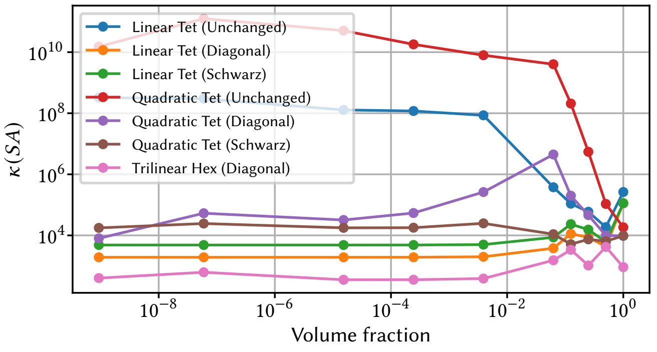
$$\lambda_k(S_K) = \begin{cases} |\lambda_k(C_K)|^{-1} & \text{if } |\lambda_k(C_K)| \geq \varepsilon\lambda_{\max}(C_K), \\ \varepsilon|\lambda_{\max}(C_K)|^{-1} & \text{otherwise} \end{cases} \tag{14}$$

with $\varepsilon \approx 10^{-13}$. Since both the eigenvectors and magnitude of the eigenvalues are preserved with respect to the original formulation, we have found the preconditioner to still be an effective treatment for the ill-conditioning caused by small intersections.

5.3 Quadrature stabilization

If the volume fraction (see Section 5.2) of a cut finite element is permitted to become arbitrarily small, the mass and stiffness matrices may become arbitrarily close to singular. In this case, the preconditioner from Section 5.2 will also break down due to numerical error. Like Patterson et al. [2012], we also discard background elements with a volume fraction lower than a certain threshold, e.g. 10^{-4}. These cells can be expected to have little impact on the simulation output, so points in the embedded geometry are instead extrapolated from the closest element still present in the background mesh.

However, the ill-conditioning for higher-order elements kicks in already at reasonably high volume fractions, and may cause issues for the Schwarz preconditioner far above the volume fraction threshold. In the FCM community, it is common practice for simulations of elasticity to insert Gauss quadrature points for the uncut cell in the portion of the element K that does not intersect the embedded geometry, i.e. $K \setminus \Omega$. This part is then simulated with a very soft material that has a vanishingly small impact on the embedded geometry, yet prevents the coefficient matrix from becoming completely singular. It is however possible to imagine geometry which would contain all these standard Gauss points, yet have a very low



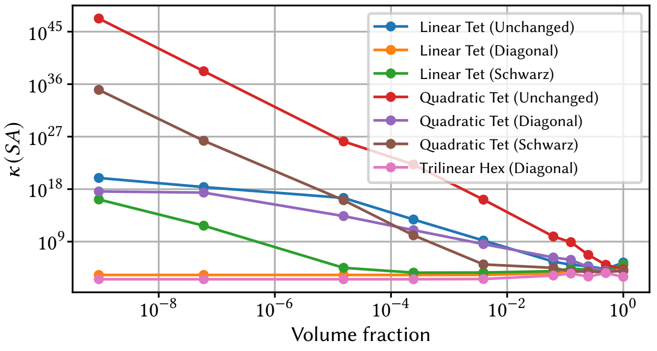


Fig. 5. Condition numbers κ for the preconditioned system $SA = S(M + (\Delta t)^2 K)$ for different preconditioners for linear and quadratic tetrahedra, with (top) and without (bottom) stabilization of the quadrature rule.

volume fraction and potentially lead to poor conditioning. In order to ensure robustness, we therefore insert all the Gauss points of the uncut cell, but instead of framing the problem in terms of material coefficients, we directly scale the weights of the quadrature by a small factor, say, 10^{-6}. This stabilization procedure ensures that all basis functions are supported on the original finite element domain, although poorly scaled. We remark that we have found it sufficient only to stabilize the mass matrix when direct solvers are employed, and first-order elements need no stabilization at all when a diagonal preconditioner is used.

5.4 Numerical verification

In order to verify the efficacy of our preconditioner and quadrature stabilization, we construct a $4 \times 4 \times 4$ voxel grid covering the domain $[0, 4/3]^3$. We then embed another box $[0, 1+\epsilon/3]^3$ into the voxel grid for decreasing values of $\epsilon \in (0, 1]$, so that the mesh contains a mixed amount of interior cells and cut cells that are small in one, two and three coordinate directions. The cell with the smallest intersection with the embedded intersection has volume $(\epsilon/3)^3$, and we use its *volume fraction*, i.e. the ratio between the embedded intersection volume and the volume of the background cell, as a measure of the smallest overlap with the embedded geometry. We consider the condition number of the coefficient matrix associated with a typical Backward Euler step $M + (\Delta t)^2 K(u)$ evaluated with displacement $\mathbf{u}(\mathbf{X}) = (u_x, u_y, u_z) = (X, X, -X)$, $\Delta t = 1/60$ and Dirichlet boundary condition $\mathbf{u} = 0$ for $\mathbf{X} = 0$. We note that for a matrix A, the spectrum of SA and $S^{1/2}AS^{1/2}$ are the same due to similarity, so we study the condition number of SA. Similar experiments have been carried out in the FCM literature but with a focus on hexahedral meshes. Hence, we instead consider uniform linear and quadratic tetrahedral meshes as an additional data point. The material model used was the Stable Neo-Hookean model [Smith et al. 2018] with Young's modulus $3 \cdot 10^6$ Pa and Poisson's ratio 0.4.

The effectiveness of the Additive-Schwarz-type preconditioner is demonstrated in Figure 5. The ill-conditioning problems associated with small intersection volumes are effectively resolved when the quadrature is stabilized, here with a stabilization parameter of 10^{-6}. However, the preconditioner fails to resolve ill-conditioning when no stabilization is used, which demonstrates that stabilization is essential to ensuring robustness. The results also corroborate the effectiveness of a simple diagonal preconditioner for first-order elements (both tet and hex), for which no stabilization is needed.

6 SIMULATION RESULTS

We evaluate our embedded simulation approach, show comparisons with standard FEM, which uses boundary-conforming elements, and compare the effectiveness of linear and higher-order elements. Moreover, we demonstrate that our finite cell method is able to handle different kinds of element types. We consider standard linear and quadratic tetrahedra, denoted as Tet4 and Tet10 respectively, 8-noded (trilinear) hexahedral elements, denoted as Hex8, and 20-noded quadratic Serendipity elements, denoted as Hex20, and employ tetrahedral and hexahedral quadrature rules published by Witherden and Vincent [2015]. For the quadratic tetrahedra, we use quadrature that is exact up to polynomial order 3, for Hex8 we use order 2 and for Hex20 we use order 4, and use our simplification algorithm unless otherwise noted. For integrating the mass matrix terms we use a much higher degree quadrature to guarantee non-singular matrices, but avoid simplification since this quadrature is only needed once. In all experiments we use the non-linear Stable Neo-Hookean material model [Smith et al. 2018]. For the Newton iterations we used the stopping criterion $\|M\dot{\mathbf{v}} - \mathbf{f}^{\text{total}}\| \leq 10^{-5}\|M\mathbf{a}_{\text{rep}}\|$, where finite differences are used to approximate $\dot{\mathbf{v}}$ and gravity is used as a representative acceleration $\mathbf{a}_{\text{rep}}$. The timings in this section are measured on a PC with two 14-core Intel Xeon E5-2690v4 processors. We used our own in-house FEM library written in Rust for all our experiments with the `nalgebra` library for dense matrix-vector operations [Crozet et al. 2019]. We used Nested Cages [Sacht et al. 2015] to generate background surface meshes, TetWild [Hu et al. 2018] to generate tetrahedral meshes, MeshLab [Cignoni et al. 2008] to compute Hausdorff distances for error analysis, the Google GLOP solver to solve all LPs, and Intel MKL 2020 for sparse matrix-vector operations and direct sparse solvers. Direct solvers were used for the twisting cylinder and hollow ball scenarios due to the challenging material parameters, which would otherwise require sophisticated preconditioning or much smaller time steps. We stress that this is also true for FEM; the FCM preconditioner only treats ill-conditioning due to small volume overlaps, not due to stiff material. To make Dirichlet boundary conditions consistent across different meshes, all meshes were adapted to contain identical planar boundary-conforming regions where the BCs were applied.

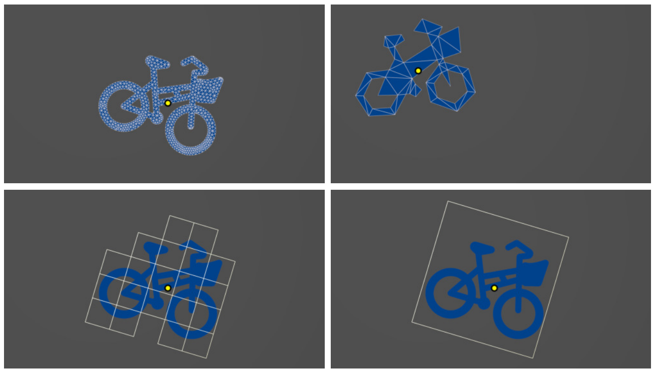


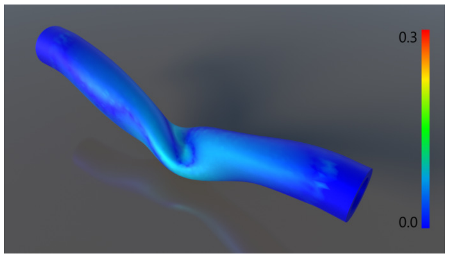
Fig. 6. A comparison of the ability to capture rigid body modes. Top left: The fine FEM mesh (1k linear triangles) is used as ground truth. Top right: The coarse FEM mesh (54 triangles) does not have the correct center of mass. Bottom: FCM captures the correct rotation at both coarse (20 bilinear quads) and very coarse (1 quad) resolutions.

6.1 Rigid body motion

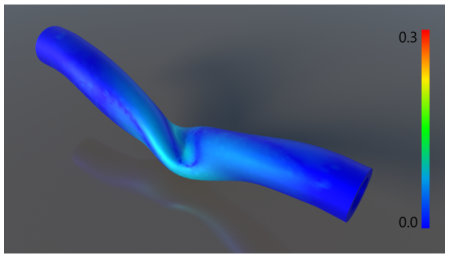
We simulated a stiff fine-resolution 2D bike model with 1k linear triangle elements spinning around its center of mass (see Figure 6 and the supplemental video) and compared it to a coarse simulation with 54 triangles. All discretizations are initialized with the same initial velocity field, which represents an angular velocity around the exact geometry's center of mass. The coarse finite element mesh fails to capture the correct mass distribution of the object and quickly drifts off from the correct center of mass. In comparison, embedding the bike into an FCM mesh with bilinear quads correctly and automatically captures the correct rigid body motion, even when only a single element is used. We conclude that correctly integrating over the embedded domain is crucial to accurately capturing inertial properties of the embedded object.

6.2 Accuracy & performance

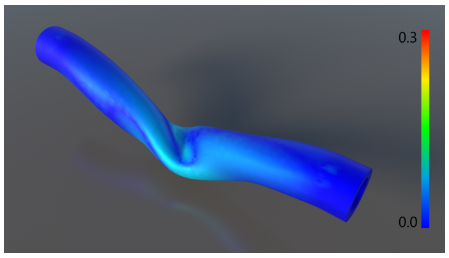
In order to compare the finite element method and the finite cell method using linear and higher-order elements, we simulated the twisting of a hollow cylinder. The length of the cylinder was 16 m. We used a Young's modulus of $5 \cdot 10^6$ Pa and a Poisson's ratio of 0.48. While twisting the cylinder, we get a characteristic deformation in the middle of the model (see the supplementary video). First, we computed an accurate reference solution consisting of 86k boundary-conforming Tet10 elements. The same scene was simulated with Tet4 FEM, Tet10 FEM, Hex8 FCM and Hex20 FCM discretizations of different resolutions (see the supplementary video). For each discretization, the root mean square (RMS) value of the Hausdorff distance from the discretization to the reference solution was computed for the final timestep of the simulation. Based on the results, we picked the RMS value 0.015 m as a representative threshold for when the approximations look sufficiently similar to the reference solution. For each discretization, we picked the resolution with the RMS value closest to the threshold (see Figure 7) and compared the



(a) FCM - 2 016 Hex20 elements, 11 572 nodes, Hausdorff distance (RMS): 0.015 m.



(b) FCM - 98 939 Hex8 elements, 121 692 nodes, Hausdorff distance (RMS): 0.015 m.



(c) FEM - 1 633 559 Tet4 elements, 320 000 nodes, Hausdorff distance (RMS): 0.015 m.

Fig. 7. Comparison of FCM using higher-order Hex20 elements, FCM using Hex8 elements, and FEM using Tet4 elements with a reference solution. We simulated the twisted hollow cylinder shown in the supplementary video with all methods. The figures show the characteristic deformation of the hollow cylinder, where the color-coding represents the Hausdorff distance to the reference solution.

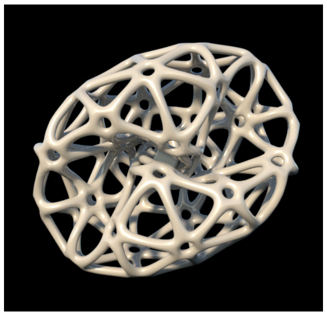
Table 1. Performance comparison of FCM using Hex20 elements, FCM using Hex8 elements, FEM using Tet4 elements, and FEM using Tet10 elements in a simulation of a hollow cylinder (see Figure 7). The table shows the number of elements and nodes and the average time per simulation step.

	FCM		FEM		Reference
	Hex20	Hex8	Tet4	Tet10	solution
# elem.	2 016	98 939	1 633 559	17 695	86 292
# nodes	11 572	121 692	320 000	31 805	135 631
time step	359 ms	2 907 ms	56 321 ms	560 ms	21 157 ms

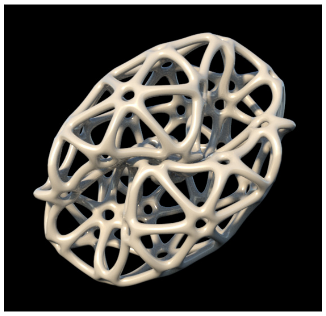
runtimes (see Table 1). Note that for Tet10 the RMS was also 0.015 m. The first-order elements (Hex8 FCM, Tet4 FEM) struggled with slow spatial convergence, and the Tet4 elements in particular. For the very high resolution Tet4 mesh (Figure 7c), the Newton solver only succeeded when projecting the stiffness matrix to semidefiniteness (see e.g. [Smith et al. 2018]). On the other hand, the higher-order elements (Hex20 FCM, Tet10 FEM) performed very well, with the FCM roughly 1.6 times faster than the Tet10 FEM. The results demonstrate that higher-order elements are essential to simultaneously achieving high fidelity and performance in this case, and our embedding methodology unlocks the potential of higher-order methods without conformance to the boundary of the embedded geometry.

6.3 Quadrature construction & simplification

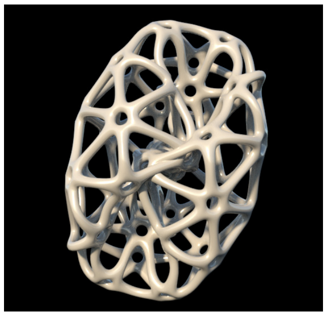
To demonstrate the effectiveness of our quadrature simplification procedure, we repeated the Hex20 FCM simulation without quadrature simplification. We confirmed that the RMS Hausdorff distance was essentially the same, 0.0149 m with simplification and 0.0153 m without. An average time step took 5691 ms, which compared to 359 ms for the simulation with the simplified quadrature means that the simplification enabled a 15.8 times speedup by lowering the associated assembly costs. Simplification reduced the number of points from 5 857 880 to 70 560 — a reduction by a factor of 83. The quadrature generation (Section 4.5) required 255 ms, while the simplification step finished in 5.23 s. In other words, the cost of the simplification algorithm paid off after the first time step.



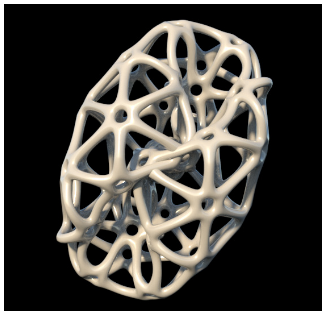
(a) Tet4 FEM - 83 411 elements, 27 766 nodes.



(b) Tet4 FEM - 192 031 elements, 52 076 nodes.



(c) Tet10 FCM - 22 938 elements, 47 591 nodes.



(d) Reference solution.

Fig. 8. Comparison of the final deformation of the hollow ball simulation in Figure 1 for different methods. The FEM with boundary-conforming Tet4 elements is not able to capture the complex deformation of the model correctly. Our method captures the deformation much better while being 8 times faster than the reference solution.

6.4 Complex deformation

In the preceding experiment we showed that our embedded simulation method outperforms standard FEM when using hexahedral elements. In this comparison we simulate the complex deformation behavior of a hollow ball with different holes (see Figure 1) using Tet4 and Tet10 elements. As reference solution, we consider a high-resolution Tet10 boundary-conforming FEM discretization with 192k elements and 332k nodes, at an average cost of 5930 ms per time step. For the comparison we performed the simulation again using a medium and high resolution FEM discretization with Tet4 elements and a lower resolution FCM with Tet10 elements. Figure 8 shows the final deformation of all simulations after the ball was compressed and twisted. The figure shows that the high-resolution FEM simulation with Tet4 elements is not able to capture the complex deformation of the ball correctly, at a cost of 519 ms per time step on average. The coarse Tet10 FCM simulation, on the other hand, is visually almost indistinguishable from the reference solution, at an average cost of 739 ms per time step. Preprocessing took approximately 12s, and the number of quadrature points for boundary elements after simplification were reduced from 27 M to 0.7 M. Although further refinement of the Tet4 FEM mesh might give a comparable solution in quality (at increasing cost), we remark that the Tet10 FCM directly produced a high-quality solution for the coarsest background mesh that we were able to produce that still respects the topological features of the embedded geometry. It is therefore arguably a more reliable (yet competitive) “default" discretization. Finally, even though in this case the coarsest background mesh we were able to produce is visually similar to the exact geometry, the FCM produced a substantially more accurate solution at a modest computational overhead compared to a Tet10 FEM simulation on the same background mesh, which required 607 ms per time step. The maximal RMS Hausdorff distance to the reference solution throughout all frames of the simulation for Tet10 FEM was 0.047, compared to 0.018 for the FCM.

6.5 Iterative solvers

To demonstrate that the preconditioner from Section 5.2 effectively enables the use of iterative solvers for embedded higher-order simulation, we consider the armadillo slingshot scenario shown in Figure 9. The armadillo (Young’s modulus: $5 \cdot 10^5$ Pa, Poisson’s ratio: 0.4) is stretched backwards and then released, resulting in a catapulting motion forward. We consider two Tet10 meshes with 5k and 35k elements, respectively, and we simulate both resolutions with FEM and FCM. The coarse background mesh is shown in Figure 9a. In the case of the FCM, a high-resolution mesh with 124k tetrahedra was embedded. For the coarse model (5k), FEM took on average 184 ms and 102 CG iterations per time step, while FCM required on average 230 ms and 147 CG iterations. For the fine model (35k), FEM took 1465 ms and 231 CG iterations, while FCM took 1683 ms and 293 CG iterations. Only cells with a volume fraction lower than 0.5 were treated with the preconditioner. A higher value would lead to fewer CG iterations, but the preconditioner would be more expensive. The experiment demonstrates that the performance of the preconditioned FCM is on par with that of the same model without any embedding.

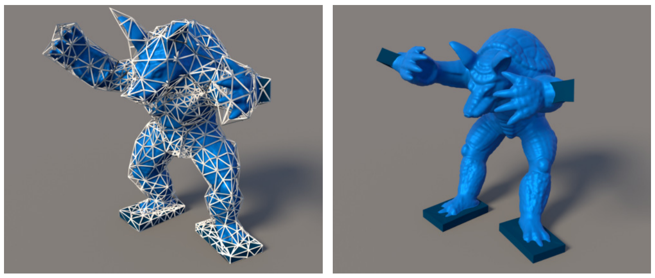

(a) Coarse simulation mesh (b) Deformed model

Fig. 9. FCM simulation of an embedded deformable model with 4916 Tet10 elements and 8634 nodes. Boundary conditions are defined at the dark blue boxes, including one on the back that is used to deform the model.

6.6 Convergence rate

We study the L^2 convergence rate for a static equilibrium problem with a linearly elastic material model and a hemisphere-like geometry. We choose a sequence of regular hexahedral meshes parameterized by mesh width h and measure the error for FEM and FCM Hex8 and Hex20 discretizations relative to a high-resolution boundary-conforming reference solution. The results are presented in Figure 10. Here we see that the poorly approximated geometry for the FEM leads to a dramatically reduced convergence rate, whereas the FCM simulations attain optimal convergence rates $O(h^2)$ and $O(h^3)$. We refer to the technical supplement for a detailed account of the experimental setup and further discussion.

7 CONCLUSION AND FUTURE WORK

Our robust quadrature generation procedure opens up interesting new opportunities for high-fidelity embedded simulation using higher-order elements. The ability to accurately capture complex embedded geometry in larger elements significantly expands the potential applications for higher-order elements.

We have only considered meshes where the entire mesh consists of higher-order elements. We do not believe this is practical for most graphics applications, as linear elements are typically more economical for less demanding scenarios, and moreover may be superior if

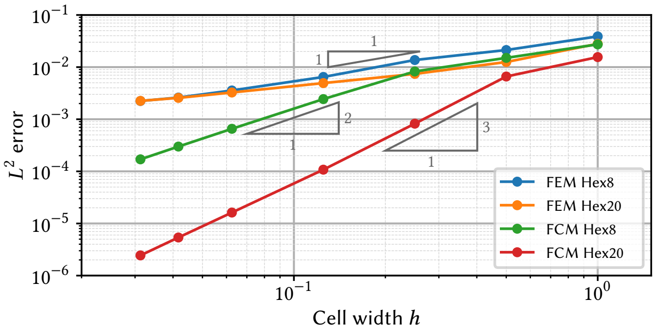


Fig. 10. L^2 errors relative to a high-resolution reference solution for a static equilibrium problem. The slopes of the triangles correspond to $O(h)$, $O(h^2)$ and $O(h^3)$ convergence.

the deformation is highly irregular. Therefore, adaptive embedded methods that select higher-order elements when the displacement and stress fields are sufficiently smooth, and the topology of the embedded object can be captured well, seem like promising avenues for future research.

While we have shown that the Additive-Schwarz preconditioner is effective in treating ill-conditioning and enables iterative solvers in embedded simulations, the necessity to compute eigenvalue decompositions may be prohibitive for some applications in which the majority of elements are cut. This may particularly be problematic for even higher-order elements with more nodes per element than we have presented here, due to the cubic complexity of the eigenvalue decomposition. On the other hand, if the number of cut elements that pass the volume fraction threshold is only a small proportion of the elements, the preconditioner is quite inexpensive. In either case, it would be preferable to develop a preconditioner that would be inexpensive for all applications.

We have not addressed the problem of properly handling embedded Dirichlet boundary conditions. Since we impose no particular constraints on the shape of our background meshes, it is currently possible to design the background mesh so that nodes can be constrained directly as in the standard FEM. However, we plan to investigate more accurate approaches to boundary handling. In a similar vein, we have not considered the handling of contacts. Here it is important to be able to retain some or all of the higher-order convergence, which puts more stringent requirements on the contact model. This is a topic of ongoing research for us, and we have had some promising preliminary results with a variant of the Mortar paradigm, which we hope to fully develop in a future publication. The supplemental video shows a preview of our work on contacts.

Finally, we believe that the robustness and strong theoretical guarantees of our quadrature generation procedure makes it exceptionally well-suited for use in simulations of accurate cutting and fracture, where new quadrature rules may need to be constructed on-the-fly depending on the progression of the simulation.

ACKNOWLEDGMENTS

We would like to thank Dan Koschier for his insightful feedback, and the anonymous reviewers for their valuable suggestions. The Armadillo and Bunny models are courtesy of the Stanford Computer Graphics Laboratory. The frog and bicycle models are courtesy of SVG Repo, the lion figure of OpenClipart, and the hollow ball of GeniyRocka/TurboSquid. This paper was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number BE 5132/5-1.

REFERENCES

Adam W. Bargteil and Elaine Cohen. 2014. Animation of Deformable Bodies with Quadratic Bézier Finite Elements. *ACM Trans. Graph.* 33, 3, Article 27 (2014), 10 pages.

Klaus-Jürgen Bathe. 2006. *Finite element procedures.* Klaus-Jurgen Bathe.

Max Budninskiy, Houman Owhadi, and Mathieu Desbrun. 2019. Operator-adapted wavelets for finite-element differential forms. *J. Comput. Phys.* 388 (2019), 144–177.

Jiong Chen, Hujun Bao, Tianyu Wang, Mathieu Desbrun, and Jin Huang. 2018. Numerical Coarsening Using Discontinuous Shape Functions. *ACM Trans. Graph.* 37, 4, Article 120 (2018), 12 pages.

Jiong Chen, Max Budninskiy, Houman Owhadi, Hujun Bao, Jin Huang, and Mathieu Desbrun. 2019. Material-Adapted Refinable Basis Functions for Elasticity Simulation. *ACM Trans. Graph.* 38, 6, Article 161 (2019), 15 pages.

Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference.* The Eurographics Association.

Sébastien Crozet et al. 2019. nalgebra: a linear algebra library for Rust. https://nalgebra.org

Frits de Prenter, CV Verhoosel, and EH van Brummelen. 2019. Preconditioning immersed isogeometric finite element methods with application to flow problems. *Computer Methods in Applied Mechanics and Engineering* 348 (2019), 604–631.

Frits de Prenter, Clemens V Verhoosel, Gert J van Zwieten, and E Harald van Brummelen. 2017. Condition number analysis and preconditioning of the finite cell method. *Computer Methods in Applied Mechanics and Engineering* 316 (2017), 297–327.

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. 2001. Dynamic Real-Time Deformations using Space & Time Adaptive Sampling. In *ACM Conference on Computer Graphics and Interactive Techniques.* ACM, 31–36.

Sascha Duczek, Fabian Duvigneau, and Ulrich Gabbert. 2016. The finite cell method for tetrahedral meshes. *Finite Elements in Analysis and Design* 121 (2016), 18–32.

Alexander Düster, Jamshid Parvizian, Zhengxiong Yang, and Ernst Rank. 2008. The finite cell method for three-dimensional problems of solid mechanics. *Computer methods in applied mechanics and engineering* 197, 45-48 (2008), 3768–3782.

P. Faloutsos, M. Van de Panne, and D. Terzopoulos. 1997. Dynamic free-form deformations for animation synthesis. *IEEE Trans. on Vis. and Comp. Graph.* 3, 3 (1997).

Christian Hafner, Christian Schumacher, Espen Knoop, Thomas Auzinger, Bernd Bickel, and Moritz Bächer. 2019. X-CAD: optimizing CAD models with extended finite elements. *ACM Trans. Graph.* 38, 6 (2019), 1–15.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4, Article 117 (2020).

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (2018).

John D Jakeman and Akil Narayan. 2018. Generation and application of multivariate polynomial quadrature rules. *Comp. Methods in Applied Mech. and Eng.* 338 (2018).

Doug L James, Jernej Barbič, and Christopher D Twigg. 2004. Squashing cubes: Automating deformable model construction for graphics. In *ACM SIGGRAPH Sketches.*

John N Jomo, Frits de Prenter, Mohamed Elhaddad, Davide D'Angella, Clemens V Verhoosel, Stefan Kollmannsberger, Jan S Kirschke, Vera Nübel, EH van Brummelen, and Ernst Rank. 2019. Robust and parallel scalable iterative solutions for large-scale finite cell analyses. *Finite Elements in Analysis and Design* 163 (2019), 14–30.

Peter Kaufmann, Sebastian Martin, Mario Botsch, Eitan Grinspun, and Markus Gross. 2009. Enrichment Textures for Detailed Cutting of Shells. *ACM Trans. Graph.* 28, 3 (2009), 50:1–50:10.

Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. 2008. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 105–115.

Vahid Keshavarzzadeh, Robert M Kirby, and Akil Narayan. 2018. Numerical integration in multiple dimensions with designed quadrature. *SIAM Journal on Scientific Computing* 40, 4 (2018), A2033–A2061.

Lily Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun. 2009. Numerical Coarsening of Inhomogeneous Elastic Materials. *ACM Trans. Graph.* 28, 3, Article 51 (2009), 8 pages.

Dan Koschier, Jan Bender, and Nils Thuerey. 2017. Robust EXtended Finite Elements for Complex Cutting of Deformables. *ACM Trans. Graph.* 36, 4, Article 55 (2017).

László Kudela, Nils Zander, Stefan Kollmannsberger, and Ernst Rank. 2016. Smart octrees: Accurately integrating discontinuous functions in 3D. *Computer Methods in Applied Mechanics and Engineering* 306 (2016), 406–426.

Tassilo Kugelstadt, Dan Koschier, and Jan Bender. 2018. Fast Corotated FEM using Operator Splitting. In *Computer Graphics Forum*, Vol. 37.

Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. 2009. Interactive physically-based shape editing. *Comp. Aided Geom. Design* 26, 6 (2009).

Neil Molino, Zhaosheng Bao, and Ron Fedkiw. 2004. A Virtual Node Algorithm for Changing Mesh Topology During Simulation. *ACM Trans. Graph.* 23, 3 (2004).

M. Müller, M. Teschner, and M. Gross. 2004. Physically-based simulation of objects represented by surface meshes. In *Computer Graphics International.* 26–33.

B. Müller, F. Kummer, and M. Oberlack. 2013. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *Internat. J. Numer. Methods Engrg.* 96, 8 (2013), 512–528.

Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. 2006. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum* 25, 4 (2006), 809–836.

Matthieu Nesme, Paul G. Kry, Lenka Jeřábková, and François Faure. 2009. Preserving Topology and Elasticity for Embedded Deformable Models. *ACM Trans. Graph.* 28, 3, Article 52 (2009), 9 pages.

Matthieu Nesme, Yohan Payan, and François Faure. 2006. Animating shapes at arbitrary resolution with non-uniform stiffness. In *Proc. VRIPHYS.*

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization.* Springer Science & Business Media.

Christopher C Paige and Michael A Saunders. 1975. Solution of sparse indefinite systems of linear equations. *SIAM journal on numerical analysis* 12, 4 (1975), 617–629.

Jamshid Parvizian, Alexander Düster, and Ernst Rank. 2007. Finite cell method. *Computational Mechanics* 41, 1 (2007), 121–133.

Taylor Patterson, Nathan Mitchell, and Eftychios Sifakis. 2012. Simulation of Complex Nonlinear Elastic Bodies using Lattice Deformers. *ACM Trans. Graph.* 31, 6 (2012).

Olivier Rémillard and Paul G Kry. 2013. Embedded thin shells for wrinkle simulation. *ACM Trans. Graph.* 32, 4 (2013), 1–8.

Alec R. Rivers and Doug L. James. 2007. FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. *ACM Trans. Graph.* 26, 3 (2007), 82.

SH Roth, Markus H Gross, Silvio Turello, and Friedrich R Carls. 1998. A Bernstein-Bézier Based Approach to Soft Tissue Simulation. In *Computer Graphics Forum*, Vol. 17.

Ernest Ryu and Stephen Boyd. 2014. Extensions of Gauss Quadrature Via Linear Programming. *Foundations of Computational Mathematics* 15 (2014).

Leonardo Sacht, Etienne Vouga, and Alec Jacobson. 2015. Nested cages. *ACM Trans. Graph.* 34, 6 (2015), 1–14.

Dominik Schillinger and Martin Ruess. 2015. The Finite Cell Method: A review in the context of higher-order structural analysis of CAD and image-based geometric models. *Archives of Computational Methods in Engineering* 22, 3 (2015), 391–455.

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids. In *ACM SIGGRAPH Courses*. 1–50.

Eftychios Sifakis, Kevin G Der, and Ronald Fedkiw. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 73–80.

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable neo-hookean flesh simulation. *ACM Trans. Graph.* 37, 2 (2018), 12.

Daniel Weber, Jan Bender, Markus Schnoes, André Stork, and Dieter Fellner. 2013. Efficient GPU Data Structures and methods to Solve Sparse Linear Systems in Dynamics Applications. *Computer Graphics Forum* 32, 1 (2013), 16–26.

Daniel Weber, Johannes Mueller-Roemer, Christian Altenhofen, André Stork, and Dieter Fellner. 2015. Deformation simulation using cubic finite elements and efficient p-multigrid methods. *Computers & graphics* 53 (2015), 185–195.

Freddie D Witherden and Peter E Vincent. 2015. On the identification of symmetric quadrature rules for finite element methods. *Computers & Mathematics with Applications* 69, 10 (2015), 1232–1241.

Yuan Xu. 1997. On orthogonal polynomials in several variables. *Special functions, q-series and related topics, The Fields Institute for Research in Mathematical Sciences, Communications Series* 14 (1997), 247–270.