Noname manuscript No.
(will be inserted by the editor)

Numerical integration of discontinuous functions: moment fitting and smart octree

Simeon Hubrich · Paolo Di Stolfo · László Kudela · Stefan Kollmannsberger · Ernst Rank · Andreas Schröder · Alexander Düster

Received: date / Accepted: date

Abstract A fast and simple grid generation can be achieved by non-standard discretization methods where the mesh does not conform to the boundary or the internal interfaces of the problem. However, this simplification leads to discontinuous integrands for intersected elements and, therefore, standard quadrature rules do not perform well anymore. Consequently, special methods are required for the numerical integration. To this end, we present two approaches to obtain quadrature rules for arbitrary domains. The first approach is based on an extension of the moment fitting method combined with an optimization strategy for the position and weights of the quadrature points. In the second approach, we apply the smart octree, which generates curved sub-cells for the integration mesh. To demonstrate the performance of the proposed methods, we consider several numerical examples, showing that the methods lead to efficient quadrature rules, resulting in less integration points and in high accuracy.

Simeon Hubrich
E-mail: hubrich@tuhh.de
Alexander Düster
E-mail: alexander.duester@tuhh.de
Numerical Structural Analysis with Application in Ship Technology (M-10), Hamburg University of Technology, Am Schwarzenberg-Campus 4 (C), 21073 Hamburg, Germany

Paolo Di Stolfo
E-mail: paolo.distolfo@sbg.ac.at
Andreas Schröder
E-mail: Andreas.Schroeder@sbg.ac.at
Department of Mathematics, University of Salzburg, Hellbrunner Str. 34, 5020 Salzburg, Austria

László Kudela
E-mail: laszlo.kudela@tum.de
Stefan Kollmannsberger
E-mail: kollmannsberger@bv.tum.de
Ernst Rank
E-mail: rank@bv.tum.de
Chair for Computation in Engineering, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany

1 Introduction

The mesh generation process of the classical finite element method (FEM) can become labor-intensive or practically impossible when facing problems that exhibit a complex geometry or consist of geometrical features such as voids, material inclusions, or cracks. In the past few decades, novel discretization methods have therefore been introduced to simplify the mesh generation. Such methods are, for example, the eXtended finite element method (XFEM) [1–4] and the generalized finite element method (GFEM) [5–7] which are based on the partition of unit method (PUM) [8, 9], the fictitious domain technique [10–13], or the finite cell method (FCM) [14–16] which combines the fictitious domain approach with high-order finite elements. The basic idea of these methods is to separate the approximation of the primary variables from the approximation of the geometry. In doing so, the mesh is not restricted to coincide with the boundary or internal interfaces of the problem under consideration, which is why it allows for a fast and simple mesh generation using structured meshes or Cartesian grids, for instance. However, this simplification in the mesh generation comes at the expense of a more sophisticated application of boundary conditions [17–20], a bad conditioning of the equation system [21–25], the necessity of a local enrichment strategy [26–30], and a more elaborate numerical integration method to handle the discontinuous integrands for elements that are intersected by the boundary or by internal interfaces [14–16, 31–34].

In this paper, we focus on the numerical integration of arbitrarily broken elements. To distinguish these elements – which do not have to conform to the boundary or the internal

interfaces – from classical finite elements, will hereinafter denote them as cells. In the context of structural mechanics, we are interested in the computation of the integrals related to the mass and stiffness matrix as well as the body force vector. Thereby, standard quadrature rules are applied for non-broken cells which are not intersected by the boundary or by internal interfaces. These quadrature rules, however, do not perform well for broken cells which face discontinuous integrals of the following form

$$\int_{\Omega_C} \alpha(\mathbf{x}) f(\mathbf{x}) \, \mathrm{d}\Omega \quad . \tag{1}$$

In Eq. (1), Ω_C denotes the cell domain, $f(\mathbf{x})$ is a continuous and smooth function defined over Ω_C, and $\alpha(\mathbf{x})$ is a function introducing a discontinuity. In the framework of the XFEM, $\alpha(\mathbf{x})$ defines the Heaviside function – and, in the context of the FCM, $\alpha(\mathbf{x})$ is the indicator function which is one for points that are located within the physical domain, and zero otherwise.

Since standard quadrature rules do not perform well for the numerical integration of discontinuous functions adaptive integration schemes are required. To this end, one way to compute these integrals is to replace the discontinuous integrand of the cell by an equivalent polynomial and then to perform a standard Gauss quadrature. This idea has been proposed by Ventura and Benvenuti in [35, 36] and has been extended for the application to the FCM by Abedian et al. [37]. Another commonly used scheme is to split the integral in Eq. (1) into two parts

$$\int_{\Omega_C} \alpha(\mathbf{x}) f(\mathbf{x}) \, \mathrm{d}\Omega = \int_{\Omega_A} \alpha(\mathbf{x}) f(\mathbf{x}) \, \mathrm{d}\Omega + \int_{\Omega_B} \alpha(\mathbf{x}) f(\mathbf{x}) \, \mathrm{d}\Omega \tag{2}$$

whereby the integrals on the right-hand side consist of functions that are smooth and continuous over the corresponding integration sub-domains Ω_A and Ω_B, respectively. Having the discontinuous integral subdivided into two continuous integrals, however, does not ensure a straightforward application of the standard quadrature rules since the sub-domains Ω_A and Ω_B exhibit an arbitrary topology. To perform the computation of the integrals over these particular sub-domains, a local integration mesh is thus generally required, consisting of elements in which standard quadrature rules can be applied [2–4, 7, 15, 32–34, 37–45]. In the course of this, broken cells are subdivided into sub-cells, followed by computing the integrals over these particular sub-domains. Since the mesh is only needed for integration purposes, the adjacent cells do not need to conform, and the mesh may thus obtain hanging nodes. Consequently, the accuracy of the integration methods based on local meshes is related to the approximation of the corresponding integration domain. For this reason, there exist different procedures to generate these meshes, such as methods based on a quadtree or an octree subdivision [15, 32, 34, 40], low-order tesselation using triangles or tetrahedrons [3, 4, 46, 47], or applying high-order sub-cells [7, 38, 44, 45, 48–50].

The adaptive integration schemes based on spacetrees subdividing broken cells by a quadtree (in 2D) or octree in (in 3D) have the advantage that they are robust, fully automatic, and allow to control the error in integration. Moreover, their implementation is very simple. The disadvantage, on the other hand, is that – for a quadrature rule of high accuracy – a fine sub-cell mesh is usually required, resulting in a high number of integration points. This high number of integration points, in turn, renders the numerical integration expensive.

To overcome this problem, we present two methods. In the **first** method, we extend the method suggested in [51] where individual quadrature rules are generated for every broken cell by solving the moment fitting equations [52–56]. For the computation of the moments, the integration domain is represented by a triangulated surface mesh and, thus, results in a high accuracy of its geometry description when using fine meshes. Solving the moment fitting equation system is not an easy task since it is nonlinear in terms of the position of the integration points. For this reason, the method developed in [51] follows the idea of fixing the position of the integration points a priori, as suggested by Mousavi et al. [52]. In [51, 56] an adaptive point distribution scheme has been implemented to this end, subdividing the broken cell uniformly into sub-cells before the required integration points are distributed randomly in every sub-cell that is completely located within the integration domain. In doing so, the nonlinear moment fitting equation system turns into a linear system which is then solved by applying a linear least-squares fit. The advantage of this method is that the number of integration points is reduced significantly as compared to the adaptive integration based on an octree subdivision. Moreover, the adaptive point distribution scheme ensures that all integration points are located within the integration domain – which corresponds to the physical domain in the context of the FCM. The disadvantage, on the other hand, is that the integration points provided by the adaptive distribution scheme may lead to a bad conditioning of the moment fitting equation system and, thus, may decrease the accuracy of the generated quadrature rule. This problem is even more pronounced for high-order quadrature rules. In this contribution, we present two approaches to overcome this problem by optimizing the accuracy of the moment fitting quadrature rules where the integration points have to be located within the domain of the broken cell but are not restricted to the integration domain. In the context of the FCM, this means that the points may be located within the physical as well as the fictitious domain of a broken cell. In the first variant of the moment fitting method, we define the position of the integration points a priori. In doing so, we take advantage

of the position of the Gauss-Legendre points, which lead to a good conditioning of the moment fitting equation system and, thus, result in quadrature rules of high accuracy. In the second variant of the moment fitting, we generate quadrature rules by solving an appropriate optimization problem related to the nonlinear moment fitting equations. First investigations in this regard have been presented in [57] where an optimization problem of the moment fitting method is solved drawing on the procedure proposed by Ryu and Boyd [58]. Although efficient quadrature rules are obtained, resulting in a low number of integration points, solving the optimization problem becomes expensive for high-order quadrature rules. For this reason, in this paper, we formulate an alternative optimization problem which is based on the minimization of the residual of the moment fitting equation system. In doing so, the moment fitting results in efficient quadrature rules, yielding optimized points and weights where the points have to be located within the cell but are not restricted to the integration domain.

In the **second** method, we perform the numerical integration based on the smart octree method presented in [45]. The smart octree subdivides the integration domain into subdomains with smooth integrands (see Eq. (2)). Similar to the classical octree, it subdivides the cut cell into eight octants, but uses node-relocation and high-order polynomials for integration sub-cells. It thereby captures the most primitive topological cases directly at the first level of refinement and uses recursion only as a fallback option, until such a primitive case appears. The smart octree thereby inherits the positive features of the standard octree approach, such as robustness and minimal topological complexity. At the same time, it can approximate complex geometries with high accuracy. Due to this fact, the smart octree does, in general, not require many levels of subdivision. This results in a very low number of integration points where all points are located within the integration domain. Thus, the integration can be carried out with high accuracy entirely on the physical domain in the context of the FCM.

The paper is structured as follows: At first, the individual numerical integration methods are introduced and explained in detail in Sect. 2. Next, in Sect. 3, we demonstrate the efficiency and accuracy of the proposed integration methods considering different numerical examples. Here, we start off by investigating the numerical integration of polynomials on different domains before demonstrating how they are applied to the finite cell method. Finally, Sect. 4 concludes the paper giving a summary and an outlook for future works.

2 Numerical integration

2.1 Moment fitting

This section gives a brief overview of the moment fitting method. For more detailed information, the reader is referred to [53, 54, 59, 60].

The idea of the moment fitting is to generate a quadrature rule by solving the moment fitting equations

$$\sum_{i=1}^{n} f_j(\mathbf{x}_i)\, w_i = \int_{\Omega_\mathrm{A}} f_j(\mathbf{x})\, \mathrm{d}\Omega\,, \quad j = 1,...,m \tag{3}$$

where $\mathbf{x}_i$ are the n integration points, w_i are the n corresponding weights, $f_j(\mathbf{x})$ are the m independent basis functions, and Ω_A the domain of interest. In matrix notation the moment fitting equations reads

$$\begin{bmatrix} f_1(\mathbf{x}_1) & \dots & f_1(\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ f_m(\mathbf{x}_1) & \dots & f_m(\mathbf{x}_n) \end{bmatrix} \begin{Bmatrix} w_1 \\ \vdots \\ w_n \end{Bmatrix} = \begin{Bmatrix} \int_{\Omega_\mathrm{A}} f_1(\mathbf{x})\, \mathrm{d}\Omega \\ \vdots \\ \int_{\Omega_\mathrm{A}} f_m(\mathbf{x})\, \mathrm{d}\Omega \end{Bmatrix}, \tag{4}$$

and can be summarized as follows

$$\mathbf{A}\mathbf{w} = \mathbf{b} \tag{5}$$

where $\mathbf{A}$ defines the coefficient matrix composed of the function evaluations of the basis at the integration points, $\mathbf{w}$ is the vector of the weights, and $\mathbf{b}$ is the vector of the individual integrals of the basis functions over the integration domain Ω_A. The integrals of $\mathbf{b}$ are also denoted as moments. Thus, to conclude, the moment fitting can be understood as a general approach to set up a quadrature rule for an arbitrary domain Ω_A.

Since we are interested in the integration of polynomials, we use the orthogonal Legendre polynomials for the basis, which span the three-dimensional tensor product space

$$\mathscr{F} = \left\{ L_r(\xi) L_s(\eta) L_t(\zeta)\,,\ r,s,t = 0,...,p_\mathrm{q} \right\} \quad . \tag{6}$$

In Eq. (6), p_q defines the order of the basis functions that form the integrands of the right-hand side in Eq. (3) and (4). Different methods can be applied to compute these integrals. On the one hand, the volume integrals can be computed by applying special quadrature rules that suit the topology of the broken cell, as presented in [40], or by using space trees like the standard octree [15] or the smart octree [45], for instance. On the other hand, the volume integrals can be transformed into surface integrals utilizing the divergence theorem, as shown in [51]. In doing so, the integrals of the right-hand side read

$$\int_{\Omega_\mathrm{A}} f_j(\mathbf{x})\, \mathrm{d}\Omega = \int_{\Omega_\mathrm{A}} \mathrm{div}\, \mathbf{g}_j(\mathbf{x})\, \mathrm{d}\Omega = \int_{\Gamma_\mathrm{A}} \mathbf{g}_j(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})\, \mathrm{d}\Gamma \tag{7}$$

where Γ_A defines the surface of the integration domain Ω_A and $\mathbf{n}(\mathbf{x})$ its normal vector, which points in outward direction. Further, the vector $\mathbf{g}_j(\mathbf{x})$ defines the anti-derivatives which are computed according to [54] as

$$\mathbf{g}_j(\mathbf{x}) = \frac{1}{3}\begin{Bmatrix} \int f_j(\mathbf{x})\,\mathrm{d}x \\ \int f_j(\mathbf{x})\,\mathrm{d}y \\ \int f_j(\mathbf{x})\,\mathrm{d}z \end{Bmatrix} \quad . \tag{8}$$

In order to perform the integration, a parameterization of the surface is required. A rather simple approach to do so is to triangulate the surface and to apply a Gaussian quadrature on the triangles of the discretized surface [15].

Solving the moment fitting equation system is not straightforward since it is linear in the weights and nonlinear in the position of the integration points. In [57], a nonlinear optimization approach following the method suggested by Ryu et al. [58] has been applied to generate efficient quadrature rules using optimized integration points. Although it could be shown that this method is very efficient – using less integration points than the number of the basis functions – it proved to become expensive when computing quadrature rules of higher order. A remedy to solve the nonlinear moment fitting equations is to follow the approach presented in [51, 53, 54, 56, 59]. To this end, the position of the integration points is selected in advance, thus transforming the nonlinear system into a linear one. In doing so, the number of the integration points is chosen to be equal or higher than the number of the basis functions ($n \geq m$), and the linear equation system is solved using a linear least-squares fit. In [51, 56], the position of the points is selected by applying an *adaptive point distribution* scheme where the broken cells are uniformly subdivided. Then, the integration points are distributed randomly over those sub-cells that are completely located within the physical domain. The advantage of this distribution scheme is that all points are located within the physical domain. The disadvantage, on the other hand, is that the moment fitting equation system becomes ill-conditioned for high-order quadrature rules, resulting in a lower accuracy.

In [51], it is shown that the standard weights of the Gauss-Legendre quadrature can be recovered by applying the moment fitting to a non-broken cell – a cell that is not cut by the physical boundary – when selecting the position of the Gaussian points. In doing so, even the weights of the high-order quadrature rules can be computed within machine precision. Consequently, using the Gaussian points is a good choice also for broken cells because the left-hand side in Eq. (3) and (4) will be the same as for the non-broken cell. This, in turn, means that the condition number of the system remains the same as well, thus ensuring high accuracy for the high-order quadrature rules. Moreover, applying the standard Gaussian points to the moment fitting equation system transforms it into a linear square system so that the number of the integrations points equals the number of the basis function ($n = m$). Although this implies allowing the points to lie within the fictitious domain, we can show that this approach is suitable to consider linear problems of the finite cell method.

2.2 Optimization of quadrature points and weights

We denote by $\mathscr{P}_k$ the set of polynomials of degree up to k on $[-1,1]$. In this one-dimensional setting, it is well-known that, with n Gauss-Legendre points and weights, each polynomial in $\mathscr{P}_{2n-1}$ can be integrated exactly. Also, it can easily be shown that there are no n points and weights, so that each polynomial in $\mathscr{P}_{2n}$ can be integrated exactly. Therefore, the choice of points and weights in the Gauss-Legendre quadrature is optimal in the sense that each polynomial in $\mathscr{P}_k$ can be integrated exactly for the highest degree k possible when n points are used.

However, the situation in higher dimensions is significantly more complex than the one-dimensional one, mainly due to the fact that domains in higher dimensions come in a far greater variety of shapes [58, 61]. Thus, for general domains in higher dimensions, there are no general rules to determine points or weights.

Consider the moment fitting system (5) for m basis functions and $n = m = (p_\mathrm{q}+1)^3$ points. If the points are pairwise distinct, the system is invertible, implying that there is a set of m points, such that each basis function in $\mathscr{F}$ can be integrated exactly. Note that, if we applied the same rule for the number of quadrature points as in one dimension to integrate the set of m functions exactly, a Gauss-type quadrature formula in 3D would require $\left\lfloor \frac{1}{2}\left(p_\mathrm{q}+1\right)\right\rfloor^3$ points, i.e., only approximately 12.5% of the original number of points. Especially in the context of nonlinear problems, where a repeated computation of expensive integrals (such as the stiffness matrix) is required, even a slight reduction of the number of quadrature points can improve the computational complexity of the whole computation in a significant manner. Therefore, it is of great interest to numerically determine cheaper quadrature rules, i.e. rules with fewer points, by which a set of basis functions can still be integrated exactly.

A general approach aiming at the reduction of the number of points and weights n serves to solve an optimization problem related to the moment fitting idea: For a given vector $\mathbf{b}$ of integrals of a set of m basis functions, we aim to find the least number n and a set of n points and weights, such that the residual

$$\|\mathbf{r}\|_2^2 = \|\mathbf{r}(\mathbf{x}_1,\ldots,\mathbf{x}_n)\|_2^2 := \|\mathbf{b} - \mathbf{A}(\mathbf{x}_1,\ldots,\mathbf{x}_n)\mathbf{w}(\mathbf{x}_1,\ldots,\mathbf{x}_n)\|_2^2$$

vanishes, where $\mathbf{A}(\mathbf{x}_1,\ldots,\mathbf{x}_n)$ denotes the moment fitting matrix and $\mathbf{w}(\mathbf{x}_1,\ldots,\mathbf{x}_n)$ denotes the solution of the moment

fitting system (5). More precisely, our strategy aims at finding points $\mathbf{x}_1,\dots,\mathbf{x}_n$ with minimal $n<m$ such that $\|\mathbf{r}\|_2$ is reduced to ε_{mach}, where ε_{mach} is the machine precision. We note that the numerical value of $\|\mathbf{r}\|_2$ heavily depends on the condition number of the matrix $\mathbf{A}$. Since $\mathbf{A}$ is invertible and well-conditioned when the set of $m=\left(p_\mathrm{q}+1\right)^3$ tensor-product Gauss-Legendre points is used, we get the trivial upper bound $n\leq m$, which implies that the minimization problem has a solution.

As the problem belongs to the nonlinear least-squares minimization problem class, several well-established iterative techniques are available for its solution. A well-known algorithm is the Levenberg–Marquardt algorithm that, essentially, performs a combination of the Gauss–Newton method and a gradient descent. Another algorithm, which can also be applied for more general problems, is the Sequential Quadratic Programming (SQP) algorithm.

A major shortcoming of the proposed method is its computational performance.

The first performance issue stems from the fact that the minimal number n of points is not known. More importantly, it is unknown whether, for fixed $n<m$, the function $\|\mathbf{r}\|_2$ has a global optimum; also, the structure and distribution of local minima is unknown. Therefore, we have to estimate n. Since the residual $\|\mathbf{r}\|_2$ reaches machine precision for $n=m$, we may assume that $\|\mathbf{r}\|_2$ decreases when the number of points n is increased. Thus, we may assume that there is a number n, and $\|\mathbf{r}\|_2$ attains machine precision for some n points, while $\|\mathbf{r}\|_2$ does not attain machine precision for any $n-1$ points. Therefore, we may apply a binary search method that starts with 0 and m points to find the optimal number n in approximately $\log(m)$ executions of the optimization method.

A second issue addresses the performance of each execution of the underlying optimization method. The methods are usually sensitive to the choice of initial points. Suboptimal initial points may result in a slow decrease of the error, resulting in a high number of iterations. Thus, it is necessary to set a reasonable limit to the number of iterations. Even for a good choice of initial points, the method may pursue a local minimum that is far off from the optimum. As the structure of the domain of integration is unknown to the method and the computation is performed on the reference element $[-1,1]^3$, it is safest to initially distribute n points at random. We note that, in order to assess the actual minimal number n of points for given $\mathbf{b}$ with high probability, several runs of the binary search method have to be performed.

2.3 Smart octree

As mentioned in the introduction, the advantage of the classical octree-based integration is that it works robustly on any kind of geometric representation. However, it does not account for the more detailed surface information available from the geometric model. Therefore, in order to increase the accuracy of the integration, many levels of octree refinement may be required. This results in a high number of quadrature points, making the simulation expensive. To overcome this issue, the method of smart octrees has been recently introduced in [45]. This algorithm combines the robust features of the octree procedure with an automatic resolution of the intersection topology – including sharp edges and corners that may be present in the cell – and high-order surface representation. In the following, the key aspects of the algorithm are presented.

The standard octree refinement generates eight octants in the cut cells by generating 19 new internal nodes, which can be categorized as follows:

- edge nodes, lying on the bisection point of the cell edges,
- face nodes, lying on the center of gravity of each cell face,
- a mid-node, created at the center of gravity of the cell.

By construction, the octree procedure distributes the internal nodes to pre-defined locations, the center of gravity of the edges, faces, and the cell. The key idea of the smart octree method is to relay this constraint. The internal nodes can be moved around freely, as long as they stay on the respective components of the cell (e.g. an edge node remains on an edge). Therefore, before distributing the internal nodes, the method examines the components of the cell and categorizes them as follows:

- If an edge is intersected by the interface, it is identified as an active edge. The corresponding edge node is moved onto the intersection point.
- If a face is intersected by the interface, it is identified as an active face. For a given active face, the face node is shifted onto the intersection curve between this face and the interface.

Finally, if the cell contains an active element, the mid-node is shifted onto the interface while remaining inside the cell. The concept of distributing internal nodes is depicted in Fig. 1.

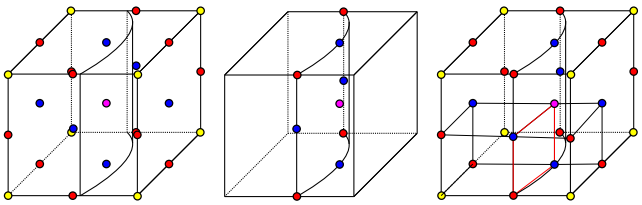


Fig. 1: Smart octree generation. The nodes on active edges, faces, and the mid-node are moved onto the interface. For simplicity, only two resulting sub-cells are shown [45]

While the location of active edge nodes becomes completely constrained by the intersection point, the active face nodes and mid-nodes have remaining degrees of freedom. For example, an active face node needs to lie on the intersection curve between the interface and the face, but can be moved along this intersection curve freely. Further, an active mid-node can be shifted to any location on the interface, as long as it remains inside the cell.

The freedom in moving the active face nodes and the mid-node becomes useful when the cell contains sharp edges or corners. Consider the situation (also depicted in Fig. 2) of a sharp edge e entering the cell through one of its faces f. In this case, the active face node on f is constrained to lie on the intersection point $e \cap f$, therefore its location becomes completely prescribed. Further, the constraint on the location of the mid-node becomes stricter: it has to lie not only on the interface, but also on the sharp edge e. However, along this edge e, it can still be shifted freely. The last remaining degree of freedom of the mid-node becomes completely constrained when the interface inside the cell contains a corner v. In this case, the active edge- and face nodes are distributed similarly as before, while the location of the mid-node is prescribed by the location of v. Fig. 3 depicts an example where the interface cutting through the cell contains a corner and three sharp edges.

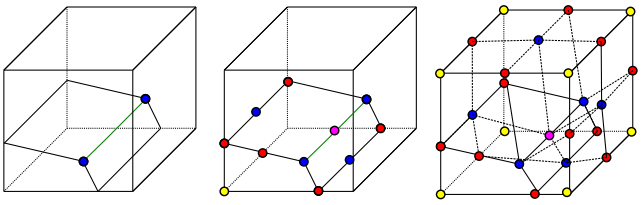


Fig. 2: Sharp edge resolution. The active surface nodes (blue) lie at the intersection between the sharp edges (green color) and the respective face. The mid-node (purple) can be placed on the sharp edge arbitrarily [45]

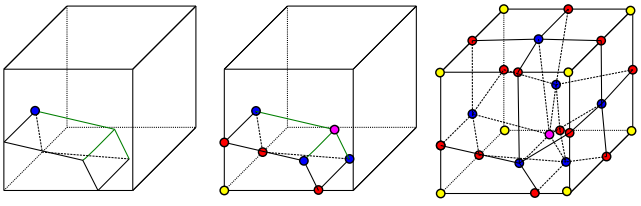


Fig. 3: Sharp corner resolution. The mid-node of the cell is moved onto the sharp corner, while the active surface nodes lie at the intersection between the sharp edges (green color) and the respective face [45]

The outlined method requires that each active edge, face, and cell can uniquely be associated to a single internal node. If this requirement is violated, the smart octree method subdivides the cell into eight equal octants by a simple octree refinement step. The algorithm then attempts to resolve the interface in each of the resulting sub cells. This octree-like refinement step is repeated recursively until the criterion of a single internal node per edge/face/cell is fulfilled. If a maximum number of refinement levels k_{max} is reached, and the cut case can still not be resolved by smart octree decomposition, the *fallback strategy* is to compute integrals by classical octree.

The outlined steps of the smart octree method partition the cut cells into eight octants, where the domain interface is approximated linearly. If the geometric model is composed of high-order entities, the trilinear cells are reparametrized using the method of *blending function interpolation* [62] in order to conform with these curved boundaries. To this end, a local reparametrization of the interface is applied by interpolating a set of sample points projected onto the interface. The sample points are interpolated using Lagrangian shape functions. In Section 3.1.3, it will be demonstrated that the accuracy of the numerical integration based on smart octrees is controlled by the polynomial order p_{B} of the interpolating polynomials.

3 Numerical examples

In this section, we discuss the performance of the proposed integration methods. To this end, we start off by investigating the accuracy in the integration of polynomial integrands for broken cells. Next, we combine the integration methods with the FCM and study the accuracy and efficiency considering problems of different complexity.

3.1 Cell cut by a sphere

In the first example, we contemplate a problem with an integration domain including a curved surface – a cell that is cut by a sphere. In doing so, we consider a hexahedral cell of the following domain

$$\Omega_{\mathrm{C}} = [0,\,1]^3 \quad . \tag{9}$$

The geometry of the sphere is given by the following level set function

$$\phi\left(\mathbf{x}\right) = (x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2 - r^2 \quad , \tag{10}$$

where the center coordinates x_c, y_c and z_c of the sphere and its radius r are given as

$$x_c = y_c = z_c = 0 \quad \text{and} \quad r = 1 \quad . \tag{11}$$

Thus, the integration domain of the broken cell is defined as

$$\Omega_{\mathrm{A}} = \{\mathbf{x} \in \Omega_{\mathrm{C}} : \phi(\mathbf{x}) \leq 0\} \quad . \tag{12}$$

Consequently, the geometry of the integration domain describes an eighth of a sphere, as depicted in Fig. 4.

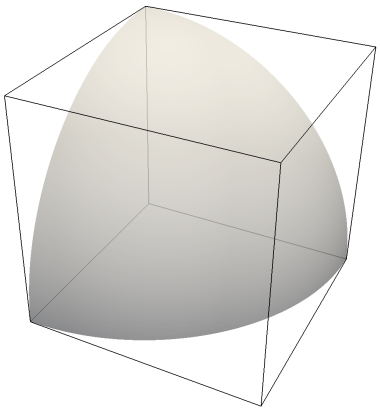


Fig. 4: Geometry of the integration domain of the broken cell cut by a sphere

3.1.1 Moment fitting

To generate quadrature rules that are *exact* up to a certain order, the integrals of the right-hand side in Eq. (4) have to be computed *exactly*. Since the integration domain is described by an eighth of a sphere, neither the quadrature of the right-hand side of Eq. (4) as a volume integral applying an octree scheme or a surface integral based on triangulation will be exact. Due to this fact, we computed the integrals *symbolically* using *Wolfram Mathematica* [63]. An appropriate choice for the position of the integration points is required to provide a good conditioning of the matrix on the left-hand side in Eq. (4). On this account, we study the condition number of the moment fitting equation system applying the points provided by the adaptive point distribution (APD) scheme provided in [51] and compare it to those using the position of the standard Gauss-Legendre points (GLP). To this end, we compute the condition number κ for different quadrature orders $p_{\mathrm{q}} = 0, ..., 16$ which are required when applying high-order shape functions in the FCM. Here, κ of $\mathbf{A}$ is defined as

$$\kappa = \frac{\sigma_{\max}}{\sigma_{\min}} \quad , \tag{13}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum singular values of the system, computed by the *LAPACK* routine *DGELSS* [64]. The condition number of both point sets is plotted in Fig. 5. As it can be seen from the figure, the GLP yield a much better conditioning than the APD. Applying the GLP improves the conditioning of the moment fitting equations system significantly. Moreover, using the GLP leads to the same conditioning of the system for any problem under consideration since the left-hand side in Eq. (4) is independent of the integration domain and only depends on the position of the integration points and the predefined basis functions. The conditioning for the APD, on the other hand, varies from problem to problem since the position of the integration points depends on the topology of the broken cell.

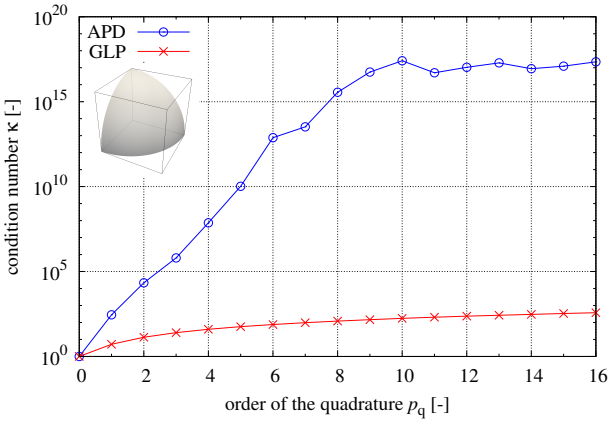


Fig. 5: Condition number of the moment fitting equation system applying different quadrature orders

Due to the better conditioning of the moment fitting equation system using the GLP, the solvability of the system is improved significantly. Next, to point out this fact, we consider the L_2-norm of the residual $||\mathbf{r}||_2$ of the moment fitting equations where the residual is defined as

$$r_j = \int_{\Omega_{\mathrm{A}}} f_j(\mathbf{x})\,\mathrm{d}\Omega - \sum_{i=1}^{n} f_j(\mathbf{x}_i)\,w_i \,, \quad j = 1, ..., m \tag{14}$$

or, in matrix notation,

$$\begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix} = \begin{Bmatrix} \int_{\Omega_A} f_1(\mathbf{x})\,\mathrm{d}\Omega \\ \vdots \\ \int_{\Omega_A} f_m(\mathbf{x})\,\mathrm{d}\Omega \end{Bmatrix} - \begin{bmatrix} f_1(\mathbf{x}_1) & \dots & f_1(\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ f_m(\mathbf{x}_1) & \dots & f_m(\mathbf{x}_n) \end{bmatrix} \begin{Bmatrix} w_1 \\ \vdots \\ w_n \end{Bmatrix} \quad , \tag{15}$$

where r_j are the individual components of the residual vector of the system. Fig. 6 shows the L_2-norm of the residual $||\mathbf{r}||_2$ applying different quadrature orders $p_{\mathrm{q}} = 1, ..., 16$ of the moment fitting. Here, it can be seen that for the application of the GLP the norm of the residual remains close to zero within machine precision for all quadrature orders,

while it appears to increase if the APD is used. Thus, we could reduce the norm of the residual by a factor within the range between 10^4 and 10^5 for the high-order quadrature rules ($p_\mathrm{q} > 9$).

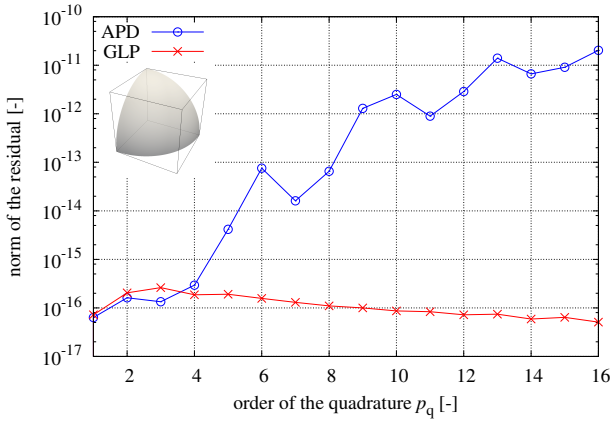


Fig. 6: Norm of the residual of the moment fitting equation system applying different quadrature orders

Next, we consider the conditioning of the moment fitting quadratures. A common definition for the condition number κ_q of quadrature rules is [65]

$$\kappa_\mathrm{q} = \sum_i^{n_\mathrm{g}} |w_i| \quad , \tag{16}$$

where w_i are the individual weights and n_g the number of integration points. To simplify the investigation of the conditioning, we normalize κ_q by the volume $\mathrm{V_A}$ of the integration domain Ω_A

$$\bar{\kappa}_\mathrm{q} = \frac{\kappa_\mathrm{q}}{\mathrm{V_A}} \quad . \tag{17}$$

Fig. 7 shows the normalized condition number $\bar{\kappa}_\mathrm{q}$ of the moment fitting using the APD and the GLP for quadratures of different order p_q. From the figure it can be seen that with increasing the order of the quadrature rule applying the APD, $\bar{\kappa}_\mathrm{q}$ deviates significantly from the optimal value which is 1. This deviation is due to two reasons. Firstly, the moment fitting quadrature is composed of negative and positive weights, and secondly, the individual weights have a high absolute value – which is in particular the case for the high-order quadratures. On the other hand, applying the GLP for the moment fitting results in a good conditioning for the quadrature rules – although some of the weights might still be negative, which is evident from the fact that $\bar{\kappa}_\mathrm{q}$ slightly oscillates between 1 and 1.54.

Finally, we investigate the accuracy of the generated quadrature rules by integrating polynomials of order $p_\mathrm{i} = 0, \ldots, 17$. In doing so, we consider four different orders of the moment

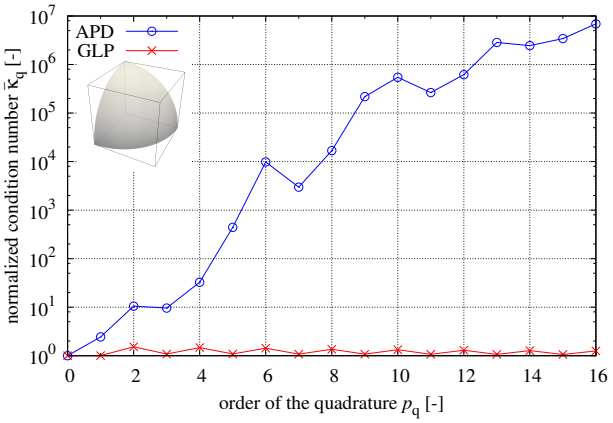


Fig. 7: Condition number of the moment fitting quadratures applying different quadrature orders

fitting quadrature $p_\mathrm{q} = 4, 7, 11, 16$. Since the integrals of the right-hand side in Eq. (4) are computed *symbolically* using *Wolfram Mathematica*, the numerical integration has to be exact for all polynomials where the order of the polynomial is less or equal to the order of the quadrature ($p_\mathrm{i} \leq p_\mathrm{q}$). To measure the accuracy, we consider the relative error e_r in integration, which is defined as

$$e_\mathrm{r} = \left| \frac{I_\mathrm{ex} - I_\mathrm{q}}{I_\mathrm{ex}} \right| \quad , \tag{18}$$

where I_q is the value of the integral obtained with the moment fitting quadrature, and I_ex is the *exact* value of the integral that is computed *symbolically* using *Wolfram Mathematica*. The results of the APD and the GLP are plotted in Fig. 8 and 9, respectively. From the figures, it can be seen that the relative error oscillates around a certain error limit for $p_\mathrm{i} \leq p_\mathrm{q}$ and that a large jump occurs if p_i exceeds p_q. However, the error level of the APD is significantly higher as for the GLP, where the error is close to zero within machine precision. This strong deviation when applying the APD is due to the high values for the norm of the residual of the moment fitting equation system, which originates from the bad conditioning of the system. Ultimately, we can reduce the relative error by a factor up to 10^6 for $p_\mathrm{q} = 16$ by utilizing the GLP. For $p_\mathrm{q} = 16$, the points of the GLP are plotted in Fig. 10a, and the points of the APD scheme as well as the corresponding sub-cell mesh are plotted in Fig. 10b and Fig. 10c, respectively.

3.1.2 Optimization of quadrature points and weights

The performance of the optimization of quadrature points and weights described in Sect. 2.2 is to be assessed in two aspects. Firstly, we want to investigate the optimal number n of quadrature points that lead to a residual $\|\mathbf{r}\|_2$ in the magnitude of the machine precision for the cell cut by a sphere.

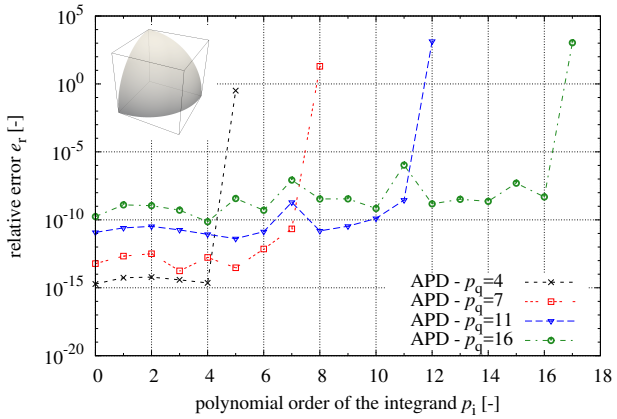


Fig. 8: Relative error in integrating polynomials with the moment fitting using the APD

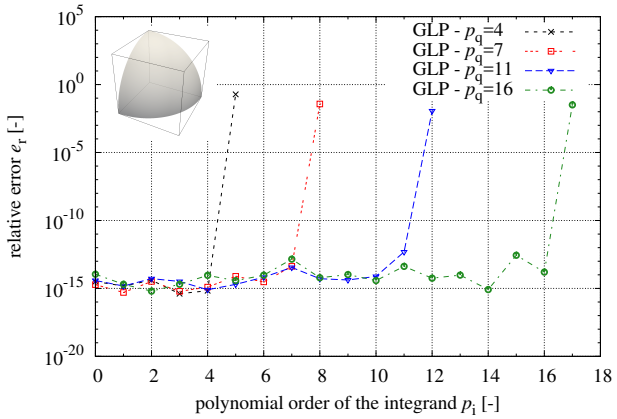


Fig. 9: Relative error in integrating polynomials with the moment fitting using the GLP

Secondly, we aim to check for accuracy by measuring the relative error in the numerical integration of the same trial polynomials as in Sect. 3.1.1, but using the quadrature points and weights determined by the optimization algorithm.

For the first aspect, we apply the binary search method described in Sect. 2.2, which aims to find an approximation to the optimal number of points n such that the residual attains machine precision, but does not attain machine precision for any $n-1$ points. As the optimization algorithm gets highly time-consuming for larger n and p_q, we only try to find the optimal number n for quadrature orders $p_q = 2,\ldots,7$. The results are listed in Table 1. The results indicate that the number of points may be reduced significantly compared to the standard Gauss-Legendre quadrature by applying the optimization algorithm. With an increase in the order of the quadrature, however, the ratio of n to the number of Gauss-Legendre points $\left(p_q+1\right)^3$ becomes poorer – in favor of the GLP. It can be seen that the final residual almost reaches machine precision for each p_q.

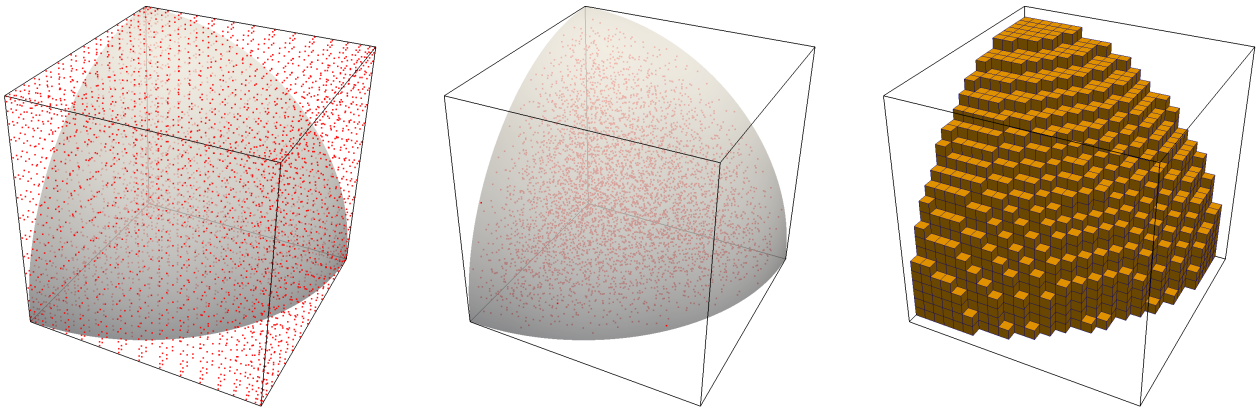
Quadrature order	Optimal n by algorithm	Residual $\|\mathbf{r}\|_2$
2	10 (<27)	$1.5\cdot10^{-16}$
3	27 (<64)	$5.3\cdot10^{-16}$
4	66 (<125)	$5.47\cdot10^{-16}$
5	120 (<216)	$4.0\cdot10^{-15}$
6	246 (<343)	$2.77\cdot10^{-15}$
7	382 (<512)	$7.93\cdot10^{-16}$

Table 1: Comparison of the least number n of points required for the residual $\|\mathbf{r}\|_2$ to reach machine precision in comparison to the number of quadrature points for the Gauss-Legendre quadrature

To assess the accuracy of the quadrature formula originating from the points and weights determined by the optimization algorithm, we measure the error that occurs when the trial polynomials from Sect. 3.1.1 are integrated. The results for integrands of polynomial order $p_i = 2,\ldots,7$ are depicted in Fig. 11. The results show that the relative error remains stable in the range of 10^{-16} to 10^{-13} for all $p_i = 2,\ldots,7$. Integrands of higher degree than the order p_q cannot be integrated with sufficient precision.

3.1.3 Smart octree

For comparison, the 18 different polynomial integrands were also integrated using the smart octree decomposition algorithm, which was discussed in Section 2.3. Because each edge and face contains a single active internal node, the algorithm is able to decompose the cut cell into octants without having to perform any additional octree refinements. Since the cell is cut by a curved geometry, the high-order reparameterization of the sub-cell faces needs to be employed. Fig. 12 shows the resulting integration sub-cells of the smart octree decomposition. As mentioned in Section 2.3, the accuracy of the integration is determined by the polynomial order of the surface interpolation p_B. An example is depicted in Fig. 13, where a 14^{th} order polynomial is integrated by a 20 point quadrature rule, on integration cells with differing p_B. As the polynomial order of the surface approximation increases, the error of the numerical integration drops rapidly. This relationship can also be observed in Fig. 14 and 15, where the error curves of the polynomial integrands are depicted, for $p_B = 4$ and $p_B = 12$, respectively. For a lower quality approximation, such as $p_B = 4$, the error for different polynomial integrands is in the range of $10^{-3} - 10^{-7}$. For a significantly better surface approximation, these error levels can be drastically reduced, as depicted in Fig. 15 for $p_B = 12$. Because the order of the mapping of the integration cell increases as p_B increases, the order of the quadra-



(a) (b) (c)

Fig. 10: Moment fitting quadrature for $p_\mathrm{q} = 16$. **a** GLP. **b** APD. **c** Corresponding sub-cell mesh of the APD

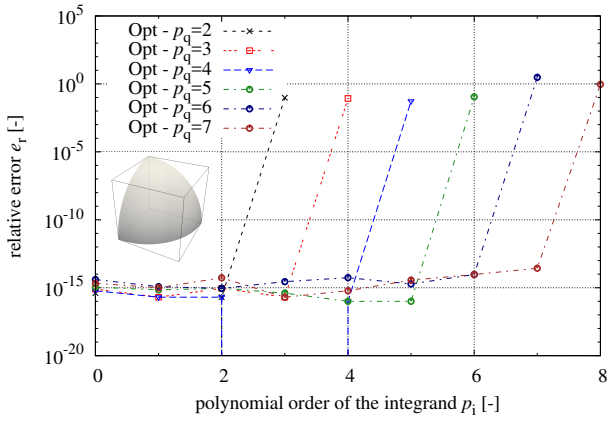


Fig. 11: Relative error in integrating polynomials with points and weights obtained by the optimization method

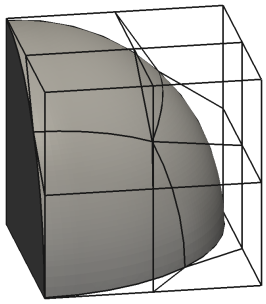


Fig. 12: Smart octree integration mesh generated on a sphere octant

ture rule needs to be increased as well. This can be observed in Fig. 15, where the curve representing $p_\mathrm{q} = 4$ is not able to reach the error levels of higher order quadrature rules. Concerning the computational effort, even in cases of a very high polynomial order p_B, the increase in integration time of the cut elements is moderate.

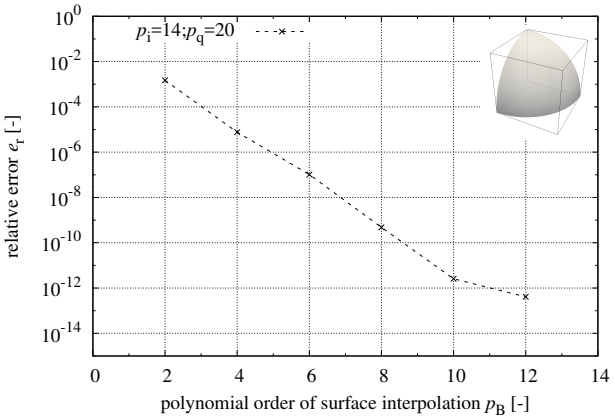


Fig. 13: Relative error of integration with smart octrees, depending on the order of surface interpolation

3.2 Hydrostatic sphere

In the next example, we apply the proposed integration methods to a problem of structural mechanics utilizing the FCM [14–16] and compare the results to the adaptive integration based on an octree subdivision – which is commonly used within the context of the FCM. To this end, we consider a sphere with a uniform traction applied to its surface in normal direction [15, 51, 66], as depicted in Fig. 16a. Thus, the

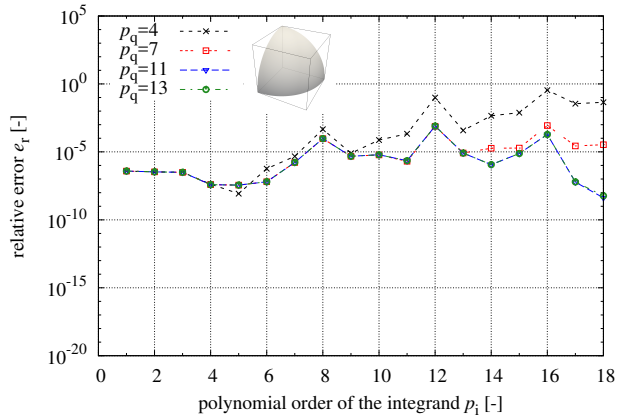


Fig. 14: Relative error in integrating polynomials with smart octrees, $p_B = 4$

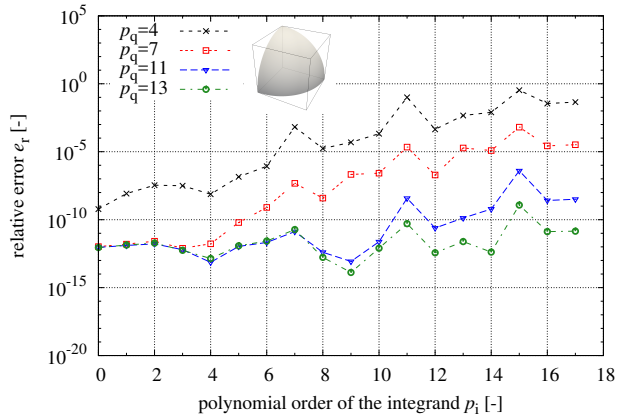


Fig. 15: Relative error in integrating polynomials with smart octrees, $p_B = 12$

sphere is subjected to a hydrostatic stress state. For the analysis, the material behavior of the sphere is assumed to be isotropic linear elastic with Young's Modulus $E = 1.0\,\mathrm{MPa}$ and Poisson's ratio $\nu = 0.3$. The radius of the sphere is $R = 5.0\,\mathrm{mm}$, and the load on its surface is $\bar{t}_n = 1.0\,\mathrm{MPa}$. Provided that the center of the sphere is fixed, the analytical solution of the displacement field of the linear problem reads

$$\mathbf{u}(\mathbf{x}) = \begin{Bmatrix} Cx \\ Cy \\ Cz \end{Bmatrix} \quad \text{with} \quad C = \frac{1}{E}(1-2\nu)\bar{t}_n \quad . \tag{19}$$

Due to the symmetry of the problem under consideration, only one eighth of the sphere has to be taken into account. Thanks to the fictitious domain approach, the sphere is embedded into a fictitious domain, resulting in a domain of an octant which can be easily discretized. Since the solution is smooth and the FCM separates the approximation of the displacement field from the approximation of the geometry, *only* one finite cell is needed for the discretization, see Fig. 16b. Due to the linearity of the displacement field, an Ansatz of order $p = 1$ is sufficient to solve the problem *exactly*. This example presents a problem that is similar to a patch test, where a homogeneous stress field is considered. Using only one cell within a fictitious domain approach allows to obtain the exact solution, provided that the geometry is treated appropriately and the computation of the stiffness matrix and load vector is consistent. Fig. 16b shows the model of the FCM analysis. Here, symmetric boundary conditions are applied to the cell by fixing its face to the bottom in z-direction, its face on the left-hand side in y-direction, and its face on the back in x-direction – while the load is applied to the curved surface of the physical domain which is located within the cell.

As it can be seen in Fig. 16b, the cell is subdivided into two domains. Here, only the physical domain defined by the sphere is of interest for the computation of the stiffness matrix. Consequently, the physical domain has to be resolved during the analysis. In the framework of the FCM, this circumstance is taken into account by introducing the indicator function

$$\alpha(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \phi(\mathbf{x}) \leq 0 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where the domain of the sphere is implicitly defined by following level set function

$$\phi(\mathbf{x}) = x^2 + y^2 + z^2 - r^2 \quad . \tag{21}$$

Having defined the indicator function $\alpha(\mathbf{x})$, the physical domain can be resolved very easily by a suitable integration mesh. In the case of the presented moment fitting methods, the physical domain is needed for the computation of the integrals of the right-hand side. To this end, the surface covering the eighth of the sphere is given by a triangulated surface mesh consisting of $525,718$ triangles. The surface mesh is depicted in Fig. 16c. Computing the volume, the relative error results in a value of about 10^{-6}. As can be inferred from Fig. 17, hierarchical refinements are required to obtain the same error level applying the adaptive integration based on an octree subdivision 8. Here, the error in volume is plotted over the total number of integration points, applying a Gaussian quadrature of order $p_q = 2$ on every sub-cell. The corresponding sub-cell mesh of 8 hierarchical octree refinements is depicted in Fig. 16d, resulting in a number of $2,061,424$ integration points.

For the application of the inhomogeneous Neumann boundary conditions, a parametric description for the curved part of an eighth of the sphere is needed – where the load acts in normal direction. To this end – in the case of the adaptive integration and the moment fitting – we use the triangular

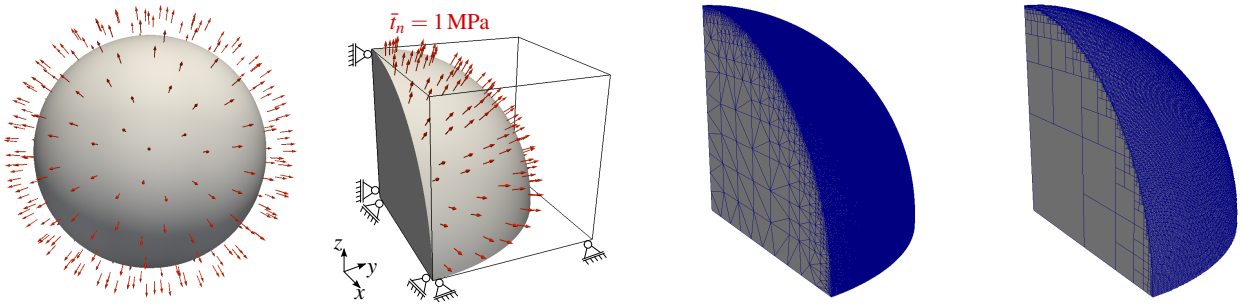


Fig. 16: Sphere under hydrostatic stress state. **a** Setup of the problem. **b** The FCM model using one finite cell. **c** Triangulated surface of the eighth of the sphere. **d** Octree mesh with 8 refinements.

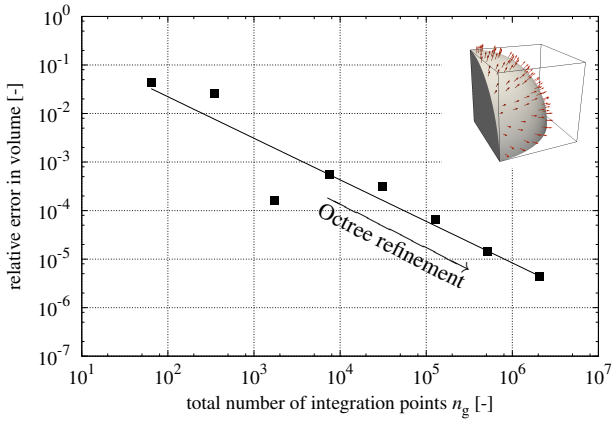


Fig. 17: Relative error in volume applying the adaptive integration based on an octree subdivision for $p_q = 2$

mesh depicted in Fig. 16c and only take triangles into account that describe the curved part of the sphere's surface. Having the parametric description of the surface, the corresponding load vector is computed performing a numerical integration on every triangle as described in [15].

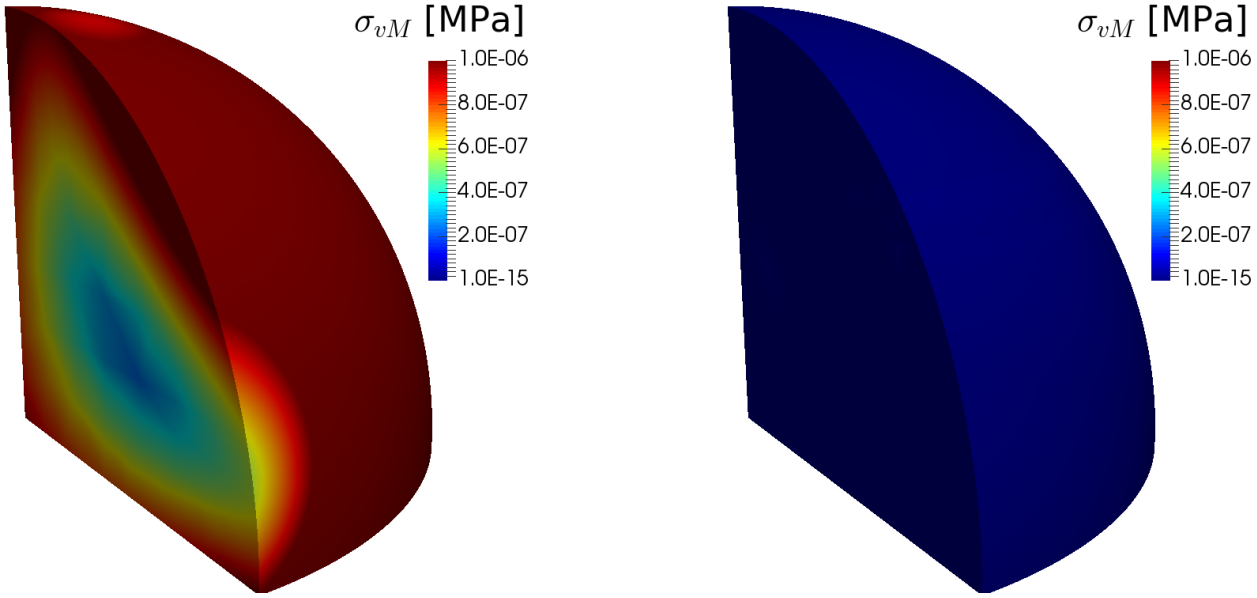
Next, we study the efficiency of the proposed integration methods by considering the error in energy norm and the von Mises stress σ_{vM} as the major quantities. Since the sphere experiences a pure hydrostatic stress state, the stress deviator vanishes – and, thus, the von Mises stress does as well. Fig. 18 shows the contour plots of the von Mises stress for the adaptive integration, the smart octree approach, and for the moment fitting based on a triangulated surface mesh and on an octree mesh for the integration of the moments. For the smart octree approach, the order of surface interpolation was chosen as $p_{\mathrm{B}} = 4$, and $(p+5)^3$ integration points were distributed in each integration cell. Here, it can be seen that the moment fitting using a surface mesh and the smart octree result in a higher accuracy than the adaptive integration and the moment fitting based on an octree integration of the moments. Using the same discretization for the integration of the stiffness matrix as for the integration of the load vector – as it is the case for the moment fitting using the triangulated surface mesh and the smart octree – results in a higher accuracy. This fact is demonstrated min more detail by considering the error in the von Mises stress. To this end, the relative error in the von Mises stress is defined as

$$e_{\mathrm{vM}} = \left| \frac{\sigma_{\mathrm{vM}}}{\bar{t}_n} \right| \times 100\,[\%] \quad . \tag{22}$$

For the investigation, we plot the error in the von Mises stress along a radial line $x = y = z \in [0, R/\sqrt{3}]$ as depicted in Fig. 19 – where APD, GLP, and OP are the results of the moment fitting using a triangulated surface mesh and $\mathrm{GLP}^{\mathrm{OT}}$ denotes the results of the moment fitting utilizing an octree mesh for the computation of the moments. It can be seen that the results of the moment fitting based on a surface mesh are almost identical with the smart octree, whereas the results of the octree are significantly higher. The results of the adaptive integration coincide with the results of the moment fitting if the same octree mesh is used for the computation of the moments. Since we have almost the same error level in the numerical integration, this difference originates from the different discretization of the mesh used for the computation of the stiffness matrix and the mesh used for the computation of the load vector, as mentioned before. Consequently, a nice feature of the moment fitting is that the same discretization can be applied for the computation of the load and the stiffness matrix. This feature also holds for the smart octree approach, because the resulting integration cells are boundary conforming. Therefore, those faces of the integration cells that lie on the interface can be directly used for the application of the boundary conditions.

In the following, we consider the relative error in energy norm as an additional quantity which is defined as

$$\|e\|_{E(\Omega)} = \sqrt{\left| \frac{\mathscr{U}_{\mathrm{ex}} - \mathscr{U}_{\mathrm{FCM}}}{\mathscr{U}_{\mathrm{ref}}} \right|} \times 100\,\% \quad . \tag{23}$$



(a) (b)

Fig. 18: The von Mises stress for the sphere under hydrostatic stress state. **a** Adaptive integration based on 8 octree refinements and moment fitting using the same adaptive integration scheme for the computation of the moments. **b** Smart octree and moment fitting for the triangulated surface mesh using APD, GLP, and OP

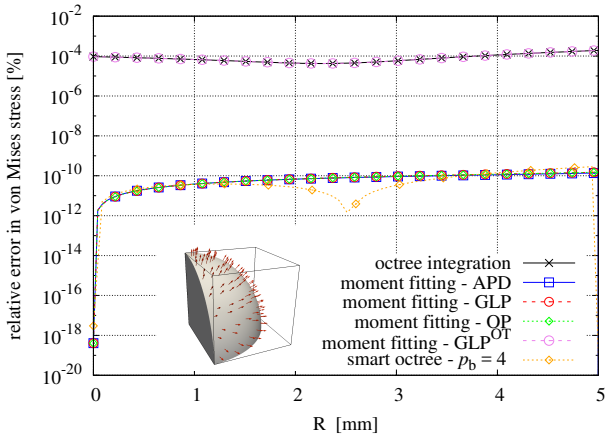


Fig. 19: Relative error in the von Mises stress along the diagonal cutline

In Eq. (23), $\mathscr{U}_{\text{ex}}$ is the analytical value of the strain energy which is defined as

$$\mathscr{U}_{\text{ex}} = \frac{1}{2}\int 3C\bar{t}_n \,\mathrm{d}\Omega = \frac{25}{2}\pi \text{ [J]} \approx 39.2699081698724 \text{ [J]} \tag{24}$$

and $\mathscr{U}_{\text{FCM}}$ is the strain energy of the FCM analysis. $\mathscr{U}_{\text{FCM}}$ is computed as

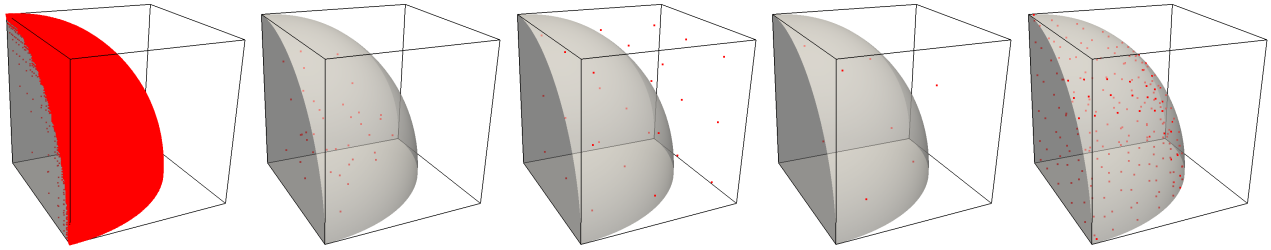
$$\mathscr{U}_{\text{FCM}} = \frac{1}{2}\mathbf{U}^{\text{T}}\mathbf{K}\mathbf{U} \quad , \tag{25}$$

where $\mathbf{U}$ is the displacement vector and $\mathbf{K}$ denotes the stiffness matrix. The values of the relative error in the energy norm of the different integration methods are listed in Tab. 2 – where the superscript "OT" denotes the octree, "SOT" the smart octree, "APD" the moment fitting using the adaptive point distribution, "GLP" the moment fitting utilizing the position of the Gauss-Legendre points, "GLP$^{\text{OT}}$" the moment fitting with position of the Gauss points applying an octree mesh for the computation of the moments, and "OP" the moment fitting with optimized position of the points and the weights. Here, it can be seen that the relative error is almost the same – with a value of about 0.2 [%] – for the moment fitting and the adaptive integration based on an octree. This is due to the fact that these integration methods have approximately the same error level in the volume integration. The lower value of the smart octree is due to the higher accuracy of the geometry approximation.

Table 2: Relative error in energy norm given in percentage

p	$\|e\|^{\text{OT}}_{E(\Omega)}$, $\|e\|^{\text{GLP}^{\text{OT}}}_{E(\Omega)}$	$\|e\|^{\text{APD}}_{E(\Omega)}$, $\|e\|^{\text{GLP}}_{E(\Omega)}$, $\|e\|^{\text{OP}}_{E(\Omega)}$	$\|e\|^{\text{SOT}}_{E(\Omega)}$
1	0.19 [%]	0.2 [%]	0.07 [%]

Considering the number of integration points, n_{g} emphasizes the main advantage of the presented integration methods in comparison to the standard adaptive integration based



(a) (b) (c) (d) (e)

Fig. 20: Position of the integration points for the sphere under hydrostatic stress state. **a** Adaptive integration based on 8 octree refinements. **b** Moment fitting using APD. **c** Moment fitting using GLP. **d** Moment fitting using OP. **e** Adaptive integration based on a smart octree using $p_{\mathrm{B}} = 4$ for the surface interpolation

on an octree. To this end, the total number of integration points of all methods are listed in Tab. 3. As it can be inferred from the table, the moment fitting reduces the number of integration points by a factor of about 10^5. Moreover, utilizing optimized points for the moment fitting reduces the number of integration points by a factor of about 3. The points of the different integration methods are illustrated in Fig. 20 with respect to the geometry of the cell and the topology of the physical domain. Here, it can be seen that the points of the adaptive integration and the moment fitting applying the APD are located within the physical domain of the cell. For the moment fitting utilizing the GLP and OP, however, the points are located within the physical and within the fictitious domain of the broken cell.

Table 3: Total number of integration points

p	$n_{\mathrm{g}}^{\mathrm{OT}}$	$n_{\mathrm{g}}^{\mathrm{APD}}, n_{\mathrm{g}}^{\mathrm{GLP}}, n_{\mathrm{g}}^{\mathrm{GLP^{OT}}}$	$n_{\mathrm{g}}^{\mathrm{OP}}$	$n_{\mathrm{g}}^{\mathrm{SOT}}$
1	2,061,424	27	8	1728

3.3 Porous domain under pressure

As the final example of this section, we investigate a problem of a porous material. In doing so, we consider a cube containing 27 ellipsoidal holes. The dimensions of the ellipsoids and their center are chosen randomly. The dimensions of the cube are given as $10 \times 10 \times 10\,\mathrm{mm}^3$. Fig. 21a shows the geometry and the boundary conditions of the problem under investigation. As it can be seen from the figure, the domain is subjected to symmetric boundary conditions – the cube's face at the back is fixed in x-direction, its face at the left-hand side in y-direction, and its face at the bottom in z-direction. A uniform pressure of 100 MPa is applied to the top face. The material behavior of the porous domain is assumed to be isotropic linear elastic where the Young's modulus E is 5.0 GPa and Poisson's ratio ν equals 0.3. For the FCM analysis, the domain is discretized with a Cartesian grid of $8 \times 8 \times 8$ cells. This results in a total number of 512 cells, of which 175 are broken.

For the investigation, we compare the moment fitting to the adaptive integration based on an octree and a smart octree subdivision of the broken cells. In doing so, we define an error level of about 10^{-5} in the volume integration for all integration methods under consideration. In case of the moment fitting – where we compute the moments via a surface integration – we generate a triangular surface mesh for the whole geometry, for which the error in the geometry description can be easily defined by any CAD program. Since the mesh, however, represents the whole geometry, boolean operations have to be performed between the broken cells and the geometry. To this end, we use the Cork Boolean/CSG library [67] and perform intersections to obtain the individual surface meshes for the physical domain of the broken cells. The intersection between a broken cell and the geometry is demonstrated in Fig. 21c, just to give an example. To obtain the same error in integration, applying the adaptive integration based on an octree subdivision is not exactly straightforward since the accuracy depends on the number of refinements and on the number of integration points of every sub-cell – which depends on the order of the quadrature rule of the sub-cell. The order of the quadrature rule p_{q}, on the other hand, depends on the order of the Ansatz p ($p_{\mathrm{q}} = 2p$). For this reason, we computed the error in the volume integration applying different orders of the quadrature rule ($p_{\mathrm{q}} = 2, 4, 8$) for 1,2,3, and 4 refinements of the octree. The results are plotted in Fig. 22. From the figure, it can be inferred that 3 octree refinements capture the error of about 10^{-5} in the volume integration for $p_{\mathrm{q}} = 4, 8$. For $p_{\mathrm{q}} = 2$, however, the error in volume integration is about 10^{-4}, even for 4 octree refinements. Due to this reason, we choose 4 octree refinements for adaptive integration, to be on the save side. The inside of the corresponding sub-cell mesh using 4 octree refinements is exemplarily depicted in Fig. 21d. The integration mesh provided by the smart octree decomposition employs a high-order reparametrization of the in-

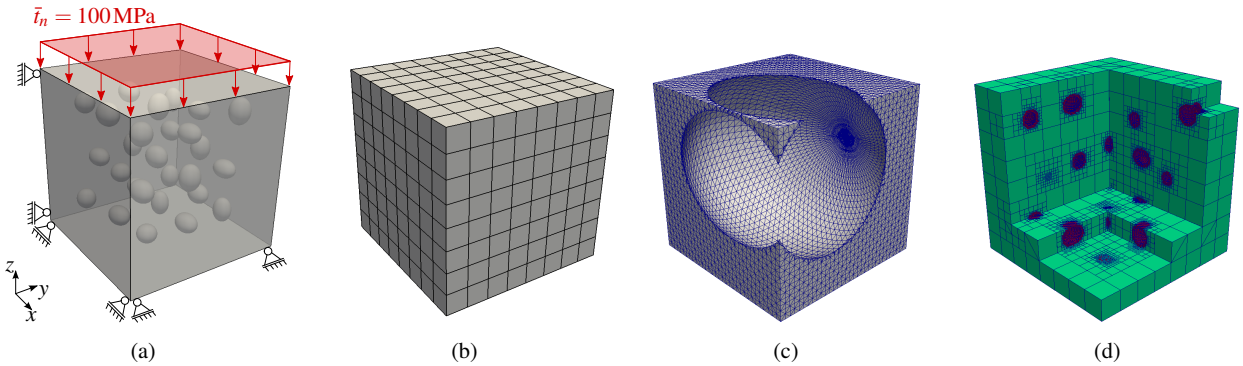


Fig. 21: Porous domain containing 27 ellipsoidal holes. **a** The FCM model. **b** Cartesian grid with $8 \times 8 \times 8$. **c** Triangular surface mesh of one broken cell used for the moment fitting. **d** Octree mesh with 4 refinements.

terface, with $p_\mathrm{B} = 4$ for the polynomial order of the surface interpolation. Thus, the high-order mapping may have to be taken into account by the order of the quadrature rule p_q. To investigate the effects of p_q on the accuracy of the FCM computations, we tested two scenarios, with $p_\mathrm{q} = p+1$ and $p_\mathrm{q} = p+4$.

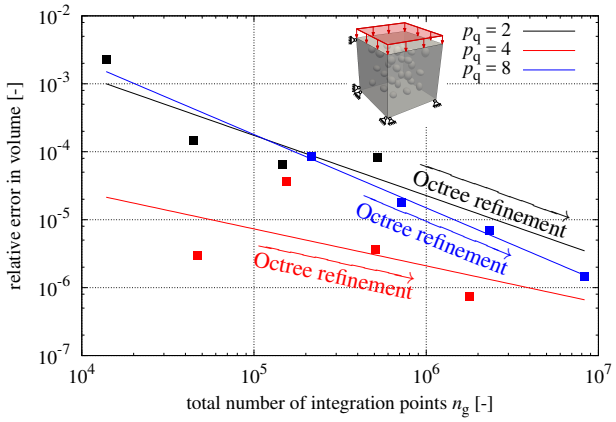


Fig. 22: Relative error in volume applying different orders p_q for the quadrature rule of the sub-cells

As a measure for the accuracy, we again consider the relative error in energy norm, which is defined in Eq. (23) and replace $\mathscr{U}_\mathrm{ex}$ by $\mathscr{U}_\mathrm{ref} = 1.065820653\,\mathrm{J}$ which is a reference solution obtained by an overkill FEM analysis. To study the convergence behavior applying the moment fitting and the adaptive octree integration, the relative error in energy norm is computed for different orders of the Ansatz p. In Fig. 23, the relative error in energy norm is plotted against the number of degrees of freedom in a double logarithmic diagram. As it can be seen from the figure, the convergence behavior is almost the same. As mentioned before, this is due to the usage of approximately the same error level for all integration methods. However, the quadrature rules based on the optimization of the position of the points and the weights are only studied for $p = 1, \ldots, 4$ since its setup becomes computationally more and more expensive for higher orders of the Ansatz. The two error curves corresponding to the smart octree method show that the error in energy norm is independent of the order of the quadrature rule employed. The reason for this behavior is that the discretization error dominates the integration error, even for high values of p.

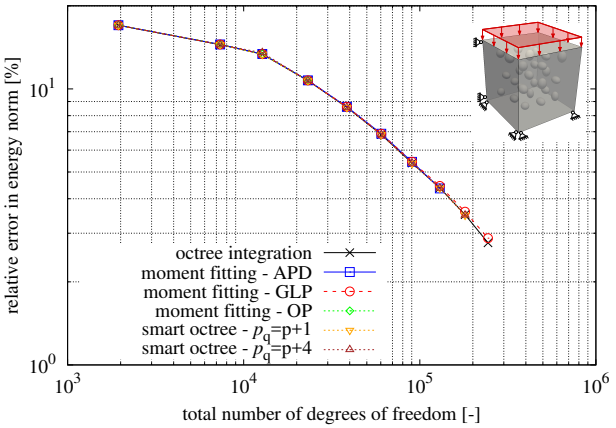


Fig. 23: Relative error in energy norm

Fig. 24 shows the total number of integration points n_g. The figure reveals that we can reduce the number of integration points significantly using the moment fitting and, thus, obtain more efficient quadrature rules as compared to the standard integration in the FCM based on an octree subdivision. Applying the optimization of points and weights, we obtain even more efficient quadrature rules, although the gain is not that high anymore.

Tab. 4 lists the total number of the points of the individual integration methods. Here, p denotes the order of the FCM analysis, n_g^{OT} and $n_g^{\mathrm{SOT[x]}}$ the number of points of the octree- and smart octree subdivision (with x denot-

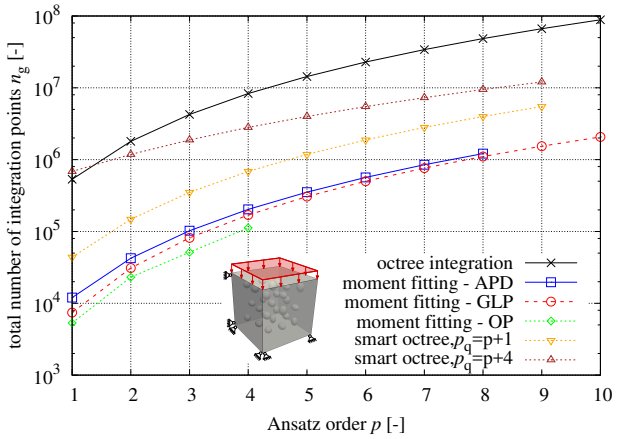


Fig. 24: Total number of integration points

ing the quadrature rule corresponding to $(p+x)^3$ quadrature points), n_g^{GLP} the number of points of the moment fitting using the position of the standard Gauss-Legendre points, and n_g^{OP} the number of points applying the moment fitting with optimized points and weights. As it can be inferred from the table, the moment fitting reduces the number of integration points by a factor of about 71 and a factor of about 42 for the lowest ($p=1$) and highest order ($p=10$) of the Ansatz, respectively. The factors for the other orders results in values between 71 and 42. Optimizing the points and weights, we can reduce the number of integration points by factors of about 100, 77, 82, and 74 as compared to the octree integration.

The number of integration points provided by the smart octree approach lies between standard octrees and the moment fitting approach. For the quadrature order $p_{\mathrm{q}} = p+4$ and low polynomial ansatz p, the amount of integration points is approximately the same for smart octrees and standard octrees. However, the cost of numerical integration is lower for smart octrees in these cases as well. Because the method separates the broken cells into integration cells that lie either completely inside or completely outside the domain, the inside-outside state of individual integration points does not need to be evaluated in the integration loop anymore. Instead, the corresponding state can be preassigned, according to the state of the integration cell where the integration points originate from. In contrast, for the octree method, there is no clear separation of the integration cells, which is why they state of the points needs to be evaluated on the leaves, making the integration loop more expensive.

To illustrate this point, we compare the computation of the stiffness matrix at $p=1$, using the standard octree with 4 levels of refinement and $(p+1)^3$ points for the smart octree with a surface representation of $p_{\mathrm{B}}=4$. All computations lead to an error in the energy norm of approximately 17 per cent. However, the smart octree is two times faster if $(p+4)^3$ integration points are used per cell, and three times faster for $(p+1)^3$.

The reduction of the number of integration points using the moment fitting, however, comes at the expense of the setup of the quadrature rule. To this end, in the following, we consider the computational time for the setup of the quadrature t_q and the stiffness matrix computation t_K applying the moment fitting, and compare it to the results of the adaptive integration based on an octree – which is commonly used in the context of the FCM. In doing so, the computations are performed in parallel on a computer featuring two Intel Xeon E5-2640 v4 processors with 128GB RAM – each of them has 10 cores. The results are listed in Tab. 5 where t_{OT} denotes the computational time of the adaptive integration based on an octree, t_{APD} the moment fitting using the APD, t_{GLP} the moment fitting applying the GLP, and $t_{GLP^{OT}}$ the moment fitting using GLP where the right-hand side in Eq. (4) is computed utilizing an octree – where the same octree is applied as for the adaptive integration. From the results it can be seen that the overhead for setting up the moment fitting quadratures using a triangulated surface description (t_{APD} and t_{GLP}) is much higher than the overhead for the adaptive integration based on an octree (t_{OT}). However, the overhead of the moment fitting could be reduced significantly by applying an octree integration for the computation of the integrals of the moment fitting equations ($t_{GLP^{OT}}$). In doing so, the set up of the quadrature is about 14 times faster. This is due to the fact that the number of integration points applying the octree ($n_g^{OT} = 22,820,133$) is less than the number of integration points which are used for the surface integration on the triangles ($n_g^{TR} = 3,318,188,460$) – where the superscript TR denotes the triangles.

The computational time required for the stiffness matrix evaluation can be reduced significantly when applying the moment fitting method. This means in the case of nonlinear analyses where the stiffness matrix has to be recomputed several times during the Newton-Raphson iterations at different load steps, the overhead of the moment fitting will be amortized. To emphasize this point in Fig. 25 and 26 the computational time is plotted versus the number of the stiffness matrix computations. In doing so, the computational time is estimated as

$$t = t_q + n_K t_K \tag{26}$$

where n_K is the number of the stiffness matrix computations. From Fig. 25, where a triangulated surface is used for the moment fitting, it can be seen that the moment fitting overtakes the octree integration after $20,\ldots,25$ stiffness matrix computations. Assuming 5 iterations per load step, this means that the moment fitting is faster if more than 4 or 5 load steps have to be performed during the analysis. Further, Fig. 26 shows that the efficiency of the moment fitting could be improved combining the octree with the GLP. Here, the

Table 4: Total number of integration points

p	n_g^{OT}	n_g^{APD}	n_g^{GLP}	n_g^{OP}	n_g^{SOT1}	n_g^{SOT4}
1	532248	12002	7421	5321	43856	685250
2	1796337	42358	30974	23099	148014	1184112
3	4257984	102322	81593	51318	350848	1880326
4	8316375	203120	169700	112125	685250	2806784
5	14370696	353062	305717		1184112	3996378
6	22820133	565141	500066		1880326	5482000
7	34063872	849355	763169		2806784	7296542
8	48501099	1213392	1105448		3996378	9472896
9	66531000		1537325		5482000	12043954
10	88552761		2069222			

Table 5: Computational costs of the moment fitting and the adaptive integration based on an octree for 512 cells of polynomial degree $p = 6$

	t_{OT}	t_{APD}	t_{GLP}	$t_{GLP^{OT}}$
t_q [s]	≈ 25.33	≈ 2992.53	≈ 2542.2	≈ 187.44
t_K [s]	≈ 122.26	≈ 2.85	≈ 2.54	≈ 2.54

moment fitting is faster than the octree integration after one stiffness matrix computation.

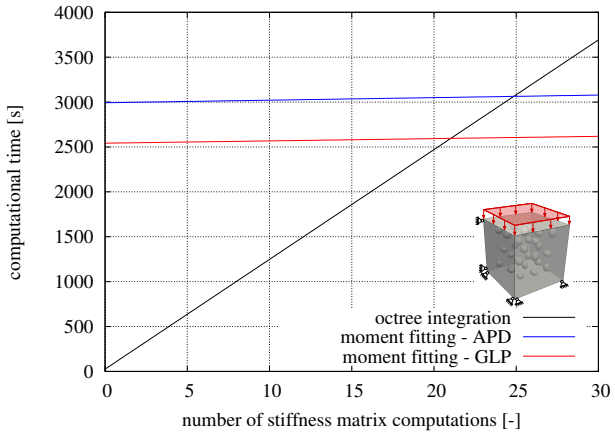


Fig. 25: Estimated computational time for stiffness matrix computations with $p = 6$ in a nonlinear analysis

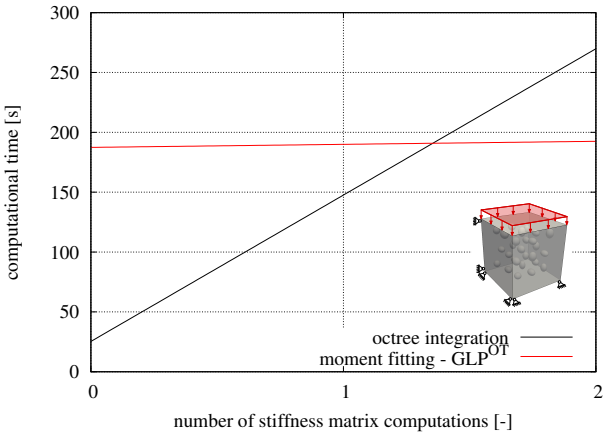


Fig. 26: Estimated computational time for stiffness matrix computations with $p = 6$ in a nonlinear analysis

4 Summary and outlook

In this paper, we presented two approaches to perform the numerical integration of discontinuous functions. The approaches can be applied in any non-standard discretization method where elements or cells are intersected arbitrarily by the physical boundary or internal interfaces. To this end, we proposed an integration scheme based on the moment fitting in our first approach. Thereby, we showed that using the position of the standard Gauss-Legendre points transforms the nonlinear moment fitting equations into a linear square system with a good conditioning behavior. Thanks to the good conditioning, the system can be solved accurately – resulting in an error close to zero within machine precision. Thus, we are able to obtain highly accurate quadrature rules for arbitrary domains, even for the high-order quadrature rules. In addition to this, we extended the moment fitting by formulating an optimization problem whose solution provides an approximation to the solution of the nonlinear moment fitting equations. In this optimization problem, we minimized

the norm of the residual of the moment fitting system. We demonstrated that the solution of the developed optimization problem results in quadrature rules with fewer points and weights. Consequently, the optimization leads to quadrature rules that are more efficient with respect to the number of quadrature points.

By applying the integration methods based on moment fitting to linear problems of the FCM, we could show that the position of the integration points is not necessarily restricted to the physical domain, so that it is also possible to have points within the fictitious domain of the broken cell. It appeared that the presented moment fitting methods were suitable to reduce the number of integration points significantly as compared to the adaptive integration based on an octree subdivision. Thus, the computation of quantities such as the stiffness matrix can be performed more efficiently. Moreover, the moment fitting yields the same results as the adaptive integration if the integrals of the moment fitting equations are computed using the same octree mesh for the integration.

In the second and final approach, we presented a composed integration scheme based on the smart octree. In this method, the broken cells are subdivided into high-order subcells, similar to the standard octree. To obtain a high accuracy for the representation of the integration domain, the smart octree provides high-order interpolation functions and a node-relocation algorithm. Due to the high-order sub-cells, we were able to show that the smart octree requires much less sub-cells – at the same time resulting in a more accurate approximation of the integration domain than the standard octree. We showed that the number of integration points could be reduced as compared to the standard adaptive integration based on an octree subdivision. Although the number of integration points turned out higher than for the moment fitting, we emphasize, that in the case of the smart octree, all points are located within the physical domain – which might be of particular importance for nonlinear computations. Further, the smart octree does not need to solve an equation system as it is the case for the moment fitting. Moreover, since the smart octree results in a higher accuracy than the standard octree, it can be used for to compute the integrals of the moment fitting equations.

In future work, we will focus on applying the proposed integration methods to nonlinear problems of the FCM. In doing so, we will investigate the influence of the location of the integration points in the moment fitting method to find out whether they should be restricted to the physical domain or whether they can also be located in the fictitious domain. Further, we want to combine the moment fitting with the smart octree and similar integration methods for the computation of the moments in order to increase the accuracy by simultaneously reducing the number of integration points. Moreover, we will improve the optimization problem of the nonlinear moment fitting equations for high-order quadrature rules to perform the optimization more efficiently.

Acknowledgements The authors gratefully acknowledge support by the Deutsche Forschungsgemeinschaft in the Priority Program 1748 "Reliable simulation techniques in solid mechanics. Development of non-standard discretization methods, mechanical and mathematical analysis" under the project DU 405/8-1, SCHR 1244/4-1, and RA 624/27-1.

References

1. T. Belytschko and T. Black. Elastic crack growth in finite elements with minimal remeshing. *International Journal for Numerical Methods in Engineering*, 45:601–620, 1999.
2. N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *International Journal for Numerical Methods in Engineering*, 64:131–150, 1999.
3. N. Sukumar, N. Moës, B. Moran, and T. Belytschko. Extended finite element method for three-dimensional crack modelling. *International Journal for Numerical Methods in Engineering*, 48:1549–1570, 2000.
4. N. Sukumar, D.L. Chopp, N. Moës, and T. Belytschko. Modeling holes and inclusions by level sets in the extended finite-element method. *Computer Methods in Applied Mechanics and Engineering*, 190:6183–6200, 2001.
5. T. Strouboulis, I. Babuška, and K. Copps. The design and analysis of the generalized Finite Element Method. *Computer Methods in Applied Mechanics and Engineering*, 181:43–69, 2000.
6. C. A. Duarte, I. Babuška, and J. T. Oden. Generalized finite element method for three-dimensional structural mechanics problems. *Computers & Structures*, 77(2):215–232, 2000.
7. T. Strouboulis, K. Copps, and I. Babuška. The generalized finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190:4081–4193, 2001.
8. J.M. Melenk and I. Babuška. The partition of unity finite element method: basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139:289–314, 1996.
9. I. Babuška and J.M. Melenk. The partition of unity method. *International Journal for Numerical Methods in Engineering*, 40:727–758, 1997.
10. V. K. Saul'ev. On solution of some boundary value problems on high performance computers by fictitious domain method. *Siberian Mathematical Journal*, 4:912–925, 1963.
11. R. Glowinski and Y. Kuznetsov. Distributed Lagrange multipliers based on fictitious domain method for second order elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 196:1498–1506, 2007.
12. I. Ramière, P. Angot, and M. Belliard. A fictitious domain approach with spread interface for elliptic problems with general boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 196:766–781, 2007.
13. I. Ramière, P. Angot, and M. Belliard. A general fictitious domain method with immersed jumps and multilevel nested structured meshes. *Journal of Computational Physics*, 225:1347–1387, 2007.
14. J. Parvizian, A. Düster, and E. Rank. Finite cell method – h- and p-extension for embedded domain problems in solid mechanics. *Computational Mechanics*, 41:121–133, 2007.
15. A. Düster, J. Parvizian, Z. Yang, and E. Rank. The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 197:3768–3782, 2008.

16. M. Dauge, A. Düster, and E. Rank. Theoretical and numerical investigation of the finite cell method. *Journal of Scientific Computing*, 65:1039–1064, 2015.
17. 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.
18. M. Ruess, D. Schillinger, Y. Bazilevs, V. Varduhn, and E. Rank. Weakly enforced essential boundary conditions for NURBS-embedded and trimmed NURBS geometries on the basis of the finite cell method. *International Journal for Numerical Methods in Engineering*, 95(10):811–846, 2013.
19. M. Ruess, D. Schillinger, A. Özcan, and E. Rank. Weak coupling for isogeometric analysis of non-matching and trimmed multi-patch geometries. *Computer Methods in Applied Mechanics and Engineering*, pages 46–71, 2014.
20. S. Kollmannsberger, A. Özcan, J. Baiges, M. Ruess, E. Rank, and A. Reali. Parameter-free, weak imposition of Dirichlet boundary conditions and coupling of trimmed and non-conforming patches. *International Journal for Numerical Methods in Engineering*, 101(9):1–30, 2014.
21. E. Béchet, H. Minnebo, N. Moës, and B. Burgardt. Improved implementation and robustness study of the X-FEM for stress analysis around cracks. *International Journal for Numerical Methods in Engineering*, 64:1033–1056, 2005.
22. C. Lang, D. Makhija, A. Doostan, and K. Maute. A simple and efficient preconditioning scheme for heaviside enriched XFEM. *Computational Mechanics*, 54:1357–1374, 2014.
23. S. Heinze, M. Joulaian, H. Egger, and A. Düster. Efficient computation of cellular materials using the finite cell method. *Proceedings in Applied Mathematics and Mechanics*, 14:251–252, 2014.
24. S. Loehnert. Stabilizing the xfem for static and dynamic crack simulations. *Proceedings in Applied Mathematics and Mechanics*, 15:137–138, 2015.
25. F. de Prenter, C. V. Verhoosel, G. J. van Zwieten, and E. H. van Brummelen. Condition number analysis and preconditioning of the finite cell method. *ArXiv e-prints*, 2016.
26. D. Schillinger and E. Rank. An unfitted *hp* adaptive finite element method based on hierarchical B-splines for interface problems of complex geometry. *Computer Methods in Applied Mechanics and Engineering*, 200:3358–3380, 2011.
27. D. Schillinger, A. Düster, and E. Rank. The *hp-d*-adaptive finite cell method for geometrically nonlinear problems of solid mechanics. *International Journal for Numerical Methods in Engineering*, 89:1171–1202, 2012.
28. M. Joulaian and A. Düster. Local enrichment of the finite cell method for problems with material interfaces. *Computational Mechanics*, 52:741–762, 2013.
29. N. Zander, T. Bog, S. Kollmannsberger, D. Schillinger, and E. Rank. Multi-Level hp-Adaptivity: High-Order mesh adaptivity without the difficulties of constraining hanging nodes. *Computational Mechanics*, 55(3):499–517, 2015.
30. Nils Zander, Tino Bog, Mohamed Elhaddad, Felix Frischmann, Stefan Kollmannsberger, and Ernst Rank. The multi-level hp-method for three-dimensional problems: Dynamically changing high-order mesh refinement with arbitrary hanging nodes. *Computer Methods in Applied Mechanics and Engineering*, 310:252–277, Oktober 2016.
31. Z. Yang, M. Ruess, S. Kollmannsberger, A. Düster, and E. Rank. An efficient integration technique for the voxel-based Finite Cell Method. *International Journal for Numerical Methods in Engineering*, 91(5):457–471, 2012.
32. B. Müller, F. Kummer, M. Oberlack, and Y. Wang. Simple multidimensional integration of discontinuous functions with application to level set methods. *International Journal for Numerical Methods in Engineering*, 92:637–651, 2012.
33. Han Chen, Chohong Min, and Frédéric Gibou. A numerical scheme for the Stefan problem on adaptive Cartesian grids with supralinear convergence rate. *Journal of Computational Physics*, 228(16):5803–5818, 2009.
34. A. Abedian, J. Parvizian, A. Düster, and E. Rank. Finite cell method compared to *h*-version finite element method for elasto-plastic problems. *Applied Mathematics and Mechanics*, 35(10):1239–1248, 2014.
35. G. Ventura. On the elimination of quadrature subcells for discontinuous functions in the eXtended Finite-Element Method. *International Journal for Numerical Methods in Engineering*, 66:761–795, 2006.
36. G. Ventura and E. Benvenuti. Equivalent polynomials for quadrature in Heaviside function enrichment elements. *International Journal for Numerical Methods in Engineering*, 102:688–710, 2015.
37. A. Abedian, J. Parvizian, A. Düster, H. Khademyzadeh, and E. Rank. Performance of different integration schemes in facing discontinuities in the finite cell method. *International Journal of Computational Methods*, 10(3):1350002/1–24, 2013.
38. K.W. Cheng and T.-P. Fries. Higher-order XFEM for curved strong and weak discontinuities. *International Journal for Numerical Methods in Engineering*, 82:564–590, 2009.
39. A. Abedian, J. Parvizian, A. Düster, and E. Rank. The finite cell method for the J_2 flow theory of plasticity. *Finite Elements in Analysis and Design*, 69:37–47, 2013.
40. A. Abedian and Düster. An extension of the finite cell method using boolean operations. *Computational Mechanics*, n/a(n/a):n/a–n/a, 2017.
41. F. L. Stazi, E. Budyn, J. Chessa, and T. Belytschko. An extended finite element method with higher-order elements for curved cracks. *Computational Mechanics*, 31(1–2):38–48, 2003.
42. K. Dréau, N. Chevaugeon, and N. Moës. Studied X-FEM enrichment to handle material interfaces with higher order finite element. *Computer Methods in Applied Mechanics and Engineering*, 199(29-32):1922–1936, 2010.
43. M. Moumnassi, J. Belouettar, É Béchet, S.P.A. Bordas, Quoirin D., and M. Potier-Ferry. Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces. *Computer Methods in Applied Mechanics and Engineering*, 200:5–8, 2011.
44. László Kudela, Nils Zander, Tino Bog, Stefan Kollmannsberger, and Ernst Rank. Efficient and accurate numerical quadrature for immersed boundary methods. *Advanced Modeling and Simulation in Engineering Sciences*, 2(1):1–22, Juni 2015.
45. László Kudela, Nils Zander, Stefan Kollmannsberger, and Ernst Rank. Smart octrees: Accurately integrating discontinuous functions in 3D. *Computer Methods in Applied Mechanics and Engineering*, 306:406–426, Juli 2016.
46. N. Moës, M. Cloirec, P. Cartraud, and J.-F. Remacle. A computational approach to handle complex microstructure geometries. *Computer Methods in Applied Mechanics and Engineering*, 192:3163–3177, 2003.
47. S. Loehnert, D. S. Mueller-Hoeppe, and P. Wriggers. 3D corrected XFEM approach and extension to finite deformation theory. *International Journal for Numerical Methods in Engineering*, 86:431–452, 2011.
48. K. Höllig. *Finite Element Methods with B-Splines*. Frontiers in Applied Mathematics. SIAM Society for Industrial and Applied Mathematics, 2003.
49. K. Höllig and J. Hörner. Programming finite element methods with weighted B-splines. *Computers & Mathematics with Applications*, 70(7):1441–1456, 2015.
50. T.-P. Fries, S. Omerović, D. Schöllhammer, and J. Steidl. Higher-order meshing of implicit geometries–Part I: Integration and interpolation in cut elements. *Computer Methods in Applied Mechan-*

ics and Engineering, 313:759–784, 2017.
51. M. Joulaian, S. Hubrich, and A. Düster. Numerical integration of discontinuities on arbitrary domains based on moment fitting. *Computational Mechanics*, 57:979–999, 2016.
52. 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.
53. 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.
54. 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*, 96:512–528, 2013.
55. V. Thiagarajan and V. Shapiro. Adaptively weighted numerical integration over arbitrary domains. *Computers & Mathematics with Applications*, 67(9):1682–1702, 2014.
56. S. Hubrich, M. Joulaian, and A. Düster. Numerical integration in the finite cell method based on moment-fitting. In *Proceedings of 3rd ECCOMAS Young Investigators Conference, 6th GACM Colloquium*, pages 1–4, Aachen, Germany, 2015.
57. S. Hubrich, M. Joulaian, P. Di Stolfo, A. Schröder, and A. Düster. Efficient numerical integration of arbitrarily broken cells using the moment fitting approach. *Proceedings in Applied Mathematics and Mechanics*, 16:201–202, 2016.
58. E. K. Ryu and S. P. Boyd. Extensions of gauss quadrature via linear programming. *Found. Comput. Math.*, 15(4):953–971, 2015.
59. 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.
60. M. Joulaian, N. Zander, T. Bog, S. Kollmannsberger, E. Rank, and A. Düster. A high-order enrichment strategy for the finite cell method. *Proceedings in Applied Mathematics and Mechanics*, 15:207–208, 2015.
61. 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.
62. W.J. Gordon and Ch.A. Hall. Transfinite element methods: Blending function interpolation over arbitrary curved element domains. *Numerische Mathematik*, 21:109–129, 1973.
63. https://www.wolfram.com/mathematica/.
64. http://www.netlib.org/lapack/.
65. A.R. Krommer and C.W. Ueberhuber. *Numerical integration on advanced computer systems*. Springer, 1994.
66. H. Bröker. *Integration von geometrischer Modellierung und Berechnung nach der p-Version der FEM*. PhD thesis, Lehrstuhl für Bauinformatik, Fakultät für Bauingenieur- und Vermessungswesen, Technische Universität München, 2001.
67. https://github.com/gilbo/cork/.