Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 401 (2022) 115565
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Octree-based integration scheme with merged sub-cells for the finite cell method: Application to non-linear problems in 3D

Márton Petö[a,*], Wadhah Garhuom[b], Fabian Duvigneau[a], Sascha Eisenträger[c], Alexander Düster[b], Daniel Juhre[a]

[a] *Institute of Mechanics, Otto von Guericke University, Magdeburg, Germany*
[b] *Hamburg University of Technology, Germany*
[c] *Technical University of Darmstadt, Germany*

Received 2 May 2022; received in revised form 12 August 2022; accepted 14 August 2022
Available online xxxx

Abstract

Fictitious domain methods, such as the Finite Cell Method (FCM), allow for an efficient and accurate simulation of complex geometries by utilizing higher-order shape functions and an unfitted discretization based on rectangular elements. Since the mesh does not conform to the geometry, cut elements arise that are intersected by domain boundaries. For optimal convergence rates and the efficiency of the simulation in general, special integration schemes have to be used in such elements. In this contribution, the often used, robust octree-decomposition-based integration scheme is enhanced by a novel approach reducing the computational effort when evaluating the discontinuous integrals. This is realized by introducing an additional step, in which the local integration mesh is simplified using data compression techniques leading to fewer integration domains/points. An important advantage of the proposed method is that it can be added in a modular fashion to already existing codes. While it inherits all desired properties of the octree-decomposition-based integration scheme, it significantly reduces the number of integration points and has hardly any negative effect on the simulation accuracy. In this paper, the proposed integration scheme is introduced in detail, and investigated by means of numerical examples in the context of 3D non-linear problems.

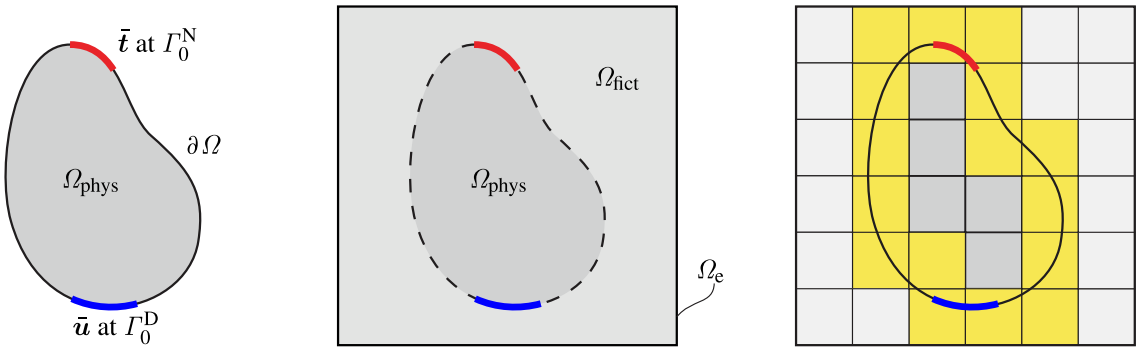
Keywords: Finite cell method; Embedded domain methods; Octree integration; Discontinuous integrals; Non-linear mechanics

1. Introduction

The finite cell method (FCM), being a combination of high-order shape functions [1,2] and the fictitious domain approach [3], enables an efficient and accurate simulation of geometrically complex structures [4,5]. Generally, the investigated physical domain Ω_{phys} (Fig. 1(a)) is embedded into a larger domain of simple shape Ω_{e} (Fig. 1(b)) to facilitate the spatial discretization by means of Cartesian meshes (Fig. 1(c)). Appropriate Dirichlet (blue) and Neumann boundary conditions (red) are defined on the physical boundary $\partial\Omega$. Consequently, the meshing procedure is detached from the geometric features of the investigated domain, and thus, the often cumbersome and error-prone discretization with geometry-conforming finite elements can be avoided. For extending the original problem

* Corresponding author.
E-mail address: marton.petoe@ovgu.de (M. Petö).

https://doi.org/10.1016/j.cma.2022.115565



(a) Original problem (b) Embedded problem (c) Discretization of Ω_e

Fig. 1. The key idea of the finite cell method. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

formulation from the physical domain Ω_{phys} to the embedding domain Ω_e, which also includes the fictitious domain Ω_{fict} of zero stiffness, the discontinuous indicator function

$$\alpha = \begin{cases} 1 & \text{in } \Omega_{phys} \\ 0 & \text{in } \Omega_{fict} \end{cases} \tag{1}$$

is defined. Provided that the embedded geometry is accurately taken into account during the integration phase, exponential convergence rates are possible [6]. For further reading on the key concepts of the FCM, we refer to Refs. [2,7].

Note that in the research of fictitious/embedded domain methods, several approaches have emerged, that are based on a similar idea. Such approaches are, e.g., the *cut finite element method* (CutFEM) [8–10], *extended/generalized FEM* (XFEM/GFEM) [11], *Cartesian grid FEM* (cgFEM) [12,13], *fixed grid FEM* (FGFEM) [14], unfitted FEM [15,16], and immersed boundary methods [17,18].

1.1. State of the art

Since its development in 2007, the FCM has been extended in various ways and applied to several problems, such as geometrical non-linearities [19–21], elasto-plasticity [22–24], foamed materials [25,26], explicit and implicit elastodynamics [27,28], thermoelasticity [29], biomechanics [30–32], piezoelectricity [33], homogenization [34,35], phase field fracture [36], topology optimization [37] and contact problems [38]. The newest applications include simulation of milling processes [39], geometries based on point cloud data structures [40], and functionally graded materials [41].

Although the FCM was originally formulated using high-order shape functions based on integrated Legendre polynomials, today, there are several implementations exploiting other Ansatz spaces: In the *Spectral Cell Method* (SCM), Lagrangian shape functions based on the Gauss–Legendre–Lobatto (GLL) nodal distribution are used [42–44]. These shape functions are especially suitable for capturing high-frequency waves in the displacement field and allow for the application of mass lumping techniques. Therefore, the SCM is typically used for wave propagation problems. Multimaterial problems, where cut cells can contain more than one physical material, require an efficient capturing of the material interfaces in the solution field. This can be achieved by local enrichment of the Ansatz space by additional shape functions that enable the C^0-continuity of the displacement field within the cut cells. This approach was adopted from the XFEM and successfully implemented and extended the FCM [45,46]. Other implementations are based on shape functions of higher differentiability across the cell boundaries using NURBS [47–49] and B-Splines [19,50]. Finally, rational shape functions over polygonal cells were also successfully implemented in the FCM [51,52].

Due to the unfitted mesh, different cell types arise that have to be treated differently: *Physical cells* are fully interior to Ω_{phys} (gray cells in Fig. 1(c)) and *fictitious cells* are completely located in Ω_{fict} (light gray cells in Fig. 1(c)). While the former ones can be treated as standard finite elements, the latter ones have zero stiffness and thus, these are removed from the simulation in the pre-processing phase. Finally, *cut cells* appear, which are intersected by $\partial\Omega$ and are partially physical and fictitious (yellow cells in Fig. 1(c)). In fact, cut cells pose some challenges in the FCM simulation pipeline regarding (i) the enforcement of boundary conditions, (ii) conditioning of the discrete problem and (iii) and integration of the cut cell matrices. While for more information on (i) and (ii) we refer to Refs. [5,11,19,47,48,53,54], respectively, the last issue constitutes the main focus of this article and will be discussed in the following.

1.2. Integration of discontinuous functions

A key requirement for an accurate FCM simulation is the proper integration of the cut cell matrices, which involves the computation of discontinuous integrals of the form $I = \int_{\Omega^c} \alpha F \, \mathrm{d}\boldsymbol{x}$. Since for such integrals, Gaussian quadrature yields a severely deteriorated integration accuracy, in the FCM, more suitable numerical integration schemes are required.

A main branch of the available approaches is based on the introduction of a local integration mesh (LIM), such that the piece-wise continuous integrand is integrated over several sub-domains with reasonable accuracy. In the remainder, the integration domains in the LIM will be referred to as (integration) sub-cells. The LIM-based integration can be realized using quadtree/octree-decomposition techniques (QTD/OTD) yielding an unfitted integration mesh with rectangular [5,22] or tetrahedral sub-cells [55,56], or by geometry conforming meshes using distorted sub-cells with linear [12,57], high-order [58,59] or blending mapping functions [60,61]. While geometry conforming meshes are known for higher integration accuracy and lower integration point count, they require explicit finding of the intersecting boundary. The QTD/OTD-based adaptive integration scheme (discussed in more detail in Section 2.3) is known for its robustness, however, it also results in a significant amount of integration points. Thus, in the recent years, further approaches have emerged to evaluate discontinuous integrals of embedded domain methods more efficiently.

Such an alternative integration scheme is seen in the recently emerging *Moment Fitting* (MF) approach, where in each cut cell, a unique quadrature rule is derived. The objective is to obtain a set of integration points, whose position and weights are such, that the discontinuous nature of the integrand is already taken into account during the numerical integration. Thus, a high integration accuracy with relatively low amount of integration points can be achieved. MF generally involves the solution of a non-linear equation system (EQS) [62,63]. A linear EQS can be obtained by pre-defining the position of the integration points either in the entire [64] or physical part of the integration domain [65–67]. Unfortunately, for non-linear problems, the standard MF is not suitable as it leads to a loss of stability of the Newton–Raphson iteration. A solution to this can be seen in the adaptive moment fitting approach, where the MF is applied individually in the cut sub-cells of a QTD/OTD [24]. Alternatively, a decomposition of each cut cell into additional material subdomains was introduced in Ref. [68], for dealing with more sophisticated material models such as multi-yield surface plasticity. Another method is to use non-negative moment fitting quadrature rules that proved well for 1D/2D small strain elasto-plastic problems [69]. Finally, an application of the MF to locally enriched material interfaces can be found in Ref. [70].

A different idea for integrating discontinuous functions over the cut cells is based on the *equivalent polynomials* [71,72] and *equivalent Legendre polynomials* [73] methods. These approaches are based on the replacement of the discontinuous integrand by a continuous one with same integral value and thus, similarly to the MF, they enable an integration over the entire cell without the need for space partitioning techniques.

Further methods are based on identifying the physical part of the cut cell $\Omega^c_{\text{phys}} \subset \Omega^c$ over which the integrand is continuous. Then, the integral over the n-dimensional Ω^c_{phys} can be computed by $n-1$ successive applications of the Divergence Theorem, leading to a set of one-dimensional integrals [74]. However, analytical computation of the required antiderivatives is often not possible or computationally not feasible. Instead, radial basis functions can be used for discrete evaluations of the antiderivatives [75,76], or a pre-computation of the integrand as it was investigated in the context of the FCM [77]. For a comparison of some of the methods mentioned above see the work by Abedian et al. [78]

1.3. Motivation

The objective of our contribution is the development of an easy-to-implement and efficient integration scheme for fictitious domain methods, which is also robustly usable for non-linear problems, where cut cell matrices have to be computed numerous times and stability of the Newton–Raphson iteration plays a crucial role. From Section 1.2 it is clear, that there is a vast literature and ongoing research on different integration schemes suitable for fictitious domain methods. The available approaches differ in their (i) accuracy, (ii) speed, (iii) robustness, (iv) implementational complexity, (v) stability during Newton–Raphson iteration, (vi) assumptions regarding the definition and dimensionality of the intersecting geometry (parametric, implicit, voxel, STL, etc.), and (vi) assumption of the integrand's nature.

In our contribution, the adaptive integration scheme based on an *Octree Decomposition* (OTD) is in the focus; an approach that has emerged in the early years of the FCM [5,22,29,34], and is still widely used nowadays [40,41,50, 79,80]. The popularity of the OTD is due to its simple implementation, straightforward execution, and robustness, since it can handle arbitrary cut cases and supports most geometry description. Furthermore, since it does not require an explicit search for the boundary, there are no additional iterations required during the procedure. The method is explained in more detail in Section 2.3. Even if other methods are based on a different notion, they often rely on an OTD, e.g., when computing the right-hand side of the moment fitting approach [68], or use it as a fallback strategy if the initially intended integration algorithm fails [60,61]. Combinations exist as well, such as the adaptive moment fitting, which relies on the OTD for being able to handle non-linear problems [24].

Although the OTD-based integration scheme has significant benefits, it has also its drawbacks, e.g., an exponentially increasing number of sub-cells n_{sc} with respect to the refinement level $\mathcal{R}$. Thus, if a high integration accuracy is required, a large amount of integration points will be generated, having a major impact on the computational time. For reducing the computational time, multiple methods have been proposed in the recent years, including subsequent decrease of quadrature order for smaller sub-cells [22,81], replacement of fictitious integration points by a significantly lower set [22,81], extension by Boolean operations [82], and reusing/updating certain integration points while simulating milling processes [39]. Note that the above method can be combined with each other as well.

Another solution for decreasing the number of integration points was proposed by the authors in a 2D setting [52,83], where the number of sub-cells resulting from the quadtree-decomposition (QTD) is reduced by transforming the sub-cells into an equivalent pixel-image to which standard image compression schemes can be applied. Finally, the compressed image is transformed back to the local space of the cut cell, yielding a significantly reduced set of sub-cells. This leads to reduced computational effort during the numerical integration while maintaining the same integration accuracy for piece-wise polynomial integrands. In this contribution, we extend the idea proposed in Refs. [52,83] to 3D problems, such that the LIM resulting from an OTD is translated into an equivalent voxel domain in which constant voxel regions can be merged using data compression techniques. The proposed integration scheme, combined with the fictitious integration points reduction approach [22,81], is tested in the context of non-linear mechanics with a Neo-Hookean material model and finite strains. Due to the computation of the discontinuous integrals in numerous load steps an efficient integration is especially required. The numerical examples in this article not only show, that a significant amount of integration points can be saved using the proposed method, but that it can handle highly complex geometries with ease as well. Finally, when investigating non-linear problems in Section 4, the proposed method shows no loss of robustness during the Newton–Raphson iteration and despite the non-polynomial nature of the integrand, the overall simulation accuracy is also maintained.

2. Non-linear analysis with the finite cell method

In the previous section, the key features of the FCM were already introduced. In this section, the governing equations for materially and geometrically non-linear analysis are presented and adopted to the FCM. The idea of large deformation analysis using the FCM is illustrated in Fig. 2 for a two-dimensional case. Note that the fictitious cells, which have been present in Fig. 1, are not depicted here, since they are not part of the simulation.

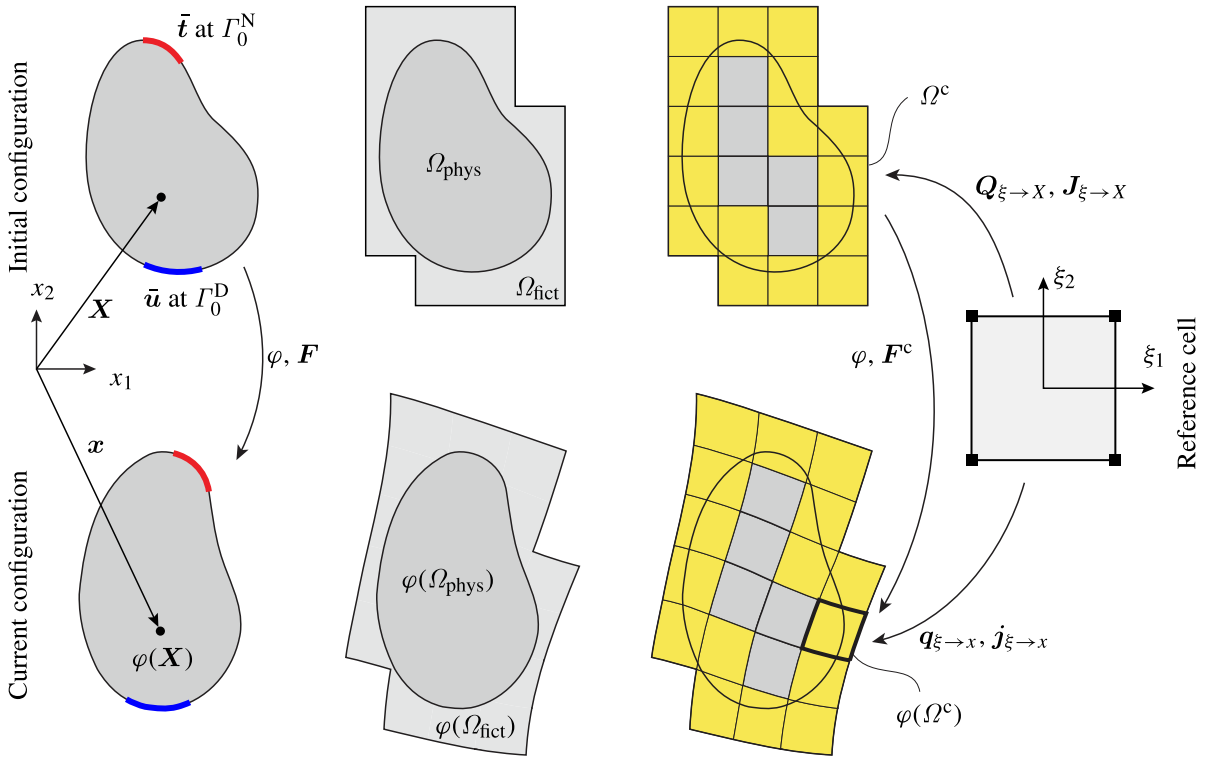


Fig. 2. Simulation of large deformations using the finite cell method.

2.1. Kinematics and constitutive relations

Here, we will consider a hyperelastic material model including both material and geometrical non-linearities. Let $\boldsymbol{X}$ and $\boldsymbol{x} = \varphi(\boldsymbol{X})$ denote the position vectors of a material point in the initial and current configurations. Using the displacement vector $\boldsymbol{u}$, the deformation map is defined as $\varphi(\boldsymbol{X}) = \boldsymbol{X} + \boldsymbol{u}(\boldsymbol{X})$. Furthermore, let $\boldsymbol{F} = \mathrm{Grad}(\varphi) = \mathbf{1} + \mathrm{Grad}(\boldsymbol{u})$ be the deformation gradient, where $\mathrm{Grad}(\cdot)$ refers to the gradient operator w.r.t. $\boldsymbol{X}$. In this contribution, for simulating finite strains, we follow the total Lagrangian formulation. Without derivation, the non-linear weak form of equilibrium in the initial configuration reads [19,84,85]

$$G_{\mathrm{e}}^{\alpha} = \int_{\Omega_{\mathrm{e}}} \alpha\, \boldsymbol{S} \cdot \delta \boldsymbol{E}\, \mathrm{d}V - \int_{\Gamma_0^N} \bar{\boldsymbol{t}} \cdot \delta \boldsymbol{u}\, \mathrm{d}A = 0\,, \tag{2}$$

which in the context of FCM, is formulated over Ω_{e}. Thus, the standard weak form is extended by the indicator function α given in Eq. (1). The first term in Eq. (2) considers the internal virtual work, while the second term corresponds to the virtual work of the external forces. For sake of simplicity, no body loads and inertia terms are considered. Furthermore, $\delta \boldsymbol{u}$ refers to the variation of displacements, $\delta \boldsymbol{E}$ is the variation of the Green–Lagrange strain tensor, and $\bar{\boldsymbol{t}}$ is the applied traction on the surface Γ_0^{N}, as depicted in Fig. 2. The second Piola–Kirchhoff stress tensor $\boldsymbol{S}$ is computed using the strain energy function W of the hyperelastic material model [86]

$$W = \frac{\mu}{2}\,(\mathrm{tr}\,(\boldsymbol{C}) - 3) + \frac{\lambda}{4}\,\left(J^2 - 1\right) - \left(\frac{\lambda}{2} + \mu\right) \ln\,(J)\,. \tag{3}$$

Here, λ and μ are the Lamé constants, $\boldsymbol{C} = \boldsymbol{F}^{\mathrm{T}}\boldsymbol{F}$ is the right Cauchy–Green tensor and $J = \sqrt{\det\,(\boldsymbol{C})}$. Thus, $\boldsymbol{S}$ can be expressed as

$$\boldsymbol{S} = 2\,\frac{\partial W}{\partial \boldsymbol{C}} = \frac{\lambda}{2}(J^2 - 1)\boldsymbol{C}^{-1} + \mu(\mathbf{1} - \boldsymbol{C}^{-1})\,. \tag{4}$$

Note that Eq. (2), is highly non-linear and will be solved iteratively with the help of a Newton–Raphson scheme. After taking the directional derivative of G_{e}^{α}, the linearized weak form reads

$$\int_{\Omega_{\mathrm{e}}} \alpha \left(\mathrm{Grad}(\Delta \boldsymbol{u})\, \boldsymbol{S} \cdot \mathrm{Grad}(\delta \boldsymbol{u}) + \delta \boldsymbol{E} \cdot \mathbb{C}\, \Delta \boldsymbol{E} \right) \mathrm{d}V = \int_{\Gamma_0^N} \bar{\boldsymbol{t}} \cdot \delta \boldsymbol{u}\, \mathrm{d}A - \int_{\Omega_{\mathrm{e}}} \alpha\, \boldsymbol{S} \cdot \delta \boldsymbol{E}\, \mathrm{d}V. \tag{5}$$

where $\Delta \boldsymbol{u}$ and $\Delta \boldsymbol{E}$ define the increment of the displacement vector and the Green–Lagrange strain tensor, respectively.

2.2. Discretization of the weak form in the initial configuration

The linearized weak form over Ω_{e} in Eq. (5) can now be discretized in the initial configuration using a Cartesian mesh consisting of n_{c} finite cells, as depicted in Fig. 2. Each cell has its local coordinate system $\boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3]^{\mathrm{T}}$ and a set of n high-order shape functions $\{N_k(\boldsymbol{\xi})\}_{k=1}^{n}$ based on integrated Legendre polynomials [87,88]. Due to the Cartesian nature of the unfitted mesh, the mapping of a given cell from its local coordinates to the initial configuration $\boldsymbol{Q}_{\xi \to X}$ is linear in $\boldsymbol{\xi}$. Consequently, $\boldsymbol{J}_{\xi \to X} = \nabla_{\xi} \otimes \boldsymbol{Q}_{\xi \to X}$ is constant. The discretized problem results in

$$\boldsymbol{K}_{\mathrm{T}}^{i}(\boldsymbol{u}^{i})\, \Delta \boldsymbol{u}^{i+1} = \boldsymbol{F}_{\mathrm{int}}^{i}(\boldsymbol{u}^{i}) - \lambda_l \boldsymbol{F}_{\mathrm{ext}}\,, \tag{6}$$

where $\boldsymbol{K}_{\mathrm{T}}^{i}(\boldsymbol{u}^{i})$ refers to the global tangent stiffness matrix, furthermore, $\boldsymbol{F}_{\mathrm{int}}$ and $\boldsymbol{F}_{\mathrm{ext}}$ correspond to the global internal and external load vectors, respectively. The parameter λ_l defines a load factor at the load increment l. The equation system in Eq. (6) is solved at every Newton–Raphson iteration i for obtaining $\Delta \boldsymbol{u}^{i+1}$. The global quantities in Eq. (6) are obtained by the assembly of the cell-specific local quantities following the standard Bubnov–Galerkin approach. The tangential stiffness matrix $\boldsymbol{k}_{\mathrm{T}}^{c,i}$, the internal load vector $\boldsymbol{f}_{\mathrm{int}}^{c,i}$, and the external load vector $\boldsymbol{f}_{\mathrm{ext}}^{c,i}$ of the cell c at the Newton iteration step i are defined as

$$\boldsymbol{k}_{\mathrm{T}}^{c,i} = \int_{\Omega^c} \alpha\, (\boldsymbol{B}^{\mathrm{T}} \mathbb{C}^{\mathrm{V}}\, \boldsymbol{B} + \boldsymbol{G}^{\mathrm{T}} \boldsymbol{S}^{\mathrm{V}}\, \boldsymbol{G})\, \mathrm{d}V, \qquad \boldsymbol{f}_{\mathrm{int}}^{c,i} = \int_{\Omega^c} \alpha\, \boldsymbol{G}^{\mathrm{T}} \boldsymbol{S}^{\mathrm{V}}\, \mathrm{d}V, \qquad \boldsymbol{f}_{\mathrm{ext}}^{c,i} = \int_{\Gamma_0^{\mathrm{N}}} \boldsymbol{N}^{\mathrm{T}}\, \bar{\boldsymbol{t}}\, \mathrm{d}A. \tag{7}$$

Here, $\mathbb{C}^{\mathrm{V}}$ and $\boldsymbol{S}^{\mathrm{V}}$ are the elasticity and second Piola–Kirchoff stress tensors in Voigt notation, respectively. Furthermore, $\boldsymbol{N}$ is a matrix containing the shape functions, $\boldsymbol{G}$ corresponds to the discrete gradient operator and $\boldsymbol{B}$ is the strain operator. Both $\boldsymbol{G}$ and $\boldsymbol{B}$ contain polynomials based on the partial derivatives of the shape functions w.r.t. $\boldsymbol{X}$, while $\boldsymbol{B}$ also contains terms related to $\boldsymbol{F}^{c,i} = \boldsymbol{1} + \mathrm{Grad}(\boldsymbol{u}^{c,i})$. The integrals given in Eq. (7) are formulated in the initial configuration. Thus, for the computation of the cell matrices in the reference domain of the cells, the change of integration coordinates and pullback of global derivatives have to be taken into account appropriately. Note that due to the Cartesian mesh, $\det(\boldsymbol{J}_{\xi \to X})$ is a constant, and computing global derivatives w.r.t. $\boldsymbol{\xi}$ only involves a simple scaling procedure. Generally, $\boldsymbol{k}_{\mathrm{T}}^{c,i}$ and $\boldsymbol{f}_{\mathrm{int}}^{c,i}$ involve computing integrals I^{c} of the form

$$I^{\mathrm{c}} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \underbrace{\alpha\big(\boldsymbol{Q}_{\xi \to X}(\boldsymbol{\xi})\big)\mathcal{F}(\boldsymbol{\xi})\det(\boldsymbol{J}_{\xi \to X})}_{F(\boldsymbol{\xi})}\, \mathrm{d}\xi_1 \mathrm{d}\xi_2 \mathrm{d}\xi_3\,, \tag{8}$$

where $\mathcal{F}(\boldsymbol{\xi})$ depends on the integrands of $\boldsymbol{k}_{\mathrm{T}}^{c,i}$ and $\boldsymbol{f}_{\mathrm{int}}^{c,i}$. The function $\mathcal{F}(\boldsymbol{\xi})$ is non-polynomial due to the hyperelastic constitutive model and the presence of $\boldsymbol{C}^{-1} = (\boldsymbol{F}^{\mathrm{T}}\boldsymbol{F})^{-1}$ in Eq. (4). Furthermore, due to the multiplication by α, the entire integrand $F(\boldsymbol{\xi})$ is also discontinuous along $\partial\Omega$. In the following sub-section, the OTD-based integration scheme often used in unfitted approaches is discussed for solving discontinuous integrals.

2.3. Integration of the cut cells via octree-decomposition

As discussed in Section 1.2, there are numerous integration schemes that can be used for computing Eq. (8). They usually differ in their robustness, accuracy and complexity. Due to reasons discussed in Section 1.3, in our contribution, the adaptive integration scheme based on an octree-decomposition is in the focus. The procedure is based on introducing a local integration mesh with an octree structure in the cell (Fig. 3), which serves integration purposes only and does not increase the total degrees of freedoms of the system. The straightforward execution of the OTD relies on a recursive sub-division of a cut cell into eight sub-cells, which are again sub-divided into even

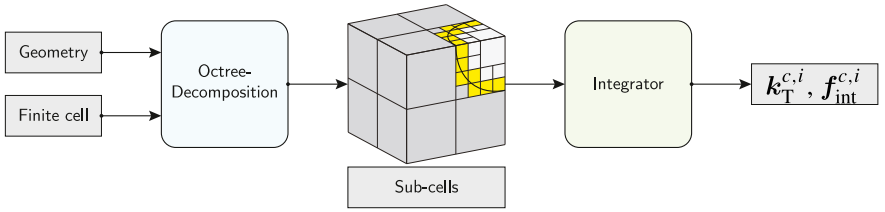


Fig. 3. Flowchart of the OTD-based integration scheme. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

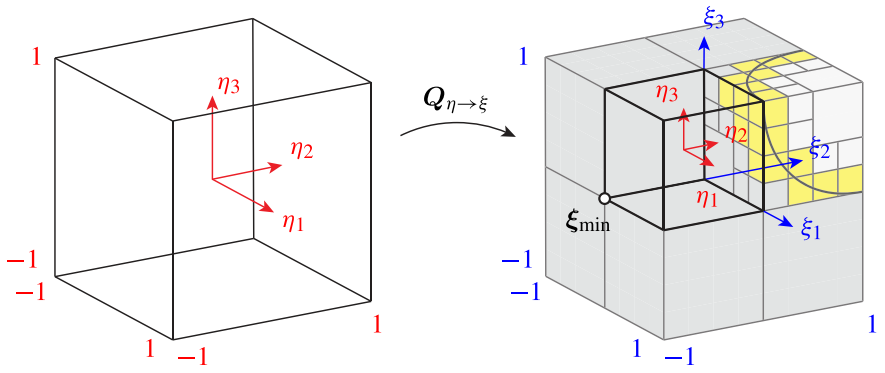


Fig. 4. Mapping of a given sub-cell to the local space of its parent cell.

smaller sub-cells, if they are cut by the boundary $\partial\Omega$. In the case of Fig. 3, the physical and fictitious domains are separated by a spherical boundary. The procedure ends when a given refinement level $\mathcal{R}_{\max}$ is reached and results in a set of cube-shaped sub-cells, which have decreased size and increased density towards $\partial\Omega$. Whether a given sub-cell is intersected can be easily evaluated for most geometry types (implicit, STL, CAD, voxel) by finding two points within the sub-cell that belong to two different domains.[1] Such a procedure is often referred to as a point membership classification (PMC). Since neither the decomposition nor the integration over the sub-cell requires explicitly finding the boundary, the OTD-based integration scheme can handle arbitrarily complex cut cases as well. Just as finite cells, the sub-cells can be also classified as physical, fictitious or cut, as indicated by the gray, light gray and yellow colors in Fig. 3, respectively.

Each sub-cell has its own local coordinate system $\{\eta_1, \eta_2, \eta_3\}$ and linear mapping function $\boldsymbol{Q}_{\eta\to\xi}$ for establishing the relation to the $\{\xi_1, \xi_2, \xi_3\}$-space of the parent cell, as depicted in Fig. 4.

$$\boldsymbol{\xi} = \boldsymbol{Q}_{\eta\to\xi}(\boldsymbol{\eta}) = \begin{bmatrix} \xi_{1\,\min} \\ \xi_{2\,\min} \\ \xi_{3\,\min} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} h_{\xi_1} & 0 & 0 \\ 0 & h_{\xi_2} & 0 \\ 0 & 0 & h_{\xi_3} \end{bmatrix} \cdot \begin{bmatrix} 1+\eta_1 \\ 1+\eta_2 \\ 1+\eta_3 \end{bmatrix} \tag{9}$$

In the above equation, $\xi_{1\,\min}$, $\xi_{2\,\min}$, and $\xi_{3\,\min}$ denote the coordinates of the highlighted corner $\boldsymbol{\xi}_{\min}$ of the sub-cell in Fig. 4. Furthermore, h_{ξ_1}, h_{ξ_2} and h_{ξ_3} mark the side lengths of the sub-cell in the $\{\xi_1, \xi_2, \xi_3\}$-space, where for the cube-shaped sub-cells, $h_{\xi_1} = h_{\xi_2} = h_{\xi_3}$. After the OTD of the cell, the discontinuous integral over the cut cell given in Eq. (8) is computed by integrating over the n_{sc} sub-cells individually

$$I^{\text{c}} = \sum_{k=1}^{n_{\text{sc}}} I_k^{\text{sc}}\,, \tag{10}$$

where the integral over the kth sub-cell is defined as

$$I_k^{\text{sc}} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} F\big(\boldsymbol{Q}_{\eta\to\xi}^{(k)}(\boldsymbol{\eta})\big)\,\det(\boldsymbol{J}_{\eta\to\xi}^{(k)})\;\mathrm{d}\eta_1\mathrm{d}\eta_2\mathrm{d}\eta_3\,. \tag{11}$$

[1] Of course, for certain geometry types, such as spheres, cylinders, ellipsoids, a cut case can be determined analytically as well.

Due to the change of integration coordinates, when integrating over the sub-cells, both the geometry mapping of the k^{th} sub-cell $\boldsymbol{Q}_{\eta\to\xi}$ and the determinant of its Jacobian $\boldsymbol{J}_{\eta\to\xi}$ must be taken into account. Following from Eq. (9), the Jacobian is computed as

$$\boldsymbol{J}_{\eta\to\xi} = \nabla_\eta \otimes \boldsymbol{Q}_{\eta\to\xi} = \frac{1}{2}\begin{bmatrix} h_{\xi_1} & 0 & 0 \\ 0 & h_{\xi_2} & 0 \\ 0 & 0 & h_{\xi_3} \end{bmatrix}, \tag{12}$$

which contains constant values only. Consequently, $\det(\boldsymbol{J}_{\eta\to\xi})$ in Eq. (11) is also constant and can be excluded from the integral. Generally, the integral in Eq. (11) is evaluated via Gaussian quadrature according to

$$I_k^{\text{sc}} = \det(\boldsymbol{J}_{\eta\to\xi}^{(k)}) \sum_{j=1}^{n_{\text{G}}} w_j F\left(\boldsymbol{Q}_{\eta\to\xi}^{(k)}(\boldsymbol{\eta}_j)\right), \tag{13}$$

where n_{G} is the total number of integration points within the sub-cell, while $\boldsymbol{\eta}_j \in [-1,1]\times[-1,1]\times[-1,1]$ and w_j are the position and weight of the jth integration point, respectively.

In this contribution, we follow the approach by Abedian et al. [22,81], for preventing ill-conditioning of the problem: First, a set of integration points according to a chosen quadrature rule is distributed within the entire cell. Then, only the integration points lying in Ω_{fict} are considered with the scaling factor $\alpha = 10^{-q}$. Thus, no fictitious sub-cells are required, and ill-conditioning is avoided at the cell-level. For the numerical examples in Section 4, $q = 5$ is used.

In case of uncut sub-cells, a high integration accuracy can be achieved, if a sufficiently accurate quadrature rule is chosen, and if F in Eq. (13) is a polynomial defined on the given sub-cell, even machine precision is possible. On the other hand, if the given sub-cell is cut, meaning F is discontinuous, Gaussian quadrature cannot integrate exactly. However, with sufficient refinement levels $\mathcal{R}$, the cut sub-cells are relatively small and the integration error is localized to a small domain.

3. Compression of sub-cells

Although the OTD-based integration scheme has significant benefits, it has also its drawbacks, e.g., an exponentially increasing number of sub-cells n_{sc} with respect to the refinement level $\mathcal{R}$. Thus, if a high integration accuracy is required, a large amount of integration points will be generated, having a major impact on the computational time. A solution to this phenomenon was already proposed by the authors in a 2D setting [52,83], where the number of sub-cells resulting from the quadtree-decomposition (QTD) is reduced by transforming the sub-cells into an equivalent pixel-image to which standard image compression schemes can be applied. Finally, the compressed image is transformed back into the local space of the cut cell, yielding a reduced set of sub-cells.

In the following, the procedure developed by Petö et al. [83] is extended to 3D problems, where in each cut cell, the set of cube-shaped sub-cells $\boldsymbol{\mathcal{S}}$ resulting from the OTD is converted to a new set $\boldsymbol{\mathcal{S}}_{\text{C}}$ consisting of fewer sub-cells.[2] The proposed compression scheme has the following benefits:

1. Due to the reduction of sub-cells, the time spent on computing the cell matrices is significantly lower. This is especially beneficial in case of non-linear FCM analyses, where once a reduced set of integration points is derived for a given mesh, it can be re-used in numerous load steps when computing the cell matrices in Eq. (7). Note that if a re-meshing is needed during the simulation of finite strains problems, the OTD and consequently, the compression of sub-cells have to be regenerated.
2. As depicted in Figs. 5 and 6, the compressed sub-cells are not cube-shaped anymore, but arbitrary block-shaped, i.e., generally, $h_{\xi_1} \neq h_{\xi_2} \neq h_{\xi_3}$ holds in Eq. (12). However, since the rectangular shape of the sub-cells is still preserved, the mapping $\boldsymbol{Q}_{\eta\to\xi}$ remains linear and $\det(\boldsymbol{J}_{\eta\to\xi})$ in Eq. (11) is still constant.
3. Although the number of integration points is significantly reduced, according to our numerical tests in Section 4, stability of the Newton–Raphson iteration in Eq. (6) during non-linear FCM analysis is still preserved.[3]

[2] Note that *fewer* does necessarily mean *minimal*. In a two dimensional case, besides performing the compression on a pixel image, an approach based on the *Minimal Rectangular Decomposition* theory also exists, that *guarantees* the fewest number of rectangular sub-cells possible [83]. However, according to our knowledge, no method exists in 3D that would guarantee a minimal set of block-shaped sub-cells.

[3] This is not necessarily the case for other methods that aim to reduce the number of integration points, such as the Moment Fitting, which is only suitable for non-linear simulation if further measures are taken [24,68,69].

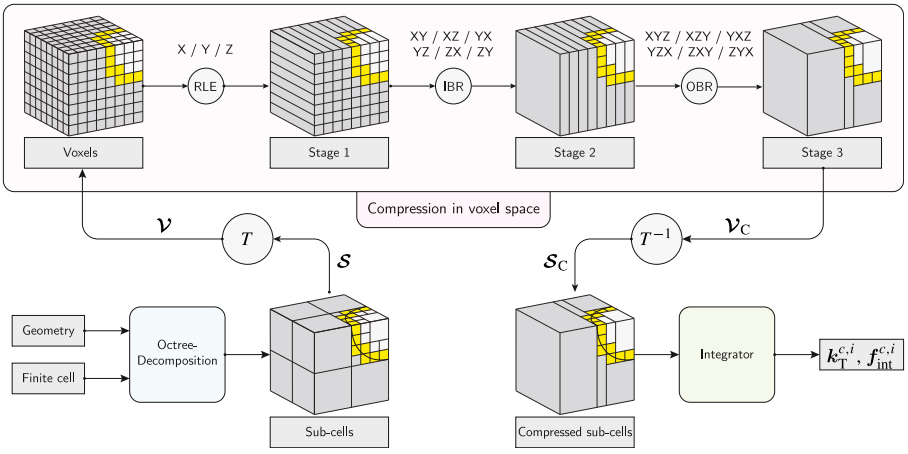


Fig. 5. Flowchart of the compression process embedded in the integration procedure as an intermediate step.

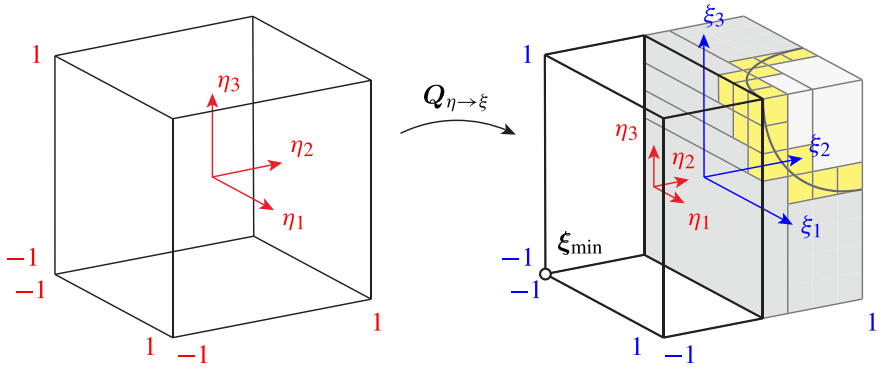


Fig. 6. Mapping of a given sub-cell $\mathcal{S} \in \boldsymbol{\mathcal{S}}_C$ to the local space of its parent cell.

4. Furthermore, since the compression is only applied to sub-cells above which the integrand is continuous, i.e., sub-cells that are not intersected by $\partial\Omega$, the integration accuracy is only marginally impacted and if the integrand is in fact a polynomial, the integration accuracy is preserved.
5. Finally, the compression can be implemented as an intermediate step between the OTD and the integration over the sub-cells (Fig. 3 vs. Fig. 5) and requires no change in an already existing code. Thus, it can be easily added in a modular way to FCM software with OTD-based integration schemes.

3.1. Outline of the algorithm

In Fig. 5, a flowchart of the OTD-based integration scheme extended by the compression of sub-cells is depicted. For reducing the number of sub-cells, an additional step is added to the integration pipeline of Fig. 3, which starts by transforming the original set of sub-cells $\boldsymbol{\mathcal{S}}$ to an equivalent set of voxels $\boldsymbol{\mathcal{V}}$ consisting of $n \times n \times n$ voxels using a simple function $T(\boldsymbol{\mathcal{S}}) = \boldsymbol{\mathcal{V}}$. Then, the compression is performed in the voxel space $C(\boldsymbol{\mathcal{V}}) = \boldsymbol{\mathcal{V}}_C$ and the results are transformed back by $T^{-1}(\boldsymbol{\mathcal{V}}_C) = \boldsymbol{\mathcal{S}}_C$ to obtain a new set of sub-cells with lower cardinality. In the following, the procedure is explained in more detail.

3.1.1. Voxel space

For a local octree mesh with refinement level $\mathcal{R}$, an equivalent voxel domain $\boldsymbol{\mathcal{V}}$ is defined consisting of $n \times n \times n$ voxels, where $n = 2^{\mathcal{R}}$, such that each leaf sub-cell corresponds to a single voxel in $\mathcal{V} \in \boldsymbol{\mathcal{V}}$, that is defined by its unique coordinates $\boldsymbol{r} = [r_x\,, r_y\,, r_z] \in [1, n] \times [1, n] \times [1, n]$. Each sub-cell $\mathcal{S} \in \boldsymbol{\mathcal{S}}$ marks a sub-set of voxels $\mathcal{S}^V \subset \boldsymbol{\mathcal{V}}$

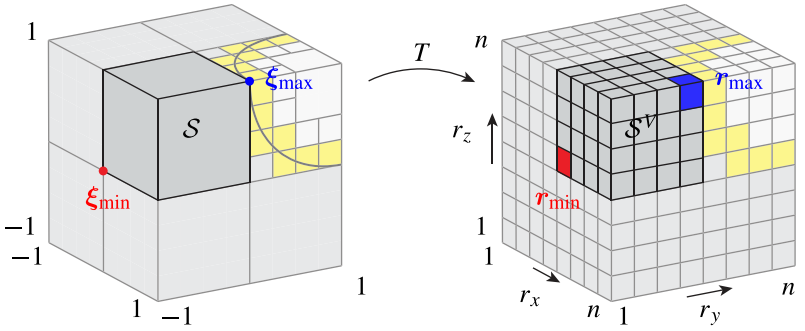


Fig. 7. A single sub-cell $\mathcal{S} \in \boldsymbol{\mathcal{S}}$ and its equivalent voxel region $\mathcal{S}^V \subset \boldsymbol{\mathcal{V}}$ characterized by $\boldsymbol{r}_{\min}$ and $\boldsymbol{r}_{\max}$. For the OTD in this example, $\mathcal{R} = 3$ was chosen.

in the voxel space (see Fig. 7), which can be characterized by the minimum and maximum coordinates $r_{i,\min}$ and $r_{i,\max}$ in the ith direction, respectively, such that

$$\mathcal{S}^V = \{\boldsymbol{r} \in \mathbb{N}^3 \mid r_i \in [r_{i,\min}, r_{i,\max}], \quad \text{for } i = x, y, z\}\,. \tag{14}$$

Corresponding to the different sub-cell types (physical, cut, fictitious), the voxels regions are assigned different integer values (e.g. 1, −1, and 0), which can be chosen arbitrarily, however, they must be unique for the different sub-cell types.

3.1.2. Transformation

The transformation $T(\boldsymbol{\mathcal{S}}) = \boldsymbol{\mathcal{V}}$ is realized by transforming each sub-cell separately by $T(\mathcal{S}) = \mathcal{S}^V$, as depicted in Fig. 7. Based on how the local octree mesh is stored, T can be implemented in different ways. If $\mathcal{S}$ contains the information regarding its location in the octree data structure, the same information can be used to mark $\mathcal{S}^V$ in $\boldsymbol{\mathcal{V}}$. Alternatively, if the sub-cells are saved by their coordinates in the $\{\xi_1, \xi_2, \xi_3\}$ coordinate system, $r_{i,\min}$ and $r_{i,\max}$ in Eq. (14) for $\mathcal{S}^V$ can be obtained by

$$r_{i,\min} = 2^{\mathcal{R}-1}\left(1 + \xi_{i,\min}\right) + 1 \tag{15}$$

$$r_{i,\max} = 2^{\mathcal{R}-1}\left(1 + \xi_{i,\max}\right)\,, \tag{16}$$

where $\boldsymbol{\xi}_{\min} = [\xi_{1,\min},\ \xi_{2,\min},\ \xi_{3,\min}]^{\mathrm{T}}$ and $\boldsymbol{\xi}_{\max} = [\xi_{1,\max},\ \xi_{2,\max},\ \xi_{3,\max}]^{\mathrm{T}}$ denote the two corners that span $\mathcal{S}$ in the $\boldsymbol{\xi}$ coordinate system (see Fig. 7). Using the above transformation, the point $\boldsymbol{\xi}_{\min} = [-1,\ -1,\ -1]$ corresponds to $\boldsymbol{r}_{\min} = [1,\ 1,\ 1]$ in the voxel space, and $\boldsymbol{\xi}_{\max} = [1,\ 1,\ 1]$ to $\boldsymbol{r}_{\max} = [2^{\mathcal{R}},\ 2^{\mathcal{R}},\ 2^{\mathcal{R}}]$. Due to their linearity, Eqs. (15) and (16) can be easily inverted, resulting in the necessary functions for $T^{-1}(\mathcal{S}_{\mathrm{C}}^{\mathrm{V}}) = \mathcal{S}_{\mathrm{C}}$

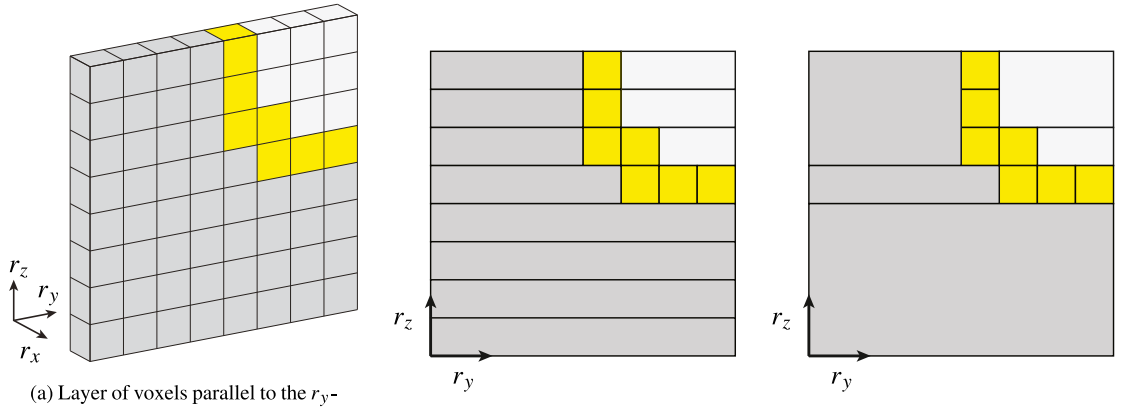
$$\xi_{i,\min} = 2^{1-\mathcal{R}}(r_{i,\min} - 1) - 1 \tag{17}$$

$$\xi_{i,\max} = 2^{1-\mathcal{R}} r_{i,\max} - 1\,. \tag{18}$$

3.1.3. Compression

After transforming the set of sub-cells to an equivalent set of voxels, readily available compression schemes can be applied to $\boldsymbol{\mathcal{V}}$ to form large blocks of connected voxel regions with uniform voxel value. The compressed voxel domain is indicated by $\boldsymbol{\mathcal{V}}_{\mathrm{C}}$. In our implementation, the compression is carried out in 3 consecutive stages using the *Run length encoding* (RLE), *Image block representation* (IBR) and *Object block representation* (OBR) algorithms, explained below. Finally, once the compression is completed, the new set of sub-cells in the local coordinate system $\boldsymbol{\xi}$ is obtained by $\boldsymbol{\mathcal{S}}_{\mathrm{C}} = T^{-1}(\boldsymbol{\mathcal{V}}_{\mathrm{C}})$. Note that the sub-cells $\mathcal{S}_{\mathrm{C}} \in \boldsymbol{\mathcal{S}}_{\mathrm{C}}$ after the compression are no longer cube-shaped, but arbitrary blocks (see Fig. 5), nonetheless, they are still valid sub-cells and the integration can be performed with a constant determinant of $\boldsymbol{J}_{\eta\to\xi}$ in Eq. (11).

Run length encoding: In the first stage, the n^3 voxels are compressed using the RLE algorithm which was originally developed for reducing the number of pixels in 2D images [89,90]. Nonetheless, its extension from 2D pixel images to 3D voxel regions is straightforward. Given a scanning direction $d_{\mathrm{RLE}} \in \{r_x, r_y, r_z\}$ in voxel coordinates (see



(a) Layer of voxels parallel to the r_y-r_z-plane

(b) $d_{\mathrm{RLE}} = r_y$

(c) $d_{\mathrm{IBR}} = r_z$

Fig. 8. Visualization of the RLE and IBR compression steps based on the voxel domain in Fig. 5.

Fig. 7), the RLE scans through all n^2 voxel-chains in the chosen direction and creates merged blocks of voxels having the same value. While scanning a selected direction, one block is completed, if the next voxel exhibits a different value.

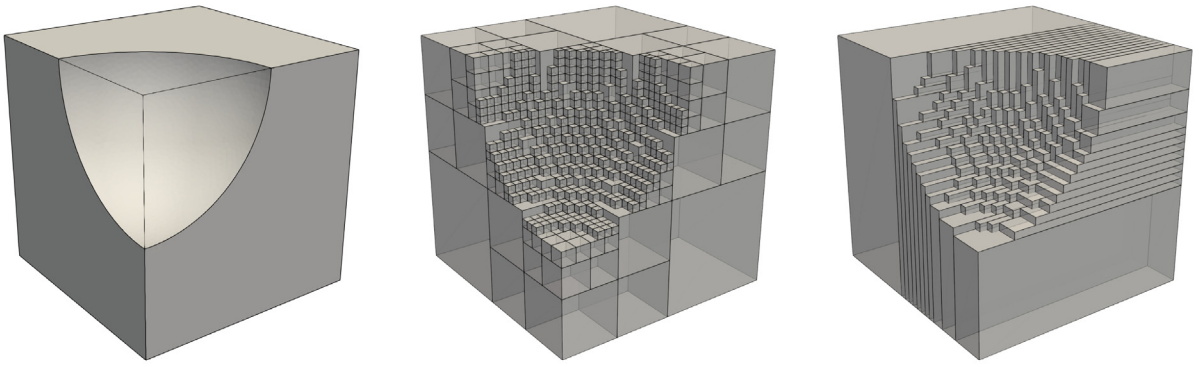
Image block representation: At the second stage, the merged voxel blocks resulting from the RLE are further compressed by the IBR algorithm [90,91]. For doing so, a secondary merging direction $d_{\mathrm{IBR}} \in \{r_x, r_y, r_z\} \setminus d_{\mathrm{RLE}}$ is selected. Once an IBR-direction is chosen, the voxel domain can be divided into n layers all lying parallel to the d_{RLE}-d_{IBR}-plane. In each plane (being equivalent to a pixel image), the blocks resulting from the RLE are further compressed to form even larger domains. Two blocks are merged if (i) they have the same start and end positions in the d_{RLE}-direction and (ii) are neighbors in the d_{IBR}-direction. An example for the IBR compression of a given voxel layer and compression direction $d_{\mathrm{RLE}} = r_y$ and $d_{\mathrm{IBR}} = r_z$ is depicted in Fig. 8. Given that there are three available directions for the RLE, and then two for the IBR, there is a total of six different options for the IBR (XY, XZ, YX, YZ, ZX, ZY), as indicated in Fig. 5.

Object block representation: Finally, at the third stage, the OBR[4] is applied to further compress the planar blocks resulting from the IBR. Given the directions of the RLE and IBR, there is only one direction left in which the OBR can be performed $d_{\mathrm{OBR}} = \{r_x, r_y, r_z\} \setminus \{d_{\mathrm{RLE}}, d_{\mathrm{IBR}}\}$. If two planar blocks in adjacent d_{RLE}-d_{IBR}-layers have overlapping corners in the d_{OBR}-direction, the blocks can be merged. After choosing d_{RLE} and d_{OBR}, there is only one available OBR-direction left. Thus, there are six available settings of how the OBR compression can be performed, as indicated in Fig. 5.

3.2. Performance

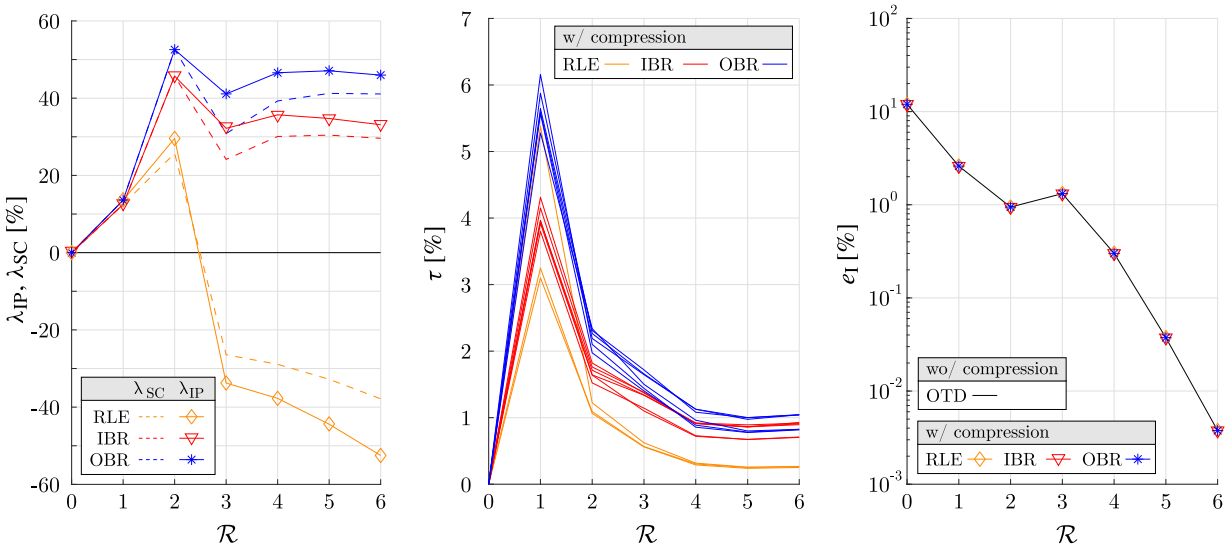
In this section, the performance of the RLE, IBR, and OBR compression schemes is discussed regarding the reduction of integration points (Section 3.2.1), time requirement (Section 3.2.2) and integration accuracy (Section 3.2.3). The mentioned aspects are investigated using a simple problem based on a cubic domain $\Omega_{\mathrm{e}} = [0, 2] \times [0, 2] \times [0, 2]$ intersected by a spherical void region, whose center is located at $[2, 2, 2]$ and has a radius of $r = 1.1$. The domain Ω_{e}, as well as the set of sub-cells before ($\boldsymbol{\mathcal{S}}$) and after the compression ($\boldsymbol{\mathcal{S}}_{\mathrm{C}}$) are depicted in Fig. 9. In each sub-cell, 4×4×4 integration points are distributed for integrating the polynomial function $F = x^4 y^7 z^2$. Since employing n integration points, the integration of polynomials up to the order of $2n - 1$ is exact. Integrals over the physical sub-cells should be computed within machine precision. Due to the fact that ill-conditioning is not an issue for numerical volume estimation, all integration points lying in the fictitious domain are neglected in the current example. The corresponding results for refinement levels $\mathcal{R} = 0$–6 are depicted in Fig. 10.

[4] Note that the name OBR is used in this paper to highlight the similarity to the IBR algorithm.



(a) Investigated domain (b) Before compression (OTD only) (c) After OBR compression (XZY)

Fig. 9. Example of a cubic domain intersected by a spherical hole region (a). The depicted physical sub-cells before and after the compression (b)–(c) correspond to a refinement level of $\mathcal{R} = 5$.



(a) Compression rates (b) Relative time requirement (c) Integration error

Fig. 10. Compression rates, relative time requirement, and integration accuracy for different refinement levels $\mathcal{R}$ and compression algorithms when applied to the problem depicted in Fig. 9.

3.2.1. Compression rate

In the following, for quantifying the saving of integration points in a relative way, we define the compression rate of the integration points as

$$\lambda_{\mathrm{IP}} = \left(1 - \frac{n_{\mathrm{IP}}(\boldsymbol{\mathcal{S}}_{\mathrm{C}})}{n_{\mathrm{IP}}(\boldsymbol{\mathcal{S}})}\right) \cdot 100\%\,, \tag{19}$$

where $n_{\mathrm{IP}}(\cdot)$ expresses the number of integration points distributed in a given set of sub-cells. In that sense, $\lambda_{\mathrm{IP}} > 0$ indicates a meaningful compression, $\lambda_{\mathrm{IP}} = 0$ no compression, and $\lambda_{\mathrm{IP}} < 0$ the undesired case, where the number of integration points has actually increased after applying the compression scheme. Similarly, the compression rate of the sub-cells λ_{SC} can be defined as well based on the number of sub-cells before and after the compression. Observing Fig. 10(a), where both λ_{IP} and λ_{SC} are depicted for the given example, following observations can be made:

1. The RLE compression results in $\lambda_{\text{IP}} < 0$, i.e., instead of reducing the number of integration points, an increase can be observed. Such a phenomenon in case of the RLE algorithm has been observed and discussed already in Ref. [83] for a 2D setting. The negative compression rate can be seen in the fact, that the OTD itself is also a compression scheme when applied to a voxel region, which obviously outperforms the RLE. Due to the poor performance of the RLE compared to the OTD, it is not recommended to use it for merging of sub-cells in the context of FCM. Nonetheless, the RLE algorithm is still needed, as it serves as a basis for the IBR and OBR compression schemes (cf. Fig. 5).
2. Both the IBR and OBR compression schemes yield a significant compression, which grows with an increasing refinement level $\mathcal{R}$ and converges to a stationary value around $\lambda_{\text{IP}} = 35$–45% in case of the current example. Due to its superiority, the focus is kept on the OBR compression in the following sections.
3. Note that as discussed in Section 3.1, there are three possible settings for the RLE compression, six for the IBR and six for the OBR. Generally, these all yield somewhat different compression rates and consequently, three, six, and six different curves should be depicted in Fig. 10(a) for the RLE, IBR and OBR, respectively. The reason for each compression scheme having only one curve is due to the symmetric nature of the chosen example, where the same compression rates are obtained for a chosen algorithm, regardless of the sequence of merging directions. More general cut cases are presented in Section 3.3.

3.2.2. Time requirement

The main goal of the compression is the reduction of the overall computational time of the FCM simulation. How much time can be saved in total, depends on the relation between the time investment for performing the compression t_{comp} and the time saving when computing the integrals over the reduced set of sub-cells t_{int}.

Time investment: Since the compression requires voxel-checking operations only, where no iterations and solutions of equation systems are needed, it poses marginal computational overhead when compared to the OTD algorithm ($t_{\text{comp}} \ll t_{\text{OTD}}$), that has to be executed regardless of the compression. In case of the current example, the relative additional time required for the compression $\tau = t_{\text{comp}}/t_{\text{OTD}} \cdot 100\%$ is indeed negligible, especially for higher values of $\mathcal{R}$. This can be seen in Fig. 10(b), where three, six and six curves are depicted, representing the individual settings for the RLE, IBR and OBR algorithms (Fig. 5), respectively. However, note that t_{comp} and t_{OTD} depend on many factors, such as the (i) chosen programming language, (ii) quality of the written code, (iii) parallelization of the code and (iv) the algorithm for identifying cut sub-cells.

Time saving: The time spent on the integration t_{int} is characterized by two factors, namely (i) the saving of integration points expressed by λ_{IP} and (ii) computational complexity of the integrand. Regarding the complexity of the integrand, the higher the computational effort for computing the integrals, the more likely that the time saved during the integration outweighs the time spent on the compression. It was already shown in Refs. [52,83], that even in the case of linear elastostatics, the overall simulation time can be reduced significantly. In case of non-linear analysis, where the cell matrices in Eq. (7) have to be computed in numerous iteration steps using the same set of integration points, the time saving is even more significant when the compression techniques are employed, making the approach very suitable for such problems. In Sections 4.1.1 and 4.2.1 it is demonstrated that reduction of computational time is almost identical to the reduction of integration points, and thus, a speed-up by $40 - 50\%$ can be achieved for the investigated examples.

3.2.3. Integration accuracy

The integral of $F = x^4 y^7 z^2$ over Ω_{phys} is computed over each sub-cell via Gaussian quadrature according to Eq. (13). A reference value for the integral is obtained analytically using the computer algebra system `Mathematica`: $I_{\text{ref}} = \int_{\Omega_{\text{phys}}} F \, \mathrm{d}\boldsymbol{x} = 93.1213176993456$. Assuming that a meaningful integration order is chosen, the accuracy of numerically solving the integrals over the compressed sub-cells is characterized by the following three properties:

Target of compression: At each level of the OTD, the intersecting boundary is located, within the cut sub-cells. Thus, the integrand over those sub-cells exhibits a strong discontinuity. Since Gaussian quadrature rules show a poor accuracy for such integrands, the integration accuracy strongly depends on the (i) size of the cut sub-cells and (ii) the integration order; or in other words: the integration point density. Since compressing the cut sub-cells leads to an increase of their size and thus, to a decrease of the integration point density, such an approach is not recommended. Of course, it is possible to increase the integration order in the compressed cut sub-cells to recover the accuracy,

however, it contradicts the original goal of the compression, i.e., the reduction of integration points in the first place. Therefore, only the physical and fictitious sub-cells are compressed, and if additionally the approach by Abedian et al. [22,81] is used (see Section 2.3), the fictitious sub-cells are completely discarded and the compression is applied only to the physical sub-cells.

Integrand F: Assuming that the above aspects regarding target of the compression and geometry mapping apply, two cases can be distinguished regarding F in Eq. (10), when investigating the integration accuracy using compressed sub-cells:

1. If F is a polynomial, an exact integration of the enlarged non-cut sub-cells is possible, regardless of their size, provided that the right integration order is used (n_i integration points for a polynomial order of $p_i \leq 2n_i - 1$ in the ith direction). In that case, a significant compression can be performed without any loss of accuracy when compared to the standard OTD. This can be observed for the given example in Fig. 10(c), where the integration error $e_\mathrm{I} = (1 - I/I_\mathrm{ref}) \times 100\%$ is depicted. Here, both the standard OTD and its compressed versions yield the exact integration. Such a scenario applies when a standard FCM simulation of linear elasticity is performed, as investigated by Petö et al. [83] in a 2D setting. In this paper, none of the simulations are conducted in the context of linear elasticity. However, based on Ref. [83] and Fig. 10(c), it is evident that the proposed integration scheme can be applied to 3D linear-elastic problems without any loss of accuracy.
2. If F is not a polynomial, Gaussian quadrature cannot integrate exactly, even if the given sub-cell is not cut and the integrand is continuous. In that case, the integration quality depends on integration point density, which is clearly decreased for the enlarged sub-cells after the compression, if the same integration order is used for $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{S}}_\mathrm{C}$. Thus, a loss of integration accuracy is inevitable. An example for this is depicted in Fig. 11, where the same problem is investigated as throughout this section, however, a non-polynomial function $F = \sin(x)\cos(3y)$ is chosen for the integrand. A reference value for the integral is obtained analytically using again `Mathematica`: $I_\mathrm{ref} = \int_{\Omega_\mathrm{phys}} F \, \mathrm{d}\boldsymbol{x} = -0.3037447157087076$. Although all six merging settings for the OBR algorithm yield the same number of sub-cells (cf. Fig. 10(a)), the integration accuracy is not the same anymore for the different settings (unlike in case of Fig. 10(c)). Instead, it strongly depends on the direction(s) in which the sub-cells are lengthened (i.e. integration point density is decreased). Note that the compression setting YZX yields a set of sub-cells with large lengthened blocks in y-direction, in which the integrand is highly non-polynomial. On the other hand, the OBR setting XZY results in a much higher sub-cell density in that critical direction. Also note that since F does not depend on z, large blocks in z-direction have no effect on the integration accuracy. In this work, non-polynomial integrands are of special importance due to the Neo-Hookean material behavior and large deformations. For realistic engineering problems, the integrand is not given explicitly. Thus, it is not possible to tell a priori, which of the possible elongation directions would minimize the integration error. However, according to our numerical experiments in Section 4, when performing an entire non-linear FCM simulation, the loss of solution accuracy turns out to be negligible.

3.3. Different merging directions

Differing from the example given in Fig. 9, not all OBR settings yield the same number of sub-cells when $\partial\Omega$ has a more general topology. Thus, the question naturally arises, how different the compression ratios can be, and whether it is meaningful to invest the computational time to find an optimal or close to optimal setting. According to our numerical experiments, the compression rates for the six different OBR settings is fairly similar. This can be seen in Fig. 13 for the examples depicted in Fig. 12 (for details regarding the used geometries see Appendix A.1), and also in Table 2 of Section 4. Although a larger deviation between the different settings might occur, this tends to happen for low values of $\mathcal{R}$ and the difference decreases with increasing refinement levels, as depicted in Figs. 13(a) and 13(b). According to our numerical studies, it is assumed that the best compression direction can be associated with the dominating dimension and topology of the boundary segment $\partial\Omega \cap \Omega^\mathrm{c}$. Note that in case of a rather complex cut scenario with no dominating cut direction, all settings yield very similar results, as depicted in Fig. 13(c) for the case where multiple ellipsoids intersect the cell. Despite these findings, it is not completely excluded, that larger variation of the compression rates for other cut cases might occur. Thus, instead of uniformly using a pre-set compression setting in each cut cell, a direction finding algorithm could be employed for suggesting a compression setting in each cell individually. The most straightforward option is to perform the

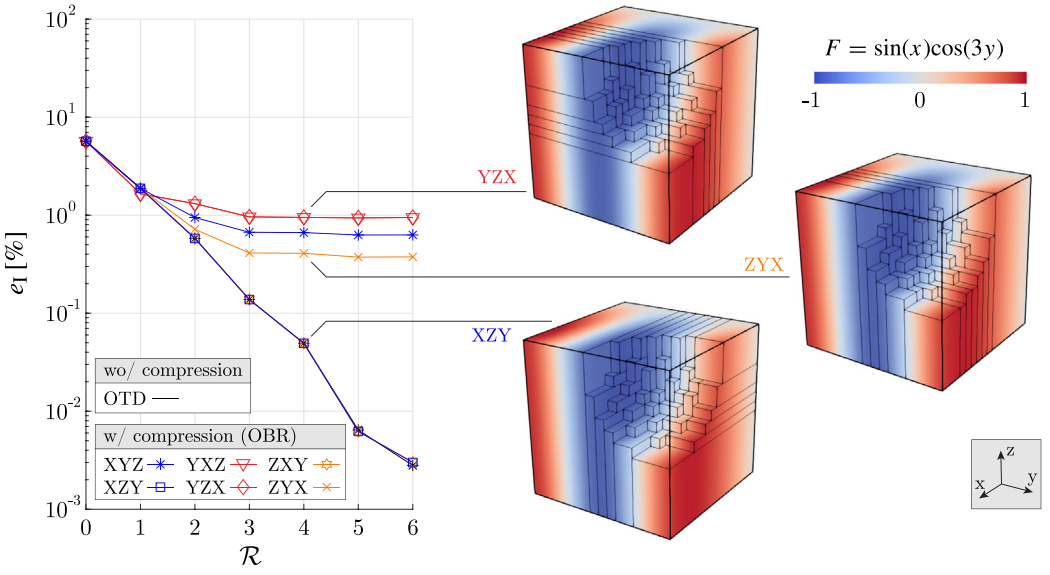
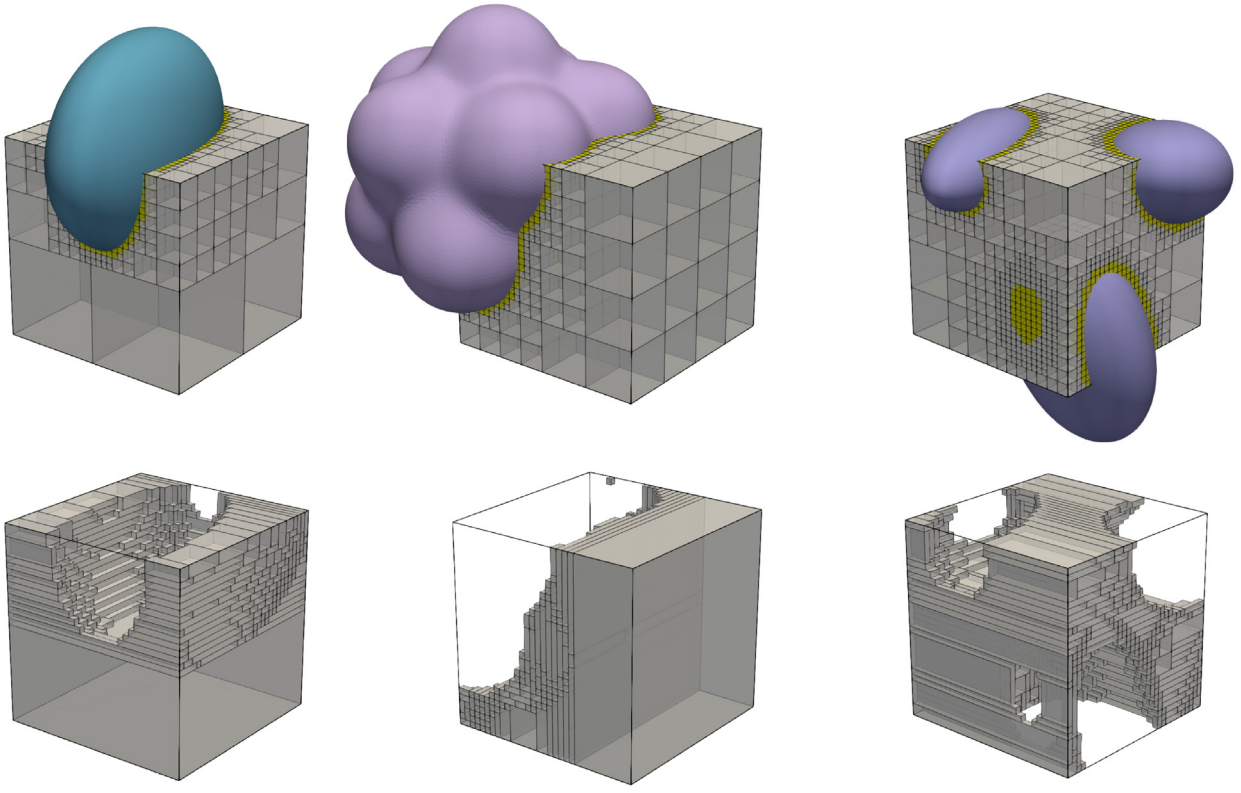
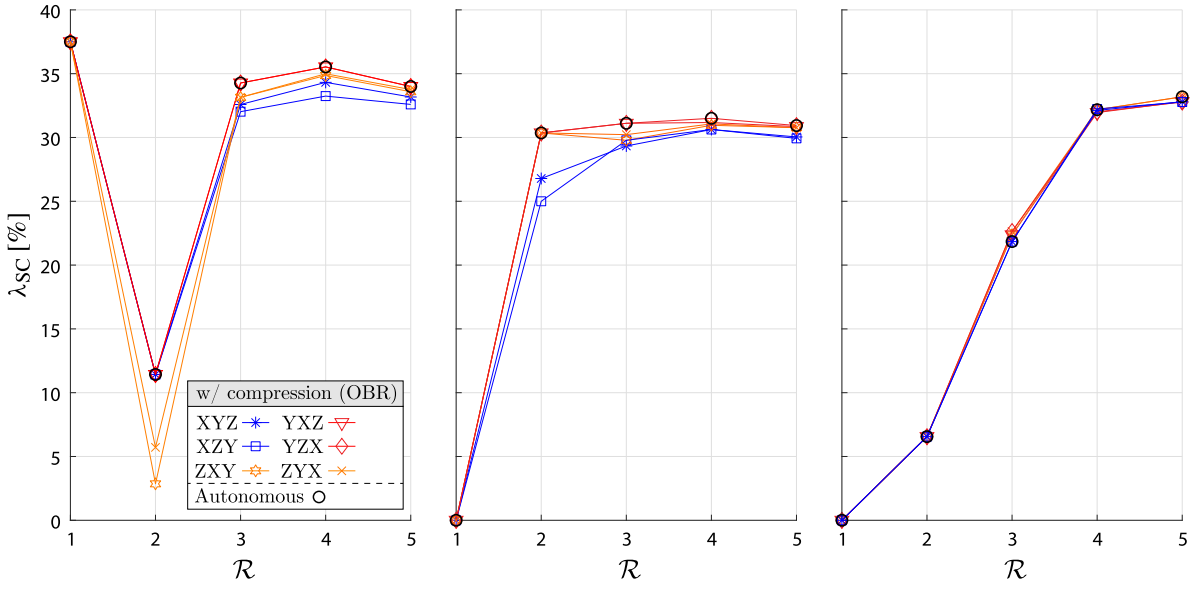


Fig. 11. Influence of the sub-cell orientations on the accuracy when integrating the non-polynomial function $F = \sin(x)\cos(3y)$ over the domain defined at the beginning of Section 3.2.



(a) Single ellipsoid (b) Popcorn flake (c) Multiple ellipsoids

Fig. 12. Examples of different cut scenarios. Top row: OTD sub-cells with intersecting geometries. Bottom row: physical sub-cells after compression. All examples are depicted for $\mathcal{R} = 5$.



(a) Based on Fig. 12a. (b) Based on Fig. 12b. (c) Based on Fig. 12c.

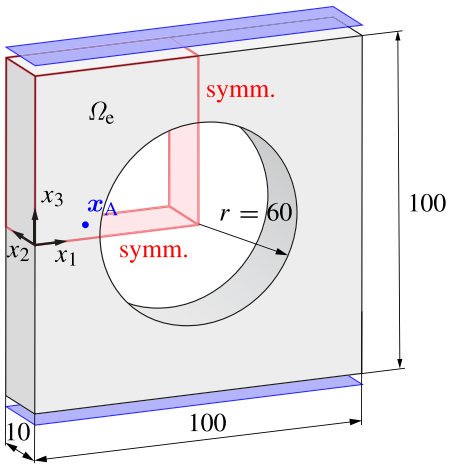
Fig. 13. Reduction of the sub-cells arising from the OTD applied to the examples depicted in Fig. 12 with different refinement levels.

OBR with all of its available settings and chose the one that yields the fewest amount of sub-cells. Based on our tests, an OBR setting yields the highest reduction of sub-cells, if for the RLE in the first stage of the compression (cf. Fig. 5) the best setting among the three available compression directions is chosen. Thus, for finding the best compression setting for the OBR, we propose performing an initial RLE with its three settings, resulting in three different sets of merged sub-cells. Then, d_{RLE} and d_{IBR} for the OBR are chosen based on which directions does the initial RLE yield the fewest and second fewest sub-cells, respectively. As depicted in Fig. 13 by black circles, the proposed approach predicts well an efficient OBR setting for the investigated examples and refinement levels. Note that the finding of the (absolute) best compression direction is not guaranteed (e.g. $\mathcal{R} = 3$ in Fig. 13(c)), however, in such cases the difference in λ-values is already marginal. In the following, we refer to OBR extended by the proposed direction finding algorithm as *autonomous* OBR compression.

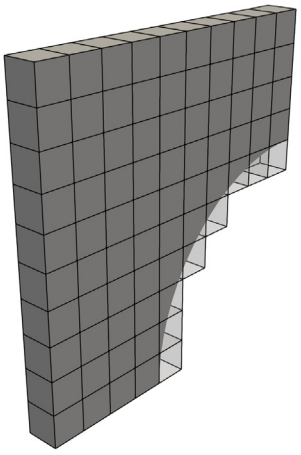
Remark: While it is possible to choose the compression settings such that λ_{IP} is the highest among the available options, this task is not that straightforward when it comes to reducing the integration error. That is to say, to steer the compression in such a way that it results in a set of sub-cells, such that the integration of the non-polynomial integral suffers the least loss of accuracy (cf. Fig. 11). On the one hand, this would require more additional computational effort, on the other hand, for most simulations, the integrand is not known a priori, but it is evaluated on each integration point individually. Thus, this particular option is not available in the first place. Nonetheless, according to the numerical simulations in Section 4, the loss of simulation accuracy is negligible, regardless of the sub-cell orientation.

4. Numerical examples

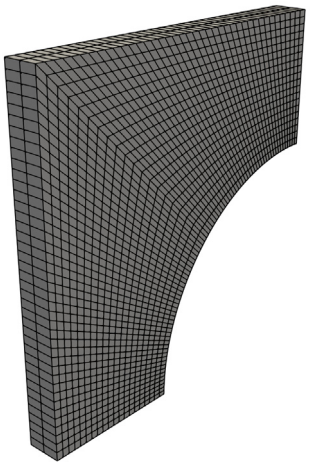
In the current section, the performance and accuracy of the OTD-based integration scheme combined with the OBR compression algorithm is investigated in the context of non-linear analysis via FCM. In all examples, the non-linearity is due to the simulation of finite strains and Neo-Hookean material properties, defined by the Lamé constants $\lambda = 28.846$ MPa and $\mu = 19.231$ MPa. During the numerical integration procedure, in each sub-cell, $(p+1)^3$ integration points are distributed according to the standard Gaussian quadrature rule, where p is the polynomial degree of the Ansatz space. Furthermore, in all examples, fictitious integration points are discarded and the contribution of Ω_{fict} with $\alpha = 10^{-5}$ is realized as described at the end of Section 2.3. Thus, fictitious sub-cells are not considered and only the physical sub-cells are merged. To this end, no re-meshing is simulated,



(a) Plate with a cylindrical hole and simulation domain indicated in red. All dimension are given in [mm].



(b) FCM mesh with $10 \times 10 \times 1$ (78 active) finite cells.



(c) FEM mesh with 3200 finite elements for the reference solution.

Fig. 14. Problem setup using different meshes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

thus, the OTD and the merging of the sub-cells are carried out only once in each cell; the derived integration points are reused in all load steps. In all investigated problems, the reduced set of integration points is computed using `Matlab` and the simulations are conducted in `AdhoC++` using higher-order hierarchic shape functions based on integrated Legendre polynomials [92]. Note that the goal of the following studies is the investigation of the effect of merged sub-cells on the simulation. For a comprehensive investigation of the upcoming problems, please visit the references given in the appropriate sub-sections.

4.1. Plate with cylindrical hole

In the first example, a quarter of a rectangular perforated plate indicated by the red domain in Fig. 14(a) is simulated for tensile ($\bar{u}_3 = 200\,\text{mm}$) and compressive loadings ($\bar{u}_3 = -16\,\text{mm}$) on the surfaces highlighted by blue. The simulation domain Ω_e is discretized by $10\times10\times1$ finite cells with a polynomial order of $p = 4$, and an implicit description of the cylindrical hole is used (see Fig. 14(b)). Dirichlet boundary conditions on the simulation domain are realized by taking the symmetry of the problem into account. For comparison purposes, the reference solution is obtained by an overkill FEM simulation using 3200 elements and $p = 4$ as depicted in Fig. 14(c). To each cut cell in the FCM mesh, an OTD with a refinement level of $\mathcal{R} = 4$ is applied for deriving the local integration mesh (Fig. 15(a)), which is then compressed by the OBR algorithm according to Section 3. In the following sub-sections, the reduction of integration points due to the compression as well as the simulation accuracy is discussed. For quantifying the solution quality, the relative error in the strain energy is evaluated at each load step i

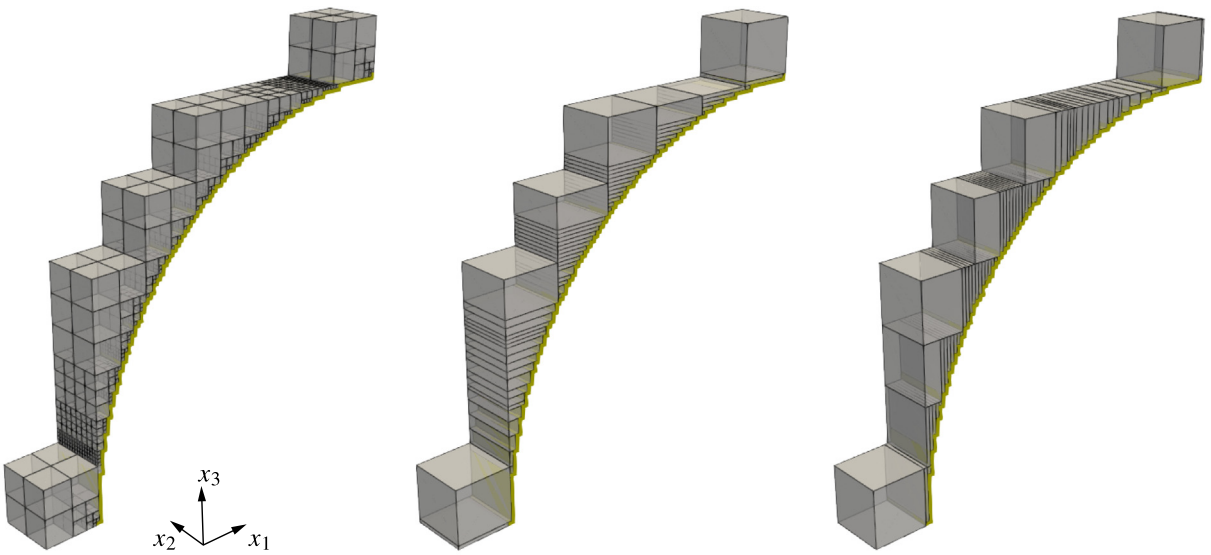
$$e_{\Pi} = \frac{|\Pi^{\text{ref}} - \Pi^{\text{FCM}}|}{\Pi^{\text{ref}}} \cdot 100\%\,, \tag{20}$$

where Π is the total elastic strain energy of the system. Furthermore, for local error estimation, the relative error in the stress component σ_{33} is used at the arbitrary point $\boldsymbol{x}_A = [35,\ 5,\ 4]$ mm

$$e_{\sigma} = \frac{|\sigma_{33}^{\text{ref}}(\boldsymbol{x}_A) - \sigma_{33}^{\text{FCM}}(\boldsymbol{x}_A)|}{|\sigma_{33}^{\text{ref}}(\boldsymbol{x}_A)|} \cdot 100\%\,. \tag{21}$$

Table 1
Reduction of integration points and computational time (excluding the time required for setting up the LIM) when using the compressed octree with merged sub-cells.

	Integration points			Load case: tension		Load case: compression	
Integration schemes	Phys.	Fict.	Reduct.	Time	Reduct.	Time	Reduct.
Octree	402915	760	–	1h 28 min 37 s	–	11 min 10 s	–
Compressed octree	193290	760	51.9%	42 min 56 s	51.5%	5 min 19 s	52.4%

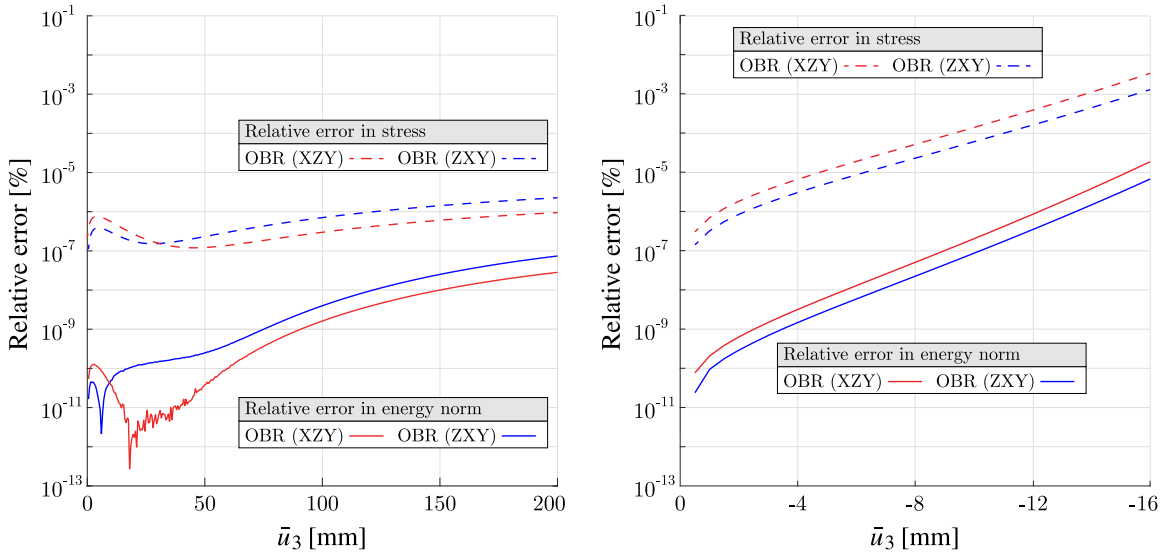


(a) Without compression: OTD (b) With compression: OBR (XZY) (c) With compression: OBR (ZXY)

Fig. 15. Physical sub-cells without and with compression (gray) and cut sub-cells (yellow) in the cut cells around the cylindrical hole (see Fig. 14(b)). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1.1. Compression rates

Due to the simple geometry, all six compression directions of the OBR algorithm reduce the number of integration points by the same amount: $\lambda_{\text{IP}} \approx 52\%$, as given in Table 1. Despite all compression settings leading to the same value of λ_{IP}, the sets of physical sub-cells are not the same: For the current example, the compression directions {XYZ, XZY, YXZ} yield the sub-cells depicted in Fig. 15(b), while the settings {ZYX, ZXY, YZX} lead to the sub-cells of Fig. 15(c). This will be of special importance in Section 4.1.2. The computation of the cut cell matrices is responsible for a major part of the computational time, especially for non-linear problems, where Eq. (7) has to be re-computed several times. In case of the current example, the overall simulation includes 400 load steps in total for the tensile and 32 for the compression load case. Thus, when the reduced set of integration points is used, which is derived in the pre-processing phase, the computational time can be significantly reduced. For the current example, based on the wall-clock time requirement of the computation phase, a speed-up by approximately 52% can be obtained, as indicated in Table 1. To this end, the time measurement includes the time requirement of the entire solution (computation of cell matrices, assembly, Newton iteration) and post-processing stages in `AdhoC++`. Note that the derivations of the octree and compressed octree are not included, since these are performed in `Matlab`, whose time requirement cannot be fairly compared to that of C++. Nonetheless, as discussed in Section 3.2.2, the compression step poses marginal additional computational overhead when compared to the OTD. Furthermore, since the compression step is performed only once, while the reduced set of integration points is re-used in all load steps, the time investment is easily amortized.



(a) Load case: tension

(b) Load case: compression

Fig. 16. Relative error in the strain energy and stress for different load cases when compared to the FCM reference solution obtained using OTD without compression. For the error measures see Eqs. (20) and (21), respectively.

4.1.2. Accuracy

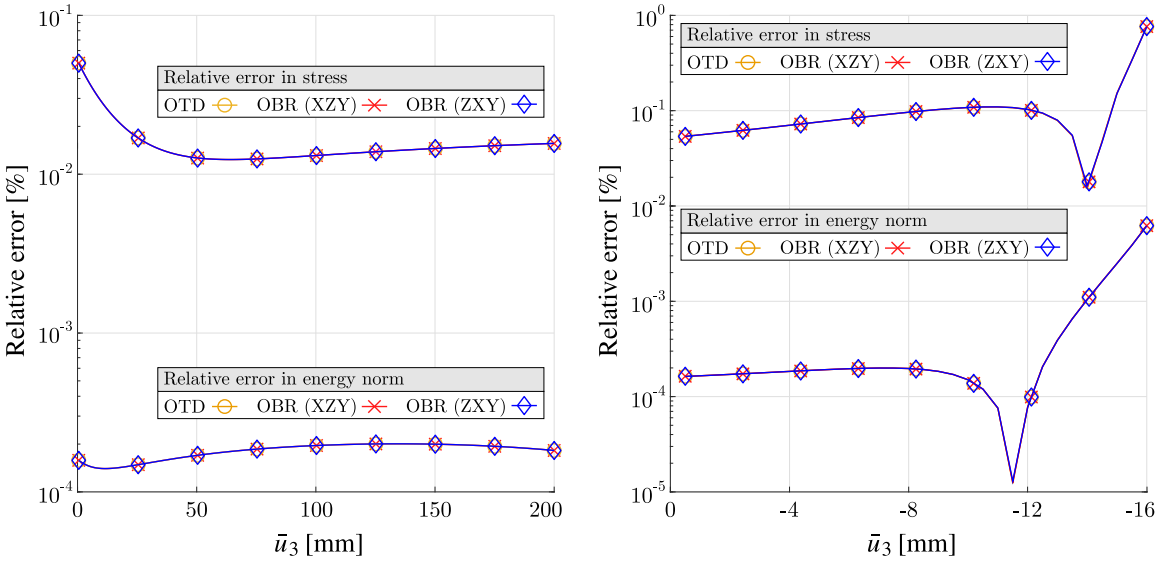
Due to the used material model and simulated large deformations, the integrand in Eq. (8) over the cell is non-polynomial. Thus, as described in Section 3.2.3, a reduction of integration points is expected to decrease the integration accuracy in the physical sub-cells, while the loss of accuracy is dependent on the orientation of the lengthened sub-cells after the compression. This can be clearly observed in Fig. 16, in which the results of an FCM simulation obtained by the compressed octree integration mesh are compared to the standard OTD-based FCM simulation. Note that in Fig. 16, only the settings XZY and ZXY are investigated (cf. Section 4.1.1). For both of these cases, the errors are growing as the loading progresses. This can be explained by the increasing complexity of the integrand for advanced load cases, rendering the decreased integration point density in the merged sub-cells less accurate.

However, for both load cases, the errors are relatively low (at most 10^{-2} % for e_σ in Fig. 16(b)). In fact, when the results are compared to an FEM reference solution, the loss of integration accuracy is negligible, and all the tested integration schemes yield basically identical results, as depicted by the overlapping curves in Fig. 17. This is due to the fact that the additional integration error caused by the compression is so small, that the overall simulation error is dominated by other parameters, e.g., the discretization (element size and polynomial degree).

Finally, we point out, that despite the significant reduction of integration points, not only a sufficient integration accuracy, but also the stability of Newton–Raphson iteration is maintained, which is crucial for non-linear simulations (see Fig. 18).

4.2. Foam

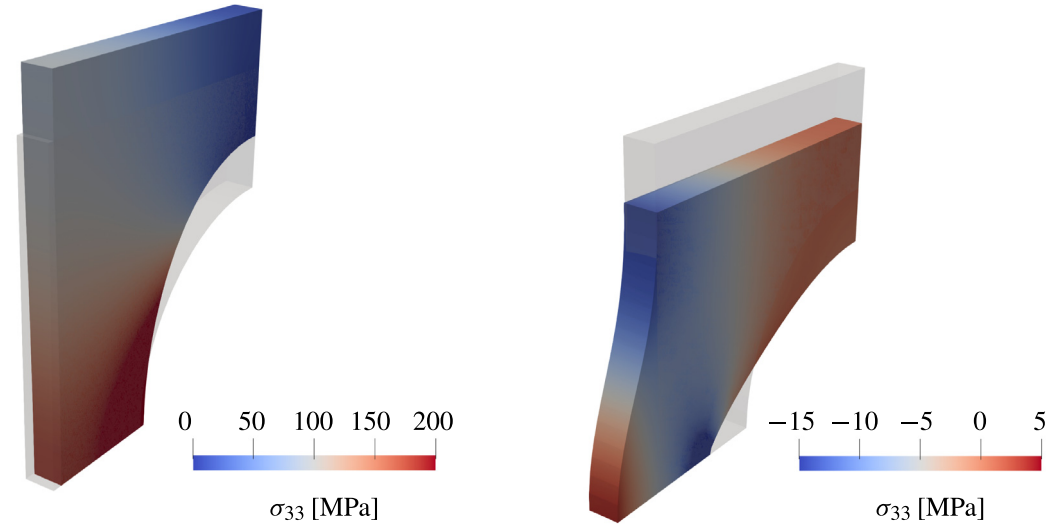
In the second example, the proposed integration scheme is applied to a more complex porous geometry represented by a section of a foam structure subjected to compression loading by a displacement of $\bar{u}_3 = -1.2\,\text{mm}$ [20, 93]. In this contribution, no self-contact is simulated. The embedded domain is discretized by $25\times25\times25$ finite cells, and the polynomial degrees of $p = 2$ and 3 are investigated. The computational mesh without fictitious cells is depicted in Fig. 19(a). Unlike in the previous example, due to the highly complex geometry, the cut cells are intersected by $\partial\Omega$ in various ways. In each of them, an OTD with $\mathcal{R} = 4$ is performed and the resulting physical sub-cells are merged by the OBR algorithm. For numerical integration, $(p+1)^3$ integration points are distributed in the physical and cut sub-cells. A visual comparison of the physical sub-cells without and with compression is depicted in Fig. 20.



(a) Load case: tension

(b) Load case: compression

Fig. 17. Relative error in the strain energy and stress for different load cases when compared to the FEM reference solution. For the error measures see Eqs. (20) and (21), respectively.



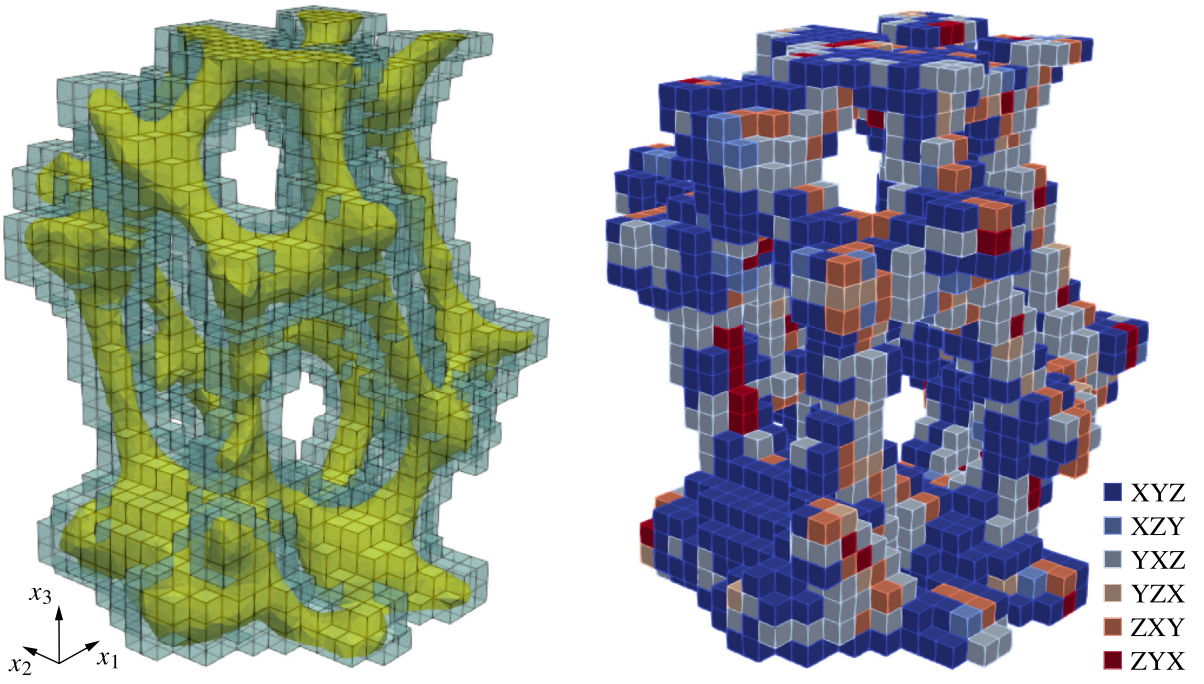
(a) Load case: tension (scaling factor: 0.09)

(b) Load case: compression (scaling factor: 1)

Fig. 18. Contour plots of the stress σ_{33} at the last load steps of the two simulated loading cases.

4.2.1. Reduction of integration points

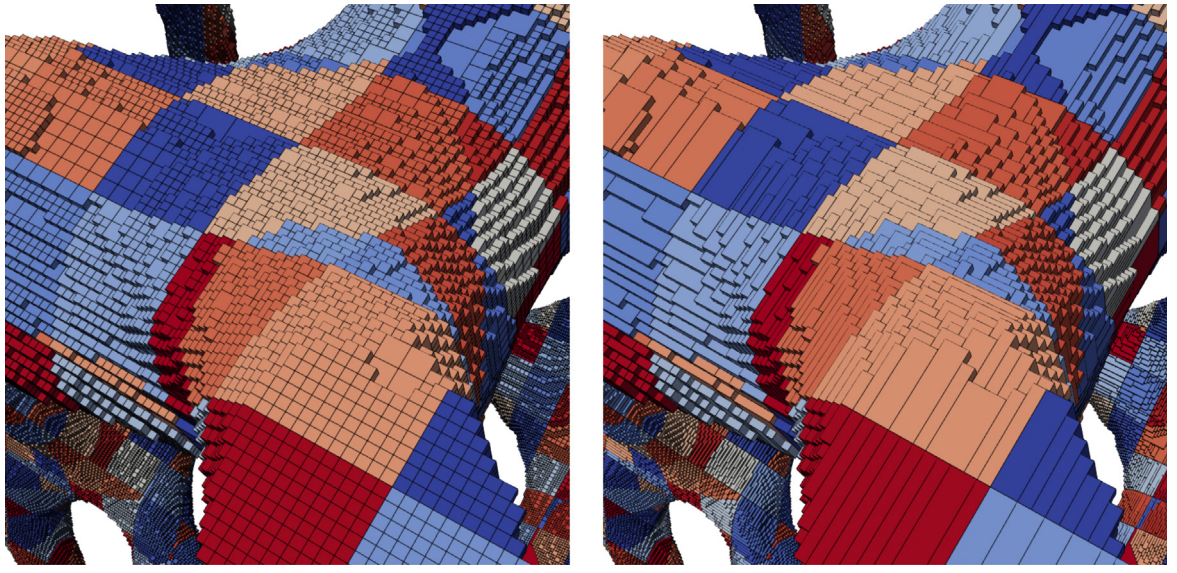
Note that due to the complexity of the given structure, it is not intuitive which composition of compression directions yields the fewest integration points. In Fig. 19(b), the cut cells are colored according to the compression setting that results in the smallest number of sub-cells. Individually choosing the best and worst compression settings in each cell, we observe that the compression ratios are quite close (see Table 2) as already seen for $\mathcal{R} > 2$ in Section 3.3. It is then also obvious, that any pre-selected compression direction will yield a λ_{IP}-value that lies within the bounds of $\lambda_{\text{IP}}^{\text{best}}$ and $\lambda_{\text{IP}}^{\text{worst}}$. Note that for such a complex structure, the compression rates are generally likely to be very similar, since a uniform compression direction may lead to sub-optimal results in some cells, while to optimal



(a) Discretization by finite cells

(b) Best compression direction for the OBR per cell.

Fig. 19. Problem setup of a foam discretized by 25×25×25 cells. Fictitious cells are removed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Physical sub-cells without compression

(b) Physical sub-cells with OBR compression

Fig. 20. Physical sub-cells without and with compression. The color coding indicates the different cells. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ones in others. Thus, for each setting, an average performance is obtained. In fact, choosing in each cell a random compression setting for the OBR yields very similar λ_{IP}-values. Of course, the compression can be enhanced by

Table 2
Comparison of compression rates for different settings of the OBR algorithm.

	Best	Worst	XYZ	XZY	YXZ	YZX	ZXY	ZYX	Rand.	Auto.
λ_{IP} [%] ($p = 2$)	45.08	41.51	43.87	43.86	43.94	43.90	43.66	43.74	43.87	44.92
λ_{IP} [%] ($p = 3$)	45.05	41.48	43.84	43.83	43.91	43.86	43.63	43.71	43.84	44.89

Table 3
Reduction of computational time (excluding the time required for setting up the LIM) when using the compressed LIM based on the autonomous OTD algorithm.

	$p = 2$		$p = 3$	
Integration schemes	Time	Reduct.	Time	Reduct.
Octree	1h 26 min 42 s	–	4h 48 min 26 s	–
Compressed octree	53 min 10 s	38.7%	2h 40 min 43 s	42.7%

the autonomous direction finding algorithm of Section 3.3, yielding even better results close to λ_{IP}^{best} in case of the current example.

When compared to the OTD without compression, merging the sub-cells by the autonomous OBR algorithm posed only 0.021% additional computational time for generating the LIM in `Matlab`. Similar to the example in Section 4.1, the reduced number of integration points can be re-used in several load steps, so that the relatively small time investment for the merged LIM becomes even more negligible. For the current example, 24 load steps were used, and the simulation time could be reduced by approximately 38.7% and 42.7% for $p = 2$ and $p = 3$, respectively, when using the integration points generated by the autonomous OTD algorithm (Table 3). Since the numerical integration accounts for the most computational effort, the reduction of integration points and computational time are very close to each other.

Remark to Fig. 19(b)**:** Note that in certain cells, multiple settings exist that lead to the same number of sub-cells. In Fig. 21, a histogram is depicted, visualizing the number of cells against the number of compression settings that yield the lowest (green) and highest (red) amount of sub-cells within the individual cells. The histogram can be understood as follows: For example, there are 860 cells, in which two of the six possible OBR settings resulted in the smallest amount of sub-cells, while for 445 cells, all six OBR settings yielded the same amount of sub-cells. In contrast, for 1845 cells there was only one setting that yielded the worst results (most sub-cells). Fig. 21 reveals that there are more ways to reach the best possible result than the worst results. Expressed quantitatively for the given example, among the six available OBR settings, on average, ∼3 settings yield best results in the individual cells, while only ∼1 settings lead to the worst.

4.2.2. Solution quality

In the previous sub-section, a significant reduction of integration points was presented when using the proposed integration scheme. In this sub-section, the simulation accuracy is investigated for two different compression strategies: Once, an arbitrarily chosen XYZ compression direction is used in each cell. In the second approach, the compression direction is chosen individually in each cell based on the autonomous direction finding algorithm discussed in Section 3.3. Note that these settings not only correspond to different compression rates (Table 2), but also to different orientations of the enlarged sub-cells. In Fig. 22(a), the strain energy is depicted for both of the investigated settings, as well as for the standard OTD. In fact, comparing the compressed octree to the standard non-compressed one (both in case of $p = 2$ and $p = 3$), it is evident that the stability and accuracy of the simulation is maintained when using the reduced integration points. Unlike in case of the example in Section 4.1, for the current problem, a reference solution via FEM is rather cumbersome to obtain due to the complex nature of the geometry. Thus, the results are compared *only* to the standard OTD-based integration scheme. In good agreement with Fig. 22(a), $e_{I\!I}$ in Fig. 22(b) depicts a marginal deviation from the original OTD. Here again, although an increasing integration error can be observed as $\bar{u}_3$ increases, it poses negligible loss of accuracy to the entire simulation. This is in good agreement with results obtained in Section 4.1. Finally, the deformed foam in two different load steps is depicted in Fig. 23.

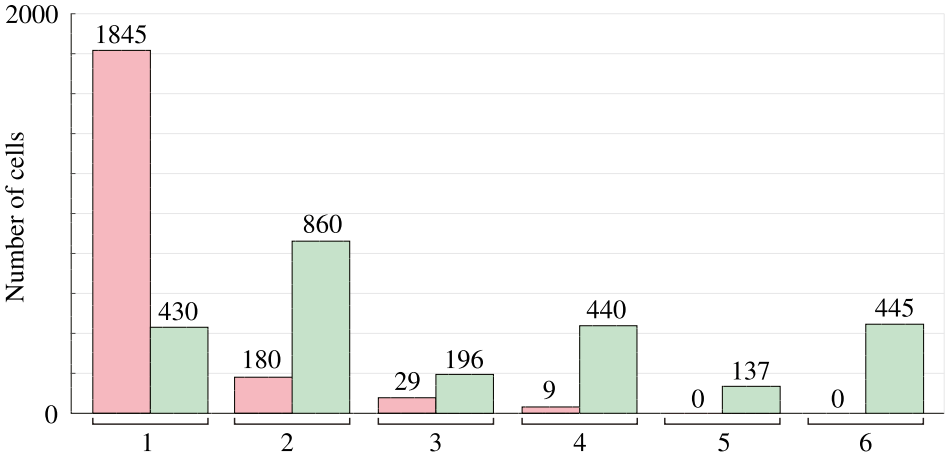
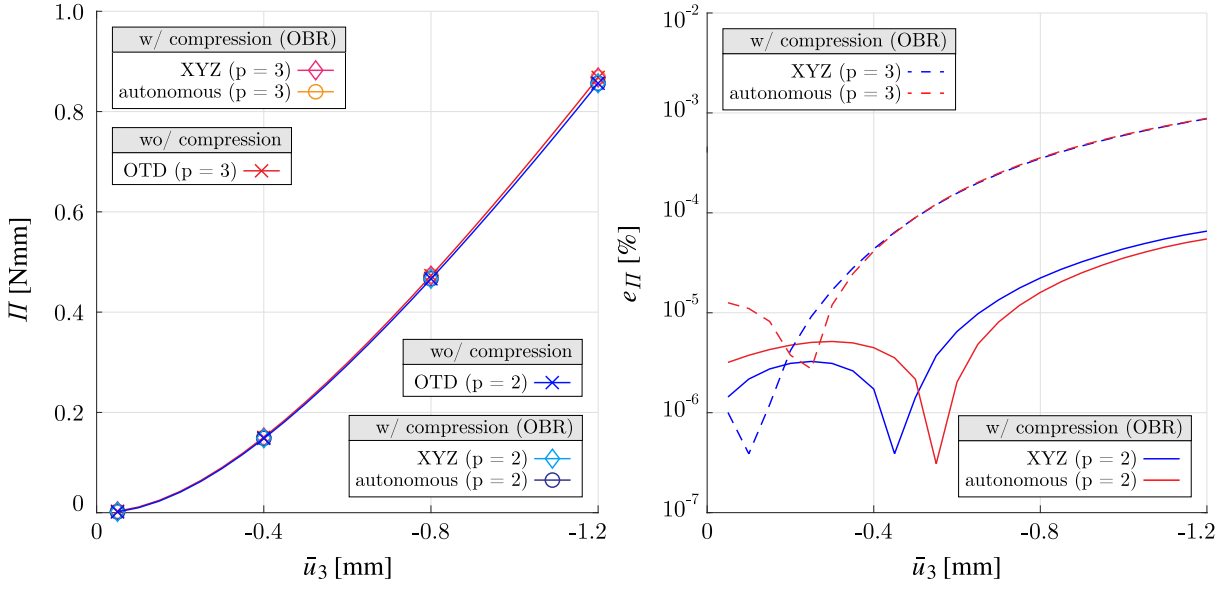


Fig. 21. Number of compression directions that lead to the fewest (green) and most (red) sub-cells. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Strain energy

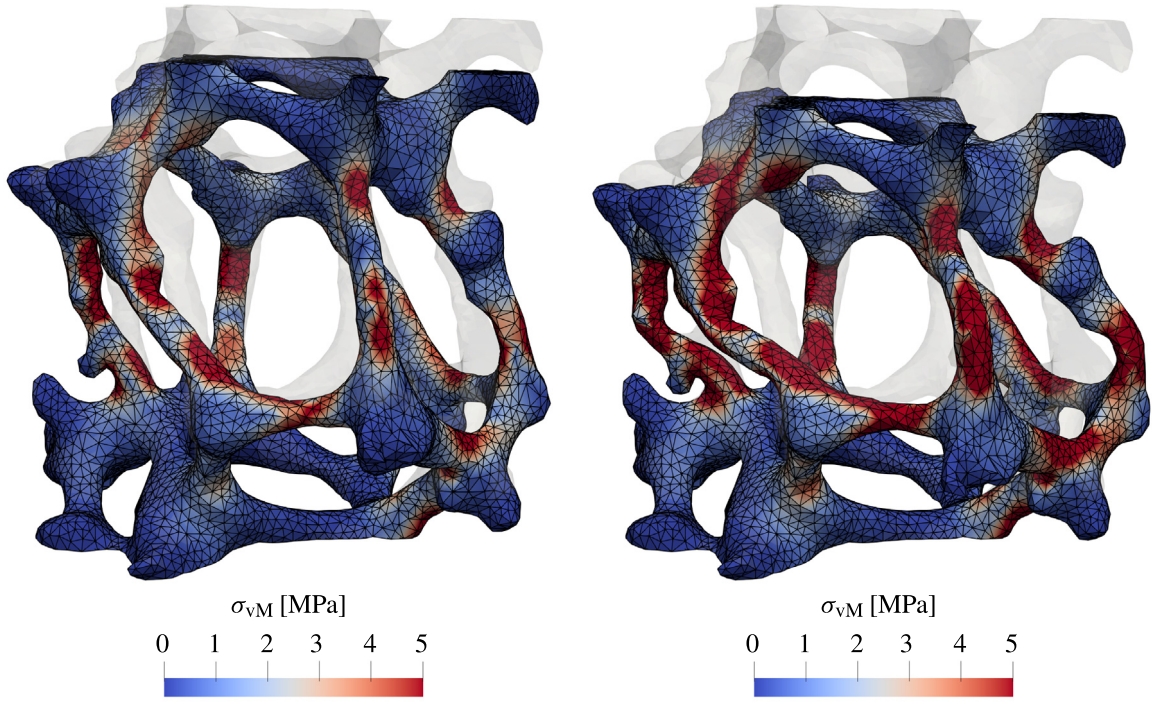
(b) Relative error in the strain energy

Fig. 22. Strain energy in different load steps using different polynomial degrees and integration schemes.

5. Conclusion

In the present work, an efficient approach was introduced for reducing the numerical effort of computing discontinuous integrals over 3D domains via octree-based integration meshes. Following our previous work [83], the proposed method is based on merging the physical sub-cells resulting from the octree-decomposition (OTD). For doing so, the original sub-cells from the OTD are transformed into an equivalent voxel representation to which different compression algorithms can be applied. Among the proposed methods, the OBR approach is the most effective (Section 3). The proposed integration scheme has numerous benefits:

1. Due to its modular nature, the compression step can be easily added as an intermediate step between the OTD and integration steps, without the need for major modifications in an already existing FCM code (Fig. 5).



(a) $\bar{u}_3 = -0.6\,\text{mm}$ (b) $\bar{u}_3 = -1.2\,\text{mm}$

Fig. 23. Von Mises stress of the foam at different load steps using $p = 2$. For visualization, the FCM results are projected onto the STL geometry.

2. Although the sub-cells are no longer cube-shaped after the compression, their regular shape is still maintained. Thus, a linear geometry mapping with a constant Jacobian still applies and the complexity of the integrand in Eq. (11) over the reduced set of sub-cells is not increased.
3. The compression is performed on an equivalent voxel domain and requires voxel-checking operations only. Consequently, it poses minimal computational overhead when deriving the local integration mesh (Section 3.2.2).
4. It is capable of reducing the number of the integration points by a significant amount, being about $\sim 40-50\%$ for the investigated examples.
5. Despite the significant reduction of the integration points, stability of the non-linear FCM analysis is still maintained (Section 4) and the loss of accuracy is negligible. In case of 3D linear elastic problems (not included in this paper), the exact same integration accuracy is expected as by the standard OTD, however, with significantly less computational costs.
6. The proposed method is especially useful in case of non-linear problems, where the compression has to be carried out only once for a given mesh to generate a reduced set of integration points, which can be then re-used in several load steps and iteration. Thus, the time investment for performing the compression is even more negligible when compared to the time saved during the numerical integration. Numerical examples in Section 4 show, that similar to reduction of integration points, the computation time of the simulation could be reduced by $\sim 40 - 50\%$.
7. Saving integration points becomes even more important when considering material models such as elasto-plasticity, where for each integration point a radial return algorithm has to be performed.

Declaration of competing interest

The authors declare that there are no competing interests.

Data availability

Data will be made available on request.

Acknowledgment

The authors thank the German Research Foundation (DFG) for its financial support under the Grant DU 1904/2-1, Project-nr. 423317638.

Appendix

A.1. Geometries in Section 3.3

In the following, the geometric properties of the examples used in Fig. 12 are given. In all three cases, the cubic domain is discretized by a single cell only.

Ellipsoid in Fig. 12(a): The cubic domain is spanned by the corners $X_{\min} = [-1,\ -1,\ -1]^{\mathrm{T}}$ and $X_{\max} = [1,\ 1,\ 1]^{\mathrm{T}}$. The intersecting ellipse is defined by the parameters

$$O = [0,\ 0.3,\ 1]^{\mathrm{T}}, \quad r = [0.6,\ 1.3,\ 1]^{\mathrm{T}}, \quad \varphi = [0,\ 0,\ 10]^{\mathrm{T}}, \tag{22}$$

where O_i, r_i and φ_i stand for the coordinates of its origin, semi-radii and rotation about the axis x_i, respectively.

Popcorn in Fig. 12(b): The cubic domain is spanned by the corners $X_{\min} = [-1.2,\ -0.65,\ -0.8]^{\mathrm{T}}$ and $X_{\max} = [-0.1,\ 0.45,\ 0.3]^{\mathrm{T}}$. For the implicit definition of popcorn flake, see Ref. [94], where following settings are used:

$$O = [0,\ 0,\ 0]^{\mathrm{T}}, \quad r_0 = 0.6, \quad \sigma = 0.2, \quad A = 2\,. \tag{23}$$

Multiple ellipsoids in Fig. 12(c): The cubic domain is spanned by the corners $X_{\min} = [-1,\ -1,\ -1]^{\mathrm{T}}$ and $X_{\max} = [1,\ 1,\ 1]^{\mathrm{T}}$. For the definition of the four intersecting ellipsoids, the following settings are used:

$$O_1 = [-0.25,\ -0.5,\ -0.85]^{\mathrm{T}}, \quad r_1 = [0.6,\ 0.75,\ 1.25]^{\mathrm{T}}, \quad \varphi_1 = [0,\ -20,\ 0]^{\mathrm{T}} \tag{24}$$

$$O_2 = [0.5,\ 0.75,\ -0.5]^{\mathrm{T}}, \quad r_2 = [0.5,\ 0.35,\ 1.25]^{\mathrm{T}}, \quad \varphi_2 = [10,\ 30,\ 0]^{\mathrm{T}} \tag{25}$$

$$O_3 = [0.75,\ -0.7,\ 0.6]^{\mathrm{T}}, \quad r_3 = [0.95,\ 0.65,\ 0.50]^{\mathrm{T}}, \quad \varphi_3 = [0,\ 20,\ 35]^{\mathrm{T}} \tag{26}$$

$$O_4 = [-0.5,\ 0.5,\ 0.8]^{\mathrm{T}}, \quad r_4 = [1,\ 0.40,\ 0.40]^{\mathrm{T}} \quad \varphi_4 = [0,\ 0,\ 20]^{\mathrm{T}} \tag{27}$$

References

[1] B. Szabó, A. Düster, E. Rank, The p-version of the finite element method, in: Encyclopedia of Computational Mechanics, 2004, chapter 5.

[2] A. Düster, E. Rank, B. Szabó, The p-version of the finite element and finite cell methods, in: Encyclopedia of Computational Mechanics, second ed., 2017, pp. 1–35.

[3] I. Ramière, P. Angot, M. Belliard, A fictitious domain approach with spread interface for elliptic problems with general boundary conditions, Comput. Methods Appl. Mech. Engrg. 196 (4–6) (2007) 766–781.

[4] J. Parvizian, A. Düster, E. Rank, Finite cell method – h- and p-extension for embedded domain problems in solid mechanics, Comput. Mech. 41 (2007) 121–133.

[5] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, Comput. Methods Appl. Mech. Engrg. 197 (2008) 3768–3782.

[6] M. Dauge, A. Düster, E. Rank, Theoretical and numerical investigation of the finite cell method, J. Sci. Comput. 65 (3) (2015) 1039–1064.

[7] D. Schillinger, M. Ruess, The finite cell method: A review in the context of higher-order structural analysis of CAD and image-based geometric models, Arch. Comput. Methods Eng. 22 (3) (2014) 391–455.

[8] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method, Comput. Methods Appl. Mech. Engrg. 199 (41–44) (2010) 2680–2686.

[9] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method, Appl. Numer. Math. 62 (4) (2012) 328–341.

[10] E. Burman, P. Hansbo, Fictitious domain methods using cut elements: III. A stabilized Nitsche method for Stokes' problem, ESAIM: Math. Model. Numer. Anal. 48 (3) (2014) 859–874.

[11] T.-P. Fries, T. Belytschko, The extended/generalized finite element method: An overview of the method and its applications, Internat. J. Numer. Methods Engrg. (2010) 253–304.
[12] E. Nadal, J.J. Ródenas, J. Albelda, M. Tur, J.E. Tarancón, F.J. Fuenmayor, Efficient finite element methodology based on Cartesian grids: Application to structural shape optimization, Abstr. Appl. Anal. 2013 (2013) 1–19.
[13] D. Muñoz, J. Albelda, J.J. Ródenas, E. Nadal, Improvement in 3D topology optimization with h -adaptive refinement using the Cartesian grid finite element method, Internat. J. Numer. Methods Engrg. (2021).
[14] M.J. García-Ruíz, G.P. Steven, Fixed grid finite elements in elasticity problems, Eng. Comput. 16 (2) (1999) 145–164.
[15] A. Hansbo, P. Hansbo, An unfitted finite element method, based on Nitsche's method, for elliptic interface problems, Comput. Methods Appl. Mech. Engrg. 191 (47–48) (2002) 5537–5552.
[16] D. Schillinger, E. Rank, An unfitted hp-adaptive finite element method based on hierarchical B-splines for interface problems of complex geometry, Comput. Methods Appl. Mech. Engrg. 200 (47–48) (2011) 3358–3380.
[17] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (2) (1999) 509–534.
[18] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[19] D. Schillinger, M. Ruess, N. Zander, Y. Bazilevs, A. Düster, E. Rank, Small and large deformation analysis with the p- and B-spline versions of the finite cell method, Comput. Mech. 50 (2012) 445–478.
[20] W. Garhuom, S. Hubrich, L. Radtke, A. Düster, A remeshing strategy for large deformations in the finite cell method, Comput. Math. Appl. (2020).
[21] W. Garhuom, K. Usman, A. Düster, An eigenvalue stabilization technique to increase the robustness of the finite cell method for finite strain problems, Comput. Mech. (2022).
[22] A. Abedian, J. Parvizian, A. Düster, E. Rank, The finite cell method for the J_2 flow theory of plasticity, Finite Elem. Anal. Des. 69 (2013) 37–47.
[23] A. Taghipour, J. Parvizian, S. Heinze, A. Düster, The finite cell method for nearly incompressible finite strain plasticity problems with complex geometries, Comput. Math. Appl. 75 (2018) 3298–3316.
[24] S. Hubrich, A. Düster, Numerical integration for nonlinear problems of the finite cell method using an adaptive scheme based on moment fitting, Comput. Math. Appl. 77 (2019) 1983–1997.
[25] S. Heinze, T. Bleistein, A. Düster, S. Diebels, A. Jung, Experimental and numerical investigation of single pores for identification of effective metal foams properties, ZAMM-Z. Angew. Math. Und Mech. 98 (2018) 682–695.
[26] S. Heinze, M. Joulaian, A. Düster, Numerical homogenization of hybrid metal foams using the finite cell method, Comput. Math. Appl. 70 (2015) 1501–1517.
[27] M. Elhaddad, N. Zander, S. Kollmannsberger, A. Shadavakhsh, V. Nübel, E. Rank, Finite cell method: High-order structural dynamics for complex geometries, Int. J. Struct. Stab. Dyn. 15 (7) (2015) 1540018.
[28] M. Joulaian, S. Duczek, U. Gabbert, A. Düster, Finite and spectral cell method for wave propagation in heterogeneous materials, Comput. Mech. 54 (2014) 661–675.
[29] N. Zander, S. Kollmannsberger, M. Ruess, Z. Yosibash, E. Rank, The finite cell method for linear thermoelasticity, Comput. Math. Appl. 64 (11) (2012) 3527–3541.
[30] Z. Yang, S. Kollmannsberger, A. Düster, M. Ruess, E.G. Garcia, R. Burgkart, E. Rank, Non-standard bone simulation: Interactive numerical analysis by computational steering, Comput. Vis. Sci. 14 (5) (2011) 207–216.
[31] M. Ruess, D. Tal, N. Trabelsi, Z. Yosibash, E. Rank, The finite cell method for bone simulations: Verification and validation, Biomech. Model. Mechanobiol. 11 (2012) 425–437.
[32] C.V. Verhoosel, G.J. van Zwieten, B. Rietbergen, R. de Borst, Image-based goal-oriented adaptive isogeometric analysis with application to the micro-mechanical modeling of trabecular bone, Comput. Methods Appl. Mech. Engrg. 284 (2015) 138–164.
[33] S. Duczek, S. Liefold, U. Gabbert, The finite and spectral cell methods for smart structure applications: Transient analysis, Acta Mech. 226 (3) (2014) 845–869.
[34] A. Düster, H.-G. Sehlhorst, E. Rank, Numerical homogenization of heterogeneous and cellular materials utilizing the finite cell method, Comput. Mech. 50 (4) (2012) 413–431.
[35] S. Heinze, M. Joulaian, A. Düster, Numerical homogenization of hybrid metal foams using the finite cell method, Comput. Math. Appl. 70 (7) (2015) 1501–1517.
[36] S. Nagaraja, M. Elhaddad, M. Ambati, S. Kollmannsberger, L. De Lorenzis, E. Rank, Phase-field modeling of brittle fracture with multi-level hp-FEM and the finite cell method, Comput. Mech. 63 (6) (2018) 1283–1300.
[37] J. Parvizian, A. Düster, E. Rank, Topology optimization using the finite cell method, Opt. Eng. 13 (1) (2011) 57–78.
[38] H. Spartali, Application of Finite Cell Method for Contact Mechanics Problems (Master s Thesis), Ruhr University Bochum, 2018.
[39] J. Wang, L. Quan, K. Tang, A prediction method based on the voxel model and the finite cell method for cutting force-induced deformation in the five-axis milling process, Comput. Methods Appl. Mech. Engrg. 367 (2020) 113110.
[40] L. Kudela, S. Kollmannsberger, U. Almac, E. Rank, Direct structural analysis of domains defined by point clouds, Comput. Methods Appl. Mech. Engrg. 358 (2020) 112581.
[41] B. Wassermann, N. Korshunova, S. Kollmannsberger, E. Rank, G. Elber, Finite cell method for functionally graded materials based on V-models and homogenized microstructures, Adv. Model. Simul. Eng. Sci. (2020).
[42] S. Duczek, Higher order finite elements and the fictitious domain concept for wave propagation analysis (Ph.D. thesis), Universitätsbibl., Otto von Guericke University Magdeburg, 2014.
[43] S. Duczek, M. Joulaian, A. Düster, U. Gabbert, Numerical analysis of Lamb waves using the finite and spectral cell methods, Internat. J. Numer. Methods Engrg. 99 (1) (2014) 26–53.

[44] F. Mossaiby, M. Joulaian, A. Düster, The spectral cell method for wave propagation in heterogeneous materials simulated on multiple GPUs and CPUs, Comput. Mech. 63 (5) (2018) 805–819.
[45] M. Joulaian, A. Düster, Local enrichment of the finite cell method for problems with material interfaces, Comput. Mech. 52 (4) (2013) 741–762.
[46] M. Joulaian, The hierarchical finite cell method for problems in structural mechanics (Ph.D. thesis), Hamburg Technical University, 2017.
[47] M. Ruess, D. Schillinger, Y. Bazilevs, V. Varduhn, E. Rank, Weakly enforced essential boundary conditions for NURBS-embedded and trimmed NURBS geometries on the basis of the finite cell method, Internat. J. Numer. Methods Engrg. 95 (10) (2013) 811–846.
[48] F. de Prenter, C.V. Verhoosel, E.H. van Brummelen, Preconditioning immersed isogeometric finite element methods with application to flow problems, Comput. Methods Appl. Mech. Engrg. 348 (2019) 604–631.
[49] D. Schillinger, L. Dedè, M.A. Scott, J.A. Evans, M.J. Borden, E. Rank, T.J.R. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Comput. Methods Appl. Mech. Engrg. 249–252 (2012) 116–150.
[50] S. Kollmannsberger, D. D'Angella, E. Rank, W. Garhuom, S. Hubrich., A. Düster, P.D. Stolfo, A. Schröder, Spline- and hp -basis functions of higher differentiability in the finite cell method, GAMM-Mitteilungen (2019).
[51] S. Duczek, U. Gabbert, The finite cell method for polygonal meshes: Poly-FCM, Comput. Mech. 58 (4) (2016) 587–618.
[52] M. Petö, F. Duvigneau, D. Juhre, S. Eisenträger, Enhanced numerical integration scheme based on image compression techniques: Application to rational polygonal interpolants, Arch. Appl. Mech. 91 (2) (2020) 753–775.
[53] L. Nguyen, S. Stoter, T. Baum, J. Kirschke, M. Ruess, Z. Yosibash, D. Schillinger, Phase-field boundary conditions for the voxel finite cell method: Surface-free stress analysis of CT-based bone structures, Int. J. Numer. Methods Biomed. Eng. 33 (12) (2017).
[54] F. de Prenter, C.V. Verhoosel, G.J. van Zwieten, E.H. van Brummelen, Condition number analysis and preconditioning of the finite cell method, Comput. Methods Appl. Mech. Engrg. 316 (2017) 297–327.
[55] S. Duczek, F. Duvigneau, U. Gabbert, The finite cell method for tetrahedral meshes, Finite Elem. Anal. Des. 121 (2016) 18–32.
[56] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, M.-C. Hsu, The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries, Comput. & Fluids 141 (2016) 135–154.
[57] G. Legrain, N. Moës, Adaptive anisotropic integration scheme for high-order fictitious domain methods: Application to thin structures, Internat. J. Numer. Methods Engrg. 114 (8) (2018) 882–904.
[58] K.W. Cheng, T.-P. Fries, Higher-order XFEM for curved strong and weak discontinuities, Internat. J. Numer. Methods Engrg. (2009).
[59] T.-P. Fries, S. Omerović, Higher-order accurate integration of implicit geometries, Internat. J. Numer. Methods Engrg. 106 (5) (2015) 323–371.
[60] L. Kudela, N. Zander, T. Bog, S. Kollmannsberger, E. Rank, Efficient and accurate numerical quadrature for immersed boundary methods, Adv. Model. Simul. Eng. Sci. 2 (1) (2015).
[61] L. Kudela, N. Zander, S. Kollmannsberger, E. Rank, Smart octrees: Accurately integrating discontinuous functions in 3D, Comput. Methods Appl. Mech. Engrg. 306 (2016) 406–426.
[62] S.E. Mousavi, H. Xiao, N. Sukumar, Generalized Gaussian quadrature rules on arbitrary polygons, Internat. J. Numer. Methods Engrg. (2009).
[63] H. Xiao, Z. Gimbutas, A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions, Comput. Math. Appl. 59 (2) (2010) 663–676.
[64] B. Müller, F. Kummer, M. Oberlack, Highly accurate surface and volume integration on implicit domains by means of moment-fitting, Internat. J. Numer. Methods Engrg. 96 (8) (2013) 512–528.
[65] M. Joulaian, S. Hubrich, A. Düster, Numerical integration of discontinuities on arbitrary domains based on moment fitting, Comput. Mech. 57 (6) (2016) 979–999.
[66] S.E. Mousavi, N. Sukumar, Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons, Comput. Mech. 47 (5) (2010) 535–554.
[67] Y. Sudhakar, W.A. Wall, Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods, Comput. Methods Appl. Mech. Engrg. 258 (2013) 39–54.
[68] H.-G. Bui, D. Schillinger, G. Meschke, Efficient cut-cell quadrature based on moment fitting for materially nonlinear analysis, Comput. Methods Appl. Mech. Engrg. 366 (2020) 113050.
[69] G. Legrain, Non-negative moment fitting quadrature rules for fictitious domain methods, Comput. Math. Appl. 99 (2021) 270–291.
[70] A. Düster, O. Allix, Selective enrichment of moment fitting and application to cut finite elements and cells, Comput. Mech. 65 (2) (2019) 429–450.
[71] G. Ventura, On the elimination of quadrature subcells for discontinuous functions in the extended finite-element method, Internat. J. Numer. Methods Engrg. 66 (5) (2006) 761–795.
[72] G. Ventura, E. Benvenuti, Equivalent polynomials for quadrature in heaviside function enriched elements, Internat. J. Numer. Methods Engrg. 102 (3–4) (2014) 688–710.
[73] A. Abedian, A. Düster, Equivalent Legendre polynomials: Numerical integration of discontinuous functions in the finite element methods, Comput. Methods Appl. Mech. Engrg. 343 (2019) 690–720.
[74] G. Dasgupta, Integration within polygonal finite elements, J. Aerosp. Eng. 16 (1) (2003) 9–18.
[75] X.-W. Gao, The radial integration method for evaluation of domain integrals with boundary-only discretization, Eng. Anal. Bound. Elem. 26 (10) (2002) 905–916.
[76] Y. Sudhakar, J.P. Moitinho de Almeida, W.A. Wall, An accurate, robust, and easy-to-implement method for integration over arbitrary polyhedra: Application to embedded interface methods, J. Comput. Phys. 273 (2014) 393–415.

[77] S. Duczek, U. Gabbert, Efficient integration method for fictitious domain approaches, Comput. Mech. 56 (4) (2015) 725–738.
[78] A. Abedian, J. Parvizian, A. Düster, H. Khademyzadeh, E. Rank, Performance of different integration schemes in facing discontinuities in the finite cell method, Int. J. Comput. Methods 10 (03) (2013) 1350002.
[79] A. Taghipour, J. Parvizian, S. Heinze, A. Düster, The finite cell method for nearly incompressible finite strain plasticity problems with complex geometries, Comput. Math. Appl. 75 (9) (2018) 3298–3316.
[80] P. Zakian, M. Nadi, M. Tohidi, Finite cell method for detection of flaws in plate structures using dynamic responses, Structures 34 (2021) 327–338.
[81] A. Abedian, J. Parvizian, A. Düster, E. Rank, Finite cell method compared to h-version finite element method for elasto-plastic problems, Appl. Math. Mech. 35 (10) (2014) 1239–1248.
[82] A. Abedian, A. Düster, An extension of the finite cell method using Boolean operations, Comput. Mech. 59 (5) (2017) 877–886.
[83] M. Petö, F. Duvigneau, S. Eisenträger, Enhanced numerical integration scheme based on image-compression techniques: Application to fictitious domain methods, Adv. Model. Simul. Eng. Sci. 7 (1) (2020).
[84] K.-J. Bathe, Finite Element Procedures, Prentice Hall, 1996.
[85] P. Wriggers, Nonlinear Finite-Element-Methods, Springer-Verlag, 2008.
[86] P.G. Ciarlet, Mathematical Elasticity, Vol. 1, Elsevier, 1988.
[87] B.A. Szabó, I. Babuška, Finite Element Analysis, John Wiley & Sons, 1991.
[88] B.A. Szabó, A. Düster, E. Rank, The p-version of the finite element method, in: E. Stein, R. de Borst, T.J.R. Hughes (Eds.), in: Encyclopedia of Computational Mechanics, vol. 1, John Wiley & Sons, 2004, pp. 119–139, chapter 5.
[89] D. Salomon, G. Motta, Handbook of Data Compression, Springer London, 2010.
[90] T. Suk, C. Höschl, J. Flusser, Rectangular decomposition of binary images, in: Advanced Concepts for Intelligent Vision Systems, Springer Berlin Heidelberg, 2012, pp. 213–224.
[91] I.M. Spiliotis, B.G. Mertzios, Real-time computation of two-dimensional moments on binary images using image block representation, IEEE Trans. Image Process. 7 (11) (1998) 1609–1615.
[92] N. Zander, T. Bog, M. Elhaddad, R. Espinoza, H. Hu, A.F. Joly, C. Wu, P. Zerbe, A. Düster, S. Kollmannsberger, J. Parvizian, M. Ruess, D. Schillinger, E. Rank, FCMLab: A finite cell research toolbox for MATLAB, Adv. Eng. Softw. 74 (2014) 49–63.
[93] S. Heinze, T. Bleistein, A. Düster, S. Diebels, A. Jung, Experimental and numerical investigation of single pores for identification of effective metal foams properties, ZAMM - J. Appl. Math. Mech. / Z. Angew. Math. Und Mech. 98 (5) (2018) 682–695.
[94] I.-L. Chern, Y.-C. Shu, A coupling interface method for elliptic interface problems, J. Comput. Phys. 225 (2) (2007) 2138–2174.