Petö *et al. Adv. Model. and Simul. in Eng. Sci.(2020)7:21*
https://doi.org/10.1186/s40323-020-00157-2

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Enhanced numerical integration scheme based on image-compression techniques: application to fictitious domain methods

Márton Petö[1*], Fabian Duvigneau[1] and Sascha Eisenträger[2]

*Correspondence: marton.petoe@ovgu.de
[1]Institute of Mechanics, Otto von Guericke University, Magdeburg, Germany
Full list of author information is available at the end of the article

Abstract

In the present work, we propose a new approach, the so-called compressed adaptive integration scheme (C-AIS), for the computation of the stiffness and mass matrices in fictitious domain methods requiring the integration of discontinuous functions. The novel approach extends the conventional quadtree-decomposition-based adaptive integration scheme (AIS) by an additional step, in which established image-compression techniques are exploited to decrease the number of integration sub-cells. The benefits of the C-AIS are manifold: First, the compression of the sub-cells inevitably leads to significant savings in terms of computational time required by the numerical integration. Second, the compression procedure, which is executed directly after the quadtree-decomposition algorithm, can be easily included in existing codes. Third, if applied to polynomial integrands, the C-AIS yields exactly the same accuracy as the conventional AIS. Finally, the fourth advantage is seen in the fact that the C-AIS can readily be combined with other approaches seeking a reduction of the number of integration points such as the Boolean-FCM. The efficiency of the C-AIS approach is presented in the context of the FCM based on Cartesian meshes applied to problems of linear elastostatics and modal analysis, while it is also suitable for the quadrature in other fictitious domain approaches, e.g., CutFEM and cgFEM.

Introduction

In engineering and physics one often deals with *continuous field problems* governed by partial differential equations (PDEs). These problems are often boundary value problems (BVP), where the primary variables have to satisfy a set of PDEs, while on the boundary of the domain of interest specific values of the field variables, formulated in form of boundary conditions (BCs), are prescribed [1]. Since these problems cannot be solved analytically in most of the cases, they have to be solved numerically.

Although the traditional FEM [2,3] has proven its power throughout the years for solving BVPs, various extensions were developed in order to obtain an even higher accuracy, a lower computational time and/or a wider field of application. Such an extension is the *fictitious domain approach* (FDA) [4–7], also known as *unfitted finite element method* [8]

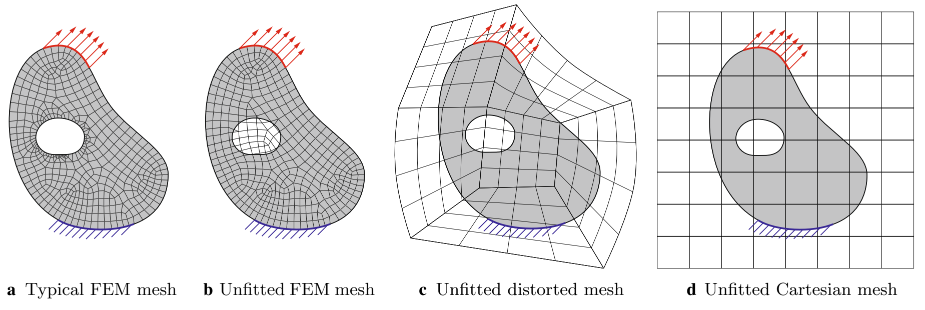


Fig. 1 Examples of typical discretizations in the framework of the FEM (**a**) and fictitious domain methods (**b**–**d**) based on unfitted meshes

or *immersed boundary method* [9], where the BVP is solved on an unfitted computational mesh independent from the original domain while maintaining the same robustness and accuracy as the FEM [10]. Such a formulation can be highly beneficial compared to the conventional FEM when it comes to automatic mesh generation, moving boundaries and simulation of complex media involving holes, inclusions and cracks. There are several possibilities of constructing an unfitted mesh, see Fig. 1 for details. One idea is to reuse an already existing FEM mesh and inserts extra features, such as voids as depicted in Fig. 1b and exemplarily discussed in [11]. Other approaches include embedding the original domain into a larger domain, which can be discretized by distorted or regular elements depending on the shape of the embedding domain, as shown in Fig. 1c, d, respectively. There is a vast variety of different methods that are based on the fundamental idea of the FDA, differing in their assumption regarding the topology of the unfitted mesh, the type of shape functions and their typical areas of application. However, it is worth to emphasize, that almost all of the different methods can adapt most of the features of the other ones. Well known methods based on the FDA are for example the *extended finite element method* (XFEM) [12–15] and *generalized finite element method* (GFEM) [15–18]; both of which are based on the *partition of unity method* (PUM) [19,20] enabling a sophisticated tool for modelling problems with singularities, kinks, jumps or other non-smooth features. An in-depth mathematical foundation of the FDA and of the implementation of boundary conditions over the unfitted meshes can be found in the literature of the *cut finite element method* (CutFEM) [21–24], where often triangular elements are used. While the above approaches do not require the computational mesh to be rectilinear, the *fixed grid finite element method* (FGFEM) [25–27] assumes the mesh to be strictly Cartesian and uses a less accurate, yet very fast, integration scheme based on pre-computation, making the FGFEM especially suitable for topology optimization [28,29]. Similar to the FGFEM, the *Cartesian grid finite element method* (cg-FEM) [30] is also restricted to rectilinear elements, however, it is combined with a hierarchic data-structure for adaptive h-refinement steered either by the curvature of the boundary or by appropriate error estimators. Finally, we mention the *finite cell method* (FCM) [31–35], which is a combination of high-order shape functions known from p-FEM resulting in an exponential convergence [33] and unfitted (often Cartesian) meshes enabling a fast discretization of the computational domain [10,31]. Please note that while the FCM can also be straightforwardly used in combination with distorted elements our focus in this article is on Cartesian

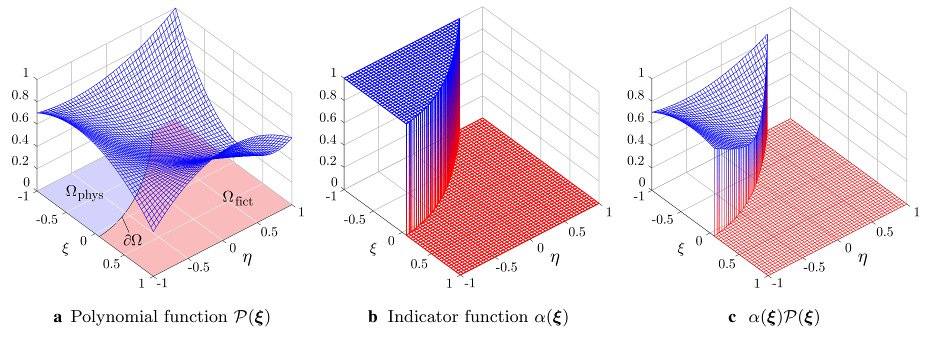


Fig. 2 A two-dimensional example of a discontinuous integral in the FCM, where the polynomial integrand $\mathcal{P}(\xi, \eta) = 0.7N_1 + 0.4N_2 + 0.6N_3 + 1N_4 + 0.8N_5 + 0.8N_6 + 0.2N_7 + 0.7N_8 + 0.4N_9$ is defined by a linear combination of the 3rd-order Lagrangian shape functions based on an equidistant nodal distribution

meshes. A detailed discussion of the FCM, which is the basis of our implementation, will be provided in "The finite cell method for linear elastostatics" section.

However, regardless of the governing equations of the problem under consideration and of the chosen FDA, all of the above methods give rise to a non boundary-conforming mesh by definition, resulting in elements cut by the boundary. Consequently, compared to the FEM, the difficulty in these methods is shifted, on the one hand, to the implementation of the boundary conditions [4–6,21,22,36–38], and on the other hand, to the computation of the cell-matrices. In this contribution, our focus is on the latter issue. A detailed discussion of different approaches that are currently used for the integration of the element matrices is provided in "Review of numerical integration methods: discontinuous integrands" section.

Review of numerical integration methods: discontinuous integrands

In order to illustrate the problem regarding the computation of the element matrices in the FDA-based methods, we consider a continuous polynomial function $\mathcal{P}(\boldsymbol{\xi})$ over a quadrilateral element $\Omega^{(e)} \in [-1, 1] \times [-1, 1]$ (Fig. 2a), which is divided by the boundary $\partial\Omega$ intersecting the element into the arbitrary-shaped disjoint physical Ω_{phys} and fictitious Ω_{fict} sub-domains. Generally in the FDA, one is interested in the integral of $\mathcal{P}(\boldsymbol{\xi})$[1] only over Ω_{phys}, however, the integration itself should be performed over the entire domain $\Omega^{(e)}$. For eliminating the contribution of the integral over Ω_{fict}, the indicator function $\alpha(\boldsymbol{\xi})$ is introduced, which is essentially a step function, having the value 1 in Ω_{phys} and the value 0 in Ω_{fict} (Fig. 2b). Then, the desired integral value I can be computed according to Eq. (1) [39], where the integrand on the right hand side is discontinuous (Fig. 2c) due to α.

$$I = \int\limits_{\Omega_{\mathrm{phys}}} \mathcal{P}(\boldsymbol{\xi})\,\mathrm{d}\Omega = \int\limits_{\Omega_{\mathrm{e}}} \alpha(\boldsymbol{\xi})\mathcal{P}(\boldsymbol{\xi})\,\mathrm{d}\Omega \tag{1}$$

The numerical computation of discontinuous integrals is an interesting topic with numerous applications which is still actively researched to all methods based on unfitted meshes. In the following, we briefly discuss the main approaches that are currently used

[1]Note that the integrand is not necessarily a polynomial function. This is only true if the employed shape functions are polynomials and if the geometry mapping is affine. This is, e.g., the case for the FCM in conjunction with Cartesian meshes.

in FDAs. Further details on these approaches can be found in Refs. [33,40,41]. Note that the discussed methods may originate in a specific field, nonetheless, they are generally applicable.

Pre-computation

If the computational mesh consists of identical elements, certain parts of the integrals can be pre-computed and reused while integrating over the individual cut elements. Both of the following two approaches assume a Cartesian mesh (see Fig. 1d).

It is a widely used practice in the FGFEM [25,26,29], that the term $\int_{\Omega_\mathrm{e}} \mathcal{P}(\boldsymbol{\xi})\ \mathrm{d}\Omega$ on the right hand side of Eq. (1) is assumed to be the same for each element regardless of the topology of discontinuity and therefore, it can be easily pre-computed. Then, for the computation of the discontinuous integral, the pre-computed term can be reused for each cut element while scaled accordingly by the ratio $A_\mathrm{phys}/A^{(\mathrm{e})}$, where A_phys and $A^{(\mathrm{e})}$ are the areas of the domains Ω_phys and $\Omega^{(\mathrm{e})}$, respectively. Clearly, such a basic computation of the discontinuous integral is related to serious errors, since it does not take the shape and position of Ω_phys into account, but only its area. Being aware of this, Maan et al. [26] used some additional equations based on the topology of the cut elements in order to determine a sufficient mesh density leading to smaller integration errors.

Another integration scheme based on the pre-computation approach was proposed by Yang et al. [42–44], where one is interested in solving an integral given in Eq. (2). At this point only a generic discussion of their approach is given, independent of the area of application.[2]

$$I = \int_{\Omega^{(\mathrm{e})}} \beta(\boldsymbol{\xi})\mathcal{P}(\boldsymbol{\xi})\ \mathrm{d}\Omega \tag{2}$$

Here, $\Omega^{(\mathrm{e})}$ is assumed to be composed of uniformly distributed rectangular regions as exemplarily depicted in Fig. 3a. In Eq. (2), β is a discontinuous function which is piece-wise constant over the individual sub-regions, cf. Fig. 3b. Assuming that the distribution of sub-domains is the same for all integration domains $\Omega^{(\mathrm{e})}$, the integral of $\mathcal{P}$ can be pre-computed over the individual domains. Then, the integral over a given element (see Fig. 3c) is computed by summing the pre-computed integrals scaled by the corresponding value of the function β, being constant for each sub-domain.

Local integration mesh

Another idea is to introduce a *local integration mesh* (LIM) in the cut cells which serves integration purposes only and does not increase the *degrees of freedom* (DOF) of the global system. Then, the integral over the entire cell $I^{(c)}$ is obtained by summing the integrals $I^{(\mathrm{sc})}$ over the n_sc integration sub-cells[3] constituting the LIM, while the sub-cells may or may not conform to the discontinuity within the cell.

$$I^{(\mathrm{c})} = \sum_{\mathrm{sc}=1}^{n_\mathrm{sc}} I^{(\mathrm{sc})} \tag{3}$$

[2] In the original contribution [42–44], the proposed integration scheme is discussed in the context of FCM applied to 3D bio-medical problems. Consequently, the computational domain is described by voxelated data, often obtained by medical-imaging such as computed tomography. Assuming constant linear elastic and isotropic material properties for the individual voxels, the stiffness and mass matrices can be pre-computed for the $n_x \times n_y \times n_z$ voxels within each element $\Omega^{(\mathrm{e})}$ using the two Lamé constants λ and μ.

[3] In the remainder, we refer to the integration sub-cells simply as sub-cells.

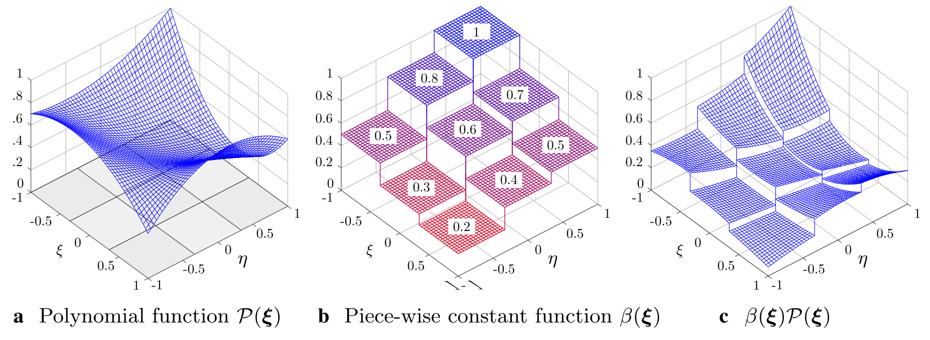


Fig. 3 Example for the discontinuous integrals in the approach by Yang et al. [42–44]. For the definition of $\mathcal{P}$ see Fig. 2

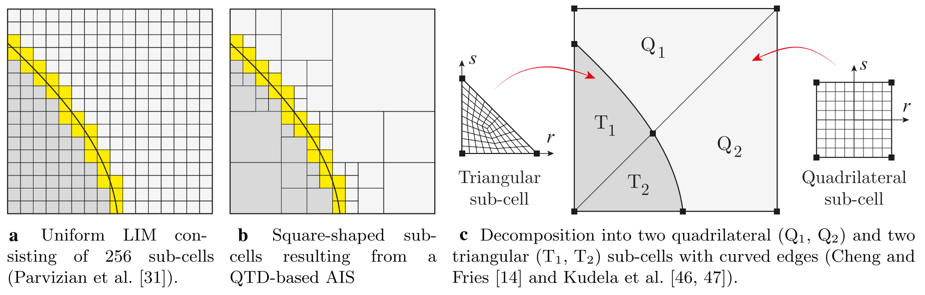

a Uniform LIM consisting of 256 sub-cells (Parvizian et al. [31]).
b Square-shaped sub-cells resulting from a QTD-based AIS
c Decomposition into two quadrilateral (Q_1, Q_2) and two triangular (T_1, T_2) sub-cells with curved edges (Cheng and Fries [14] and Kudela et al. [46, 47]).
Triangular sub-cell
Quadrilateral sub-cell
Q_1
Q_2
T_1
T_2
s
r

Fig. 4 Different ways of creating a local integration mesh for a two-dimensional cut cell. For sake of simplicity, the local coordinate system is not shown

Non-boundary-conforming LIM

In terms of a non-conforming LIM, Parvizian et al. [31] proposed a basic implementation for Eq. (3) by distributing $n \times n$ sub-cells of uniform size in the cell, over which a low order integration, such as the *trapezoidal rule* can be used (Fig. 4a). A more sophisticated approach is the *adaptive integration scheme* (AIS), which generates an LIM by successively creating smaller and smaller sub-cells in the vicinity of the discontinuity by using space-partitioning techniques such as the quadtree-decomposition (QTD) [32], yielding an integration mesh, exclusively consisting of square-shaped sub-cells (Fig. 4b). The B-FCM approach by Abedian et al. [45] is very similar to the QTD-based AIS when applied to complex regions, however, due to its improved strategy based on Boolean operations and area-calculations of Ω_{phys} and Ω_{fict}, it requires significantly fewer IPs while maintaining the same accuracy (see "Combination with the B-FCM" section).

Boundary-conforming LIM

A more accurate resolution of the discontinuity is possible using boundary-conforming sub-cells. Such an approach is the *adaptive anisotropic integration scheme* proposed by Legrain and Moës [46], generating a mesh of distorted sub-cells with increased density and decreased size in the vicinity of the discontinuity. A significantly smaller number of sub-cells is required when using a *Delaunay triangulation* to create an integration mesh that consists of triangular sub-cells with straight edges, as it was done by Nadal et al. [30]. They also note that for higher order elements also a higher order boundary representation is required for an optimal convergence rate of the simulation. Therefore, the need for sub-cells with curved edges enabling a higher approximation accuracy of the boundary emerges. In the works by Cheng and Fries [14] and Fries and Omerović [47], the discontinuity $\partial\Omega$ is assumed to be given in an implicit form by the iso-contour $\phi(\boldsymbol{x}) = 0$ of the scalar-valued level-set function and for the mapping of the sub-cells Lagrangian elements with enriched nodes are used [48], enabling a more accurate representation of the boundary. While the approach discussed in Ref. [14] is restricted to two-dimensional problems (Fig. 4c), an extension of the basic formulation to three-dimensions is proposed in Ref. [47]. Kudela et al. [49,50] follow the two-dimensional approach of Ref. [14], however, they assume a parametric formulation of a two-dimensional boundary and use the blending function method [3,51] (Fig. 4c). Finally, the *smart octree* approach by Kudela et al. [52] is an extension of this idea to three-dimensions, which is even capable of handling sharp geometric features on the interface, such as nodes and edges. In case of a parametric boundary, it is enough to find the intersection parameters by which the curve enters and leaves a cell. Finding implicitly defined boundaries renders a more challenging task, involving the solution of a non-linear equation system (EQS), which can be achieved by the *Newton-Raphson method* for example [47].

Divergence theorem

Another approach is to utilize the *divergence theorem* (DT) [53,54] in order to transform an integral over a two- or three-dimensional domain into closed surface or contour integrals, respectively, and thus reducing the dimensionality of the integration by one. The main problem one is faced with during this approach in the FEM, is that the antiderivative of the original integrand has to be computed. While Dasgupta [55] solves this integral symbolically, Sudhakar et al. [56] apply the DT on polynomial functions pre-integrated over arbitrary polyhedra without using costly symbolic computations. Then, the integral over the triangular and quadrilateral facets is computed using Gaussian quadrature rules. Finally, Duczek et al. [57] apply the DT in the framework of the FCM and assume a predefined polynomial integrand, that can be (pre-)computed symbolically. Note that this is a perfectly legitimate assumption in the context of the FCM, where the mesh consists of a regular arrangement of rectilinear cells with a constant Jacobian.

Moment fitting

A third approach is to use the *moment fitting* (MF) method, that derives an individual quadrature rule for an arbitrarily shaped domain, in our case Ω_{phys}, by solving moment fitting EQS. Generally the positions of the quadrature nodes are not known a priori and therefore one has to solve a non-linear EQS, in order to obtain both the nodal positions $\boldsymbol{x}_i$ and their corresponding weights w_i. In cases where there are more nodes than moments, the solution can be obtained according to the *least-squares Newton's method*, as it was done by Mousavi et al. [58]. In addition, Xiao et al. [59] combine the least-squares Newton's method with a node elimination scheme, which results in a decreased number of IPs. However, these approaches become computationally expensive, if applied to integrals over numerous arbitrary domains, which is exactly the case when cells are intersected by arbitrary shaped domain boundaries [60,61]. To circumvent this problem, a set of quadrature nodes with pre-defined positions can be used, resulting in a linear EQS, where only the weights are unknown. While Müller et al. [60] distribute the fixed IPs in the embedding domain $\Omega_e \supset \Omega_{\text{phys}}$, Mousavi and Sukumar [62], Sudhakar et al. [63] and Joulaian et al. [61] use IPs located strictly within the domain of interest Ω_{phys}. A major difference between the last three methods is their assumption of the geometry description. In Ref. [62], Ω_{phys} is assumed to be a convex polytope, while Ref. [63] computes the pre-defined nodal positions in both convex and concave polytopes. The MF method was applied by Hubrich et al. [64] and by Joulaian et al. [61] for three-dimensional problems with curved surfaces approximated by a low-order triangular mesh and tested in the context of FCM. It was shown, that the number of required IPs in the MF is significantly lower, than in the AIS, while maintaining the same accuracy. However, on the contrary, the derivation of individual quadrature rules for every cut cell comes with a significant computational overhead, which makes the MF best suitable for non-linear problems, where the already derived quadrature rules can be reused multiple times [61].

Equivalent polynomials

In the context of the *equivalent polynomial* (EP) approach, one seeks to solve a discontinuous integral in Eq. (1) over the embedding domain Ω_e, by replacing the discontinuous term α with an equivalent polynomial $\mathcal{E}$, which is continuous over Ω_e. Thus, it allows a standard Gaussian quadrature-based integration over the domain of interest without the need of a domain partitioning procedure [39,65].

$$\int\limits_{\Omega_e} \alpha(\boldsymbol{\xi})\mathcal{P}(\boldsymbol{\xi})\,\mathrm{d}\Omega = \int\limits_{\Omega_e} \mathcal{E}(\boldsymbol{\xi})\mathcal{P}(\boldsymbol{\xi})\,\mathrm{d}\Omega \tag{4}$$

There are various studies available in the literature [39,40,65,66], that use the EP approach. The key difference between the existing solutions lies in the formulation of the equivalent polynomial $\mathcal{E}$, the required topology of the discontinuity and the computation of the integrals of the basis function over the physical domain. The works by Ventura [66] and by Ventura and Benvenuti [65], formulated in the context of the XFEM, assume elements intersected by a single straight line in two- or by a single plane in three-dimensional settings. On the one hand, Ref. [66] divides the integration region Ω_e into two sub-regions defined by the positive and negative domains according to the Heaviside function H making the integrand discontinuous, and integrates over the sub-domains analytically using the DT. On the other hand, Ventura and Benvenuti [65] circumvent the need for subdivision of Ω_e by replacing H with its a priori known continuous regularized counterpart H_ρ. Abedian et al. [40] applied the notions of Ref. [66] in the framework of the FCM and furthermore suggested a spacetree-decomposition-based improvement, where lower order equivalent polynomials can be used for each sub-cell. However, according to Ref. [39], in this case not only that an extra spacetree-decomposition is involved, but an EQS for each sub-cell has to be computed as well and therefore this approach does not pay off compared to the traditional QTD-based AIS. Finally, Abedian and Düster [39] describe both $\mathcal{P}$ and $\mathcal{E}$ on the basis of *Legendre polynomials*, whose orthogonality property enables an easy computation of the coefficients of the equivalent Legendre polynomial, without the cost of solving a system of equations.

Motivation

Clearly, there is a wide range of methods capable of computing discontinuous integrals, which all have their own fields of application. Nonetheless, their success is strongly restricted to specific scenarios regarding the shape of the physical domain and the geometry description of the boundary. Obviously, the DT-based methods yield the exact solution with marginal effort, if the integrand is known and if the boundary is defined as a polygonal chain in two-dimension or as a triangulated surface in three-dimension. Similarly, if the discontinuity happens to be given in a parametric form, an LIM with boundary conforming sub-cells can be easily created using the blending function method, however, for implicit boundaries, creating boundary conforming sub-cells renders a much more challenging task. This is also the case for the MF and EP methods, where one way or the other, an integral over an arbitrarily shaped physical domain has to be computed, while the MF even requires a solution of an additional EQS as well, for deriving the unique quadrature rules. For most cases, if the boundary is not given in the required form, additional iterations have to be performed to capture the boundary in a way, that can be handled by the given method. Not only that all these features increase the preparation-time of the integration techniques, but they also make them less robust.

In contrast to the previously mentioned methods, the LIM-based AIS relies on a simple and straightforward algorithm ("Traditional adaptive integration scheme (AIS)" section), which neither requires any iterations nor a solution of an EQS and yet is still capable of approximating the discontinuity independent of the geometry description of the boundary. Clearly, the main advantage of this approach is that it can be robustly used for a whole variety of problems. Although the AIS leaves more detailed information about the boundary unexploited and creates a rather coarse approximation of it, its fundamental idea aligns really well to the basic concept of the FCM, being an easy and effortless meshing combined with exponential convergence rates. Unfortunately, for an accurate solution, the AIS often results in a large number of IPs and thus lengthy computational times. Therefore, in this contribution we propose the *compressed adaptive integration scheme* (C-AIS) ("Compressed adaptive integration scheme (C-AIS)" section), in order to enable a method, that has all the benefits of the AIS, however, with reduced computational costs during the numerical integration.

The finite cell method for linear elastostatics

A detailed introduction to the fundamentals of the FCM was already elaborated in many contributions, such as Refs. [31–35]. Therefore, we only briefly sketch the fundamentals of the FCM. For the sake of simplicity, we restrict ourselves to two-dimensional linear elastostatics, although the formulation of problems with governing equations of different problems, such as dynamics [67,68], fluid mechanics [69] or fracture mechanics [70,71], just to name a few, is also possible.

Note that the novel integration scheme being proposed in "Compressed adaptive integration scheme (C-AIS)" section is not restricted to a specific FDA but can be utilized in any method with unfitted meshes in a straightforward manner. However, throughout this paper we present our approach in the framework of the FCM, where the C-AIS can reach its maximum potential due to the higher order shape functions and the Cartesian meshes which are commonly used. Such a mesh generation goes hand-in-hand with the voxel-models obtained for computed tomography (CT) scans, thus, homogenization of highly complex structures [72–74] and applications to biomechanics [32,36,42–44,70,75] are major topics in the context of the FCM. Note that these are not the only fields where the

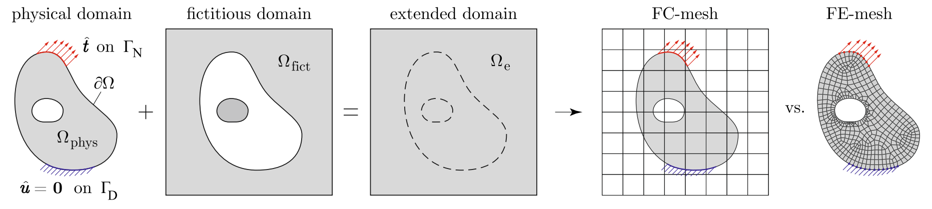


Fig. 5 Concept of the FCM including a typical discretization based on a Cartesian mesh. A comparable FE discretization is depicted on the right

capabilities of the FCM have been proven and exploited: multi-physics problems, such as thermo- [37] and electro-mechanics [76], topology optimization [77], geometrically non-linear problems. Refs. [78–82] as well as physically nonlinear problems [83–85] have also been simulated successfully. Besides static analyses, the extension and application of the FCM to dynamic problems [67,68,76,86–88], to fluid mechanics [69] and last but not least, to phase-field modelling [70,71] has been also in focus of current research activities with promising results. Note that besides the simple Cartesian mesh and hierarchic Legendre shape functions [33], also alternative mesh generation techniques, such as polygonal [89] and tetrahedral meshes [11,70] as well as alternative shape functions, such as hierarchic B-splines [78,85] and Lagrangian shape functions based on the Gauss–Legendre–Lobatto (GLL) nodal distribution (leading to the *spectral cell method*—SCM) [67,76,86] have been also investigated and successfully implemented in the FCM.

Linear elastostatics

Consider a physical domain Ω_{phys} with its corresponding boundary $\partial\Omega$. The domain is subjected to body forces $\boldsymbol{b}$ while on the *Neumann boundary* $\Gamma_{\text{N}} \subset \partial\Omega$ surface tractions $\boldsymbol{t}$ are applied. Then, without derivation, based on the *principle of virtual work* [2,31] the weak formulation of the problem can be expressed by the equality of the bilinear $\mathcal{B}(\boldsymbol{u}, \delta\boldsymbol{u})$ and linear $\mathcal{F}(\delta\boldsymbol{u})$ functionals

$$\mathcal{B}(\boldsymbol{u}, \delta\boldsymbol{u}) = \mathcal{F}(\delta\boldsymbol{u}) \quad \longrightarrow \quad \int\limits_{\Omega} (\boldsymbol{L}\delta\boldsymbol{u})^T \boldsymbol{C}\boldsymbol{L}\boldsymbol{u} \, \mathrm{d}\Omega = \int\limits_{\Omega} \delta\boldsymbol{u}^T \boldsymbol{b} \, \mathrm{d}\Omega + \int\limits_{\Gamma_{\text{N}}} \delta\boldsymbol{u}^T \hat{\boldsymbol{t}} \, \mathrm{d}\Gamma, \tag{5}$$

where $\boldsymbol{u}$ is the displacement field, $\delta\boldsymbol{u}$ the virtual displacement field, $\boldsymbol{C}$ the constitutive matrix in Voigt notation and $\boldsymbol{L}$ the standard strain-displacement operator. In this case, the bilinear functional $\mathcal{B}(\boldsymbol{u}, \delta\boldsymbol{u})$ corresponds to the virtual internal energy, while the linear functional $\mathcal{F}(\boldsymbol{u})$ expresses the virtual work of the external loads. The boundary $\partial\Omega = \Gamma_{\text{N}} \bigcup \Gamma_{\text{D}}$ is divided into the disjoint sets ($\Gamma_{\text{N}} \bigcap \Gamma_{\text{D}} = \emptyset$) of the Dirichlet Γ_{D} and Neumann boundaries Γ_{N}, on which boundary conditions (BCs) for the BVP are applied. While the Neumann BC in regard of the prescribed tractions $\hat{\boldsymbol{t}}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \Gamma_{\text{N}}$ is already included in Eq. (5), the Dirichlet-RB regarding the prescribed displacements $\hat{\boldsymbol{u}}(\boldsymbol{x})$ have to be realized as an additional equation $\boldsymbol{u}(\boldsymbol{x}) = \hat{\boldsymbol{u}}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \Gamma_{\text{D}}$, in order to prevent rigid body motions.

Fictitious domain approach

In the FCM, the physical domain Ω_{phys} is embedded into the larger extending domain Ω_{e} of a simple and regular shape (Fig. 5), and thus the *fictitious* domain is defined as: $\Omega_{\text{fict}} = \Omega_{\text{e}} \setminus \Omega_{\text{phys}}$. In order to distinguish between physical and fictitious domains, we use the indicator function $\alpha(\boldsymbol{x})$ given in Eq. (6). To avoid a possible ill-conditioning of the system matrices, instead of setting $\alpha = 0$ in Ω_{fict}, we assign a small value $\alpha << 1$. In practice, one may use $\alpha < 10^{-10}$ [77]. However, according to Fries et al. [15], a possible ill-conditioning can be expected only in cases where cells exhibit a large difference between the areas of Ω_{phys} and Ω_{fict} within them.

$$\alpha(\boldsymbol{x}) = \begin{cases} \alpha_1 = 1.0, & \forall \boldsymbol{x} \in \Omega_{\text{phys}} \\ \alpha_0 = 0.0 \text{ (in practice : } \alpha_0 << 1), & \forall \boldsymbol{x} \in \Omega_{\text{fict}} \end{cases} \tag{6}$$

Since the fictitious domain has no physical meaning, it should have no effect on the behaviour of physical domain. Therefore, the material in the fictitious domain should be significantly more compliant than that of the physical domain. Without derivation, the original problem stated in Eq. (5) can be also formulated in the extended domain using the indicator function α, as long as $\alpha(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \Omega_{\text{fict}}$ holds [31]:

$$\mathcal{B}_{\text{e}}(\boldsymbol{u}, \delta\boldsymbol{u}) = \mathcal{F}_{\text{e}}(\delta\boldsymbol{u}) \quad \longrightarrow \quad \int\limits_{\Omega_{\text{e}}} (\boldsymbol{L}\delta\boldsymbol{u})^T \cdot \alpha\boldsymbol{C} \cdot \boldsymbol{L}\boldsymbol{u} \quad \mathrm{d}\Omega = \int\limits_{\Omega_{\text{e}}} \delta\boldsymbol{u}^T \cdot \alpha\boldsymbol{b}\,\mathrm{d}\Omega + \int\limits_{\Gamma_{\text{N}}} \delta\boldsymbol{u}^T \cdot \hat{\boldsymbol{t}}\,\mathrm{d}\Gamma \tag{7}$$

Discretization of the weak form

The advantage of the aforementioned embedding in Ω_{e} is manifold: On the one hand, the regular shape of Ω_{e} enables a straightforward and effortless discretization via Cartesian meshes (Fig. 5), which is of great advantage over the FEM, where automatic mesh generation can be challenging to achieve and in some cases, specifically in 3D problems, even impossible [2]. On the other hand, due to the quadratic/cubic shape of the resulting elements of the FCM (Fig. 5), the geometry mapping $\boldsymbol{Q}_{\boldsymbol{\xi}\to\boldsymbol{x}}$ from the reference to global space has a constant Jacobian matrix $\boldsymbol{J}_{\boldsymbol{\xi}\to\boldsymbol{x}}$. This is greatly beneficial regarding the computation of the element matrices, since the Ansatz space remains unchanged by the mapping and an exact numerical integration is possible, when polynomial shape functions are used. On the contrary, in the FEM, the elements can be distorted in various ways, inhibiting an exact numerical integration due to the mapping procedure and thus affecting the quality of the solution [90]. For sake of completeness, we mention the polygonal [89] as well as the tetrahedral extensions of the FCM [11,70], where the above mentioned beneficial features of the traditional FCM regarding the simplicity of the mapping and the Jacobian matrix do not apply. Consequently, the polygonal and tetrahedral FCM are only advantageous for special applications, as for example the consideration of pores in already existing FE-meshes [91]. Furthermore, if the given problem contains non-smooth features, introduced by material interfaces, cracks or sharp corners within the elements, special FC-meshes are required, based on the local-enrichment [92,93] or multi-level hp-refinement strategy [75,94].

Unlike in the FEM, the generated elements do not conform to the boundary $\partial\Omega$. In order to distinguish them from the traditional finite elements (FEs), they are commonly referred to as finite cells (FCs). This notation is also used throughout this paper. Nevertheless, despite of this difference, the FCM is identical to the FEM in many important aspects: The FCs are still defined in the reference space with the corresponding local coordinates $\boldsymbol{\xi} = [\xi, \eta]^T$, where $\xi,\ \eta \in [-1, 1]$. Additionally, the displacement field $\boldsymbol{u}(\boldsymbol{x})$ of the individual cells is approximated analogously to the conventional FEM using the same families of shape functions. Thus, the global equation system

$$\boldsymbol{K}\boldsymbol{U} = \boldsymbol{F}, \tag{8}$$

whose solution is the global displacement vector $\boldsymbol{U}$, is obtained by the assembly of cell-specific stiffness matrices $\boldsymbol{K}^{(c)}$ and load vectors $\boldsymbol{F}^{(c)}$. Thus, already established solvers known from many FE-packages can be used straightforwardly [3,95]. As an example, the stiffness matrix $\boldsymbol{K}^{(c)}$ of a specific cell is given below

$$\boldsymbol{K}^{(c)} = \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{B}^{T}\alpha\boldsymbol{C}\boldsymbol{B}\det\left(\boldsymbol{J}^{(c)}_{\boldsymbol{\xi}\to\boldsymbol{x}}\right)\,\mathrm{d}\xi\,\mathrm{d}\eta, \tag{9}$$

where $\boldsymbol{B}$ is the so-called *strain-displacement matrix*, containing derivatives of the shape functions. Furthermore, $\boldsymbol{J}_{\boldsymbol{\xi}\to\boldsymbol{x}}$ is the Jacobian matrix of the already mentioned geometry mapping $\boldsymbol{Q}_{\boldsymbol{\xi}\to\boldsymbol{x}}$, establishing the relation between the local $\boldsymbol{\xi}$ and global $\boldsymbol{x}$ coordinates of a given cell, according to Eq. (10). Due to the simple shape of the cell in the global space, the mapping is a linear function, characterized by the global coordinates x_1 and y_1 of the lower left corner as well as by the height h_x and width h_y of the cell. It is easy to see, that in case of such a simple mapping, $\det(\boldsymbol{J}_{\boldsymbol{\xi}\to\boldsymbol{x}})$ is indeed constant and depends on the width h_x and height h_y of the cell only.

$$\boldsymbol{x} = \boldsymbol{Q}_{\boldsymbol{\xi}\to\boldsymbol{x}}(\boldsymbol{\xi}) = \begin{bmatrix} x_1 + 1/2(1+\xi)h_\mathrm{x} \\ y_1 + 1/2(1+\eta)h_\mathrm{y} \end{bmatrix} \longrightarrow \boldsymbol{J}_{\boldsymbol{\xi}\to\boldsymbol{x}} = \mathrm{grad}^{T}\left[\boldsymbol{Q}_{\boldsymbol{\xi}\to\boldsymbol{x}}(\boldsymbol{\xi})\right] = \frac{1}{2}\begin{bmatrix} h_x & 0 \\ 0 & h_y \end{bmatrix} \tag{10}$$

Due to the extension to Ω_e and the non boundary-conforming nature of the discretization, the generated cells can be classified[4] as follows:

1. *Physical cells* are located entirely in the physical domain and their stiffness matrices $\boldsymbol{K}^{(c)}_{\mathrm{phys}}$ are identical to the ones of the FEs, due to the fact that $\alpha = 1$ in Eq. (9). Assuming a polynomial Ansatz space and constant material properties, $\boldsymbol{K}^{(c)}_{\mathrm{phys}}$ can be integrated via Gaussian quadrature to machine precision [88]. Furthermore, due to their uniform shape and size, a pre-computation of $\boldsymbol{K}^{(c)}_{\mathrm{phys}}$ is possible, which then applies to all physical cells (see dark gray cells in Fig. 6) [25].
2. *Fictitious cells*, which are located entirely in the fictitious domain, have no physical meaning and therefore, they can be excluded from the simulation (see light grey cells in Fig. 6).
3. *Cut cells*,[5] as the name implies, are intersected by the boundary $\partial\Omega$, and hence they contain pieces of both the physical and fictitious domains (see yellow cells in Fig. 6). The difficulty of computing accurate results for Eq. (9) in this case arises due to the discontinuous nature of α within the cell (see Fig. 2), as discussed in "Review of numerical integration methods: discontinuous integrands" section.

In the FCM, significant drawbacks of the FEM, such as the typically demanding discretization procedure and integration error caused by the distorted elements in the initial mesh, can be avoided. However, due to the cut cells present in the FCM (and generally in

[4]Note that other fictitious domain methods may use different notations, such as *internal, external* and *boundary elements* [30] or *in (I), out (O)* and *neither- in- nor- out (NIO)* elements [25]. However, essentially they all refer to the same classification.
[5]Often also referred to as *broken cells.*

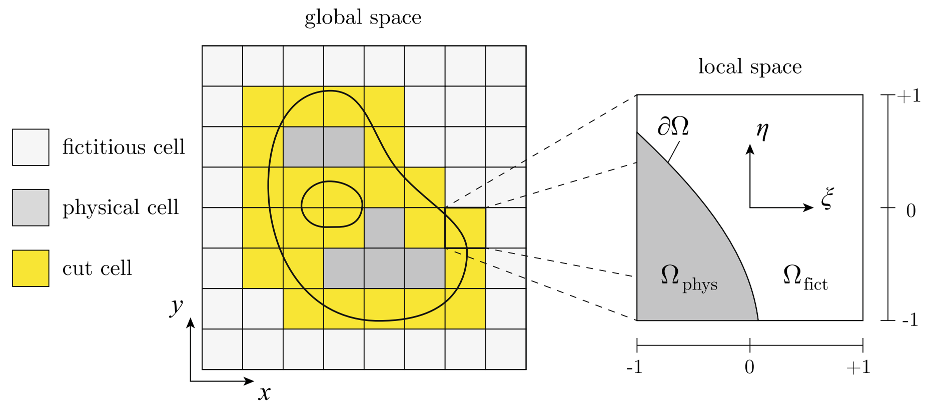


Fig. 6 Different types of sub-cells (left) and a cut cell with its local coordinate system $\boldsymbol{\xi} = [\xi, \ \eta]^T$ (right)

all fictitious domain methods), the main challenge is shifted to the implementation of the BCs and to the computation of the cell matrices of cut cells [31,33]. While an exhaustive introduction was given regarding the computation of cell matrices in "Introduction", we only briefly mention the enforcement of the BCs in the FCM and refer the reader to the pertinent literature.

Inhomogeneous Neumann BCs can be implemented by considering the last term in Eq. (7), expressing an integral over the part of the boundary $\Gamma_{\mathrm{N}} \subset \partial\Omega$, located within the cut cells. A detailed description of the implementation of such BCs for three-dimensional triangulated surfaces can be found in [32]. On the other hand, homogeneous Neumann BCs require no special treatment, assuming that the fictitious domain has a material with zero stiffness.

Methods such as the *penalty method*, *Lagrange multiplier method* and *Nitsche's method* can be used to enforce Dirichlet BCs in a weak sense by focusing on finding the stationary point of the original problem augmented by a constraint potential. Detailed explanations of these methods can be found in Refs. [4,37,38,96].

Adaptive integration schemes

In "Motivation" section, after discussing the different approaches typically used in the fictitious domain concept for computing discontinuous integrands, the benefits of the AIS compared to the other methods were stated. In this section, the AIS is studied in greater detail ("Traditional adaptive integration scheme (AIS)" section), then, the C-AIS approach is proposed and investigated with regard to the reduction of the computational costs compared to the AIS ("Compressed adaptive integration scheme (C-AIS)" section). Finally, in "Combination with other approaches" section, the performance of the C-AIS is further enhanced by combining it with other methods discussed in the literature.

Traditional adaptive integration scheme (AIS)

The computation of cell-matrices of the broken cells is challenging, since it involves discontinuous integrands, where a strong discontinuity is introduced by the indicator function α ("Review of numerical integration methods: discontinuous integrands" section). At this point we recall the general form of such discontinuous integrals given in Eq. (1), while

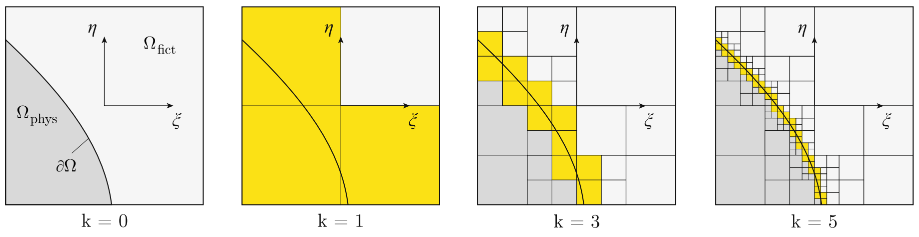


Fig. 7 QTD of the individual cell highlighted in Fig. 6 for three different refinement levels $k = 1, 3, 5$

a specific example in form of the cell stiffness matrix is given in Eq. (9). Since the aim of this work is the improvement of the computational speed of the AIS, let us briefly outline its procedure. In the traditional AIS [32], the LIM (see "Local integration mesh" section) is created in two-dimensional cases via a *quadtree-decomposition* (QTD), which is based on the recursive sub-division of the cut sub-cells in four equal sized quadrants. This procedure continues until a refinement level k defined by the user is reached. In the end, the QTD produces a non-uniform integration mesh, consisting of quadratic sub-cells, with decreased size and increased density in the vicinity of the discontinuity (Fig. 7) [78,97]. Similar to the classification of cells, the resulting sub-cells as well are classified as physical (dark grey), fictitious (light grey) and cut/broken (yellow). At this point we emphasize again, that the created mesh serves integration purposes only and it does not introduce new DOFs to the global system. The equivalent three-dimensional version of the QTD is the *octree-decomposition* (OTD), during which every cut sub-cell divides into eight equal sized cube-shaped octants.

Now, in terms of the stiffness matrix $\boldsymbol{K}^{(c)}$, the AIS being based on an LIM ("Local integration mesh" section), is computed according to Eq. (3) over the n_{sc} sub-cells as follows

$$\boldsymbol{K}^{(c)} = \sum_{sc=1}^{n_{sc}} \boldsymbol{K}^{(c,sc)}, \tag{11}$$

where the stiffness matrix $\boldsymbol{K}^{(c,sc)}$ of a particular sub-cell sc in a given cell c is

$$\boldsymbol{K}^{(c,sc)} = \int_{-1}^{1}\int_{-1}^{1} \boldsymbol{B}^T(\boldsymbol{\xi}(\boldsymbol{r}))\,\alpha(\boldsymbol{x}(\boldsymbol{\xi}(\boldsymbol{r})))\boldsymbol{C}\,\boldsymbol{B}(\boldsymbol{\xi}(\boldsymbol{r}))\;\underbrace{\det\left(\boldsymbol{J}^{(sc)}_{\boldsymbol{r}\to\boldsymbol{\xi}}\right)}_{\text{const.}}\,\underbrace{\det\left(\boldsymbol{J}^{(c)}_{\boldsymbol{\xi}\to\boldsymbol{x}}\right)}_{\text{const.}}\;\mathrm{d}r\mathrm{d}s \tag{12}$$

Note that the above integral is computed in the local coordinates $\boldsymbol{r} = [r, s]^T$ of the sub-cells, hence the integrand in Eq. (9) is extended by the term $\det(\boldsymbol{J}^{(sc)}_{\boldsymbol{r}\to\boldsymbol{\xi}})$, where $\boldsymbol{J}^{(sc)}_{\boldsymbol{r}\to\boldsymbol{\xi}}$ is the Jacobian of the geometry mapping $\boldsymbol{\xi} = \boldsymbol{Q}_{\boldsymbol{r}\to\boldsymbol{\xi}}(\boldsymbol{r})$, defining the position of a sub-cell in $\boldsymbol{\xi}$ (see Fig. 8) [32]. Again, due to the simple shape of the sub-cells,[6] similar to Eq. (10), the mapping is linear and has a constant Jacobian. In Eq. (13), ξ_1 and η_1 refer to the lower left corner of a given sub-cell, while h_ξ and h_η are its height and width measured in the cell (Fig. 8).

$$\boldsymbol{\xi} = \boldsymbol{Q}_{\boldsymbol{r}\to\boldsymbol{\xi}}(\boldsymbol{r}) = \begin{bmatrix} \xi_1 + 1/2(1+r)h_\xi \\ \eta_1 + 1/2(1+s)h_\eta \end{bmatrix} \longrightarrow \boldsymbol{J}_{\boldsymbol{r}\to\boldsymbol{\xi}} = \mathrm{grad}^T\left[\boldsymbol{Q}_{\boldsymbol{r}\to\boldsymbol{\xi}}(\boldsymbol{r})\right] = \frac{1}{2}\begin{bmatrix} h_\xi & 0 \\ 0 & h_\eta \end{bmatrix} \tag{13}$$

[6]Although not all sub-cells are of the same size, they are all quadratic.

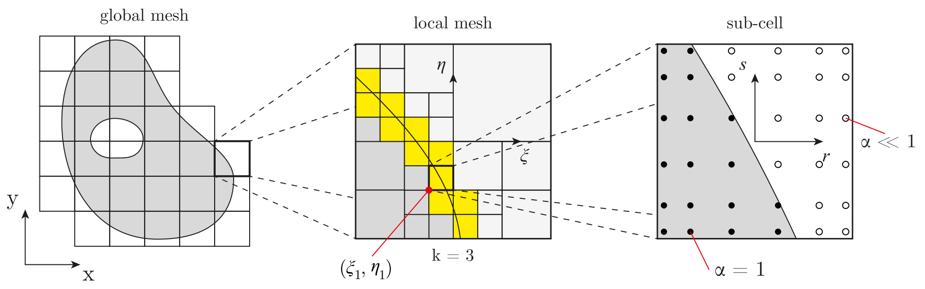


Fig. 8 Illustration of the basic idea of the AIS: The numerical integration of each sub-cells is executed in the local *r*-*s*-coordinate system, where the integrand is evaluated at all IPs and the result is scaled based on the α-value obtained from Eq. (6)

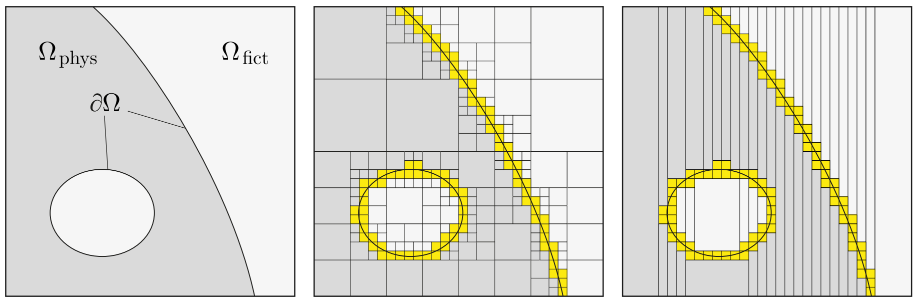
In the AIS, the integration over the individual sub-cells is computed via Gaussian quadrature, as depicted in Fig. 8, where the values of the integrand at the IPs in the physical and fictitious domains are scaled by α according to Eq. (6). The resolution of the mesh and therefore the accuracy of the integration is determined by the refinement level k. Due to α being constant within the physical and fictitious sub-cells, applying Gaussian quadrature rules to the sub-cells for computing Eq. (12) yields exact results, assuming $\boldsymbol{B}^T\boldsymbol{C}\boldsymbol{B}$ is a polynomial function. However, the application of Gaussian quadrature rules to the cut cells results in significant errors as it is applied to non-polynomial functions, in this case discontinuous functions. Since the boundary in the cut sub-cells is captured by the IPs, the integration error can be decreased when using sufficiently small sub-cells and/or a high-order Gaussian quadrature rule [77]. While using the first approach the total number of IPs increases exponentially with $\mathcal{O}(2^k)$, the latter method yields a quadratic increase $\mathcal{O}(n_{\text{dir}}^2)$, where n_{dir} refers to the number of IPs in the ξ- and η-directions for each sub-cell. Note that despite these challenges it has been mathematically shown that if the integration error is below the discretization error, i.e., the discretization error dominates the overall numerical error, the theoretical optimal convergence known from the FEM is also attained in the FCM [10].

The main advantage of the AIS lies in its simplicity and robustness. While the former attribute enables a straightforward implementation, the latter one results from its capability of handling any kind of geometry representation by requiring inside-outside tests[7] only. If two points in the same cell belonging to two different domains are found, the cell must be cut. Its weaknesses lie in its rather poor boundary approximation and in the exponentially increasing rate of n_{sc} for incrementally increasing k. Thus, not only the time required by the QTD grows exponentially, but also the number of the IPs when integrating over the sub-cells, leading to extreme computation costs for the AIS [40,52,57].

Compressed adaptive integration scheme (C-AIS)

At the end of "Traditional adaptive integration scheme (AIS)" section, we stated the drawbacks of the AIS. In order to reduce the computational effort needed for an accurate numerical integration, different approaches have been developed [38,40,45,83,84], which aim at the reduction of the IPs during the AIS. In this section we present a new approach,

[7]The term inside-outside test refers to a procedure that is capable of determining whether a given point is located in the interior (inside) or exterior (outside) of the physical domain.



a Example of a cut cell. **b** Set of sub-cells $\boldsymbol{\mathcal{S}}$ ($n_{\text{sc}} = 253$) resulting from the QTD with $k = 5$. **c** Set of sub-cells $\boldsymbol{\mathcal{R}}$ ($n_{\text{R}} = 147$) after the compression of $\boldsymbol{\mathcal{S}}$ using IBR.

Fig. 9 Exemplary cut cell (**a**) and the resulting sets of integration sub-cells when a QTD (**b**) or the IBR algorithm (**c**) are applied. Note that the local coordinate systems $\boldsymbol{\xi}$ are not depicted in the sub-figures for the sake of clarity. For more details concerning the different compression techniques, please consult "Compression techniques" section

the *compressed adaptive integration scheme* (C-AIS), which seeks the reduction of the IPs by applying image-compression techniques on the set of n_{sc} sub-cells $\boldsymbol{\mathcal{S}} = \{\mathcal{S}_i\}_{i=1}^{n_{\text{sc}}}$ resulting from the QTD. An example of a cut cell as well as its QTD are depicted in Fig. 9a, b, respectively. The compression itself is realized as an additional step being executed after the QTD, resulting in a new set of non-overlapping rectangular sub-cells $\boldsymbol{\mathcal{R}} = \{\mathcal{R}_j\}_{i=1}^{n_{\text{R}}}$, where $n_{\text{R}} \leq n_{\text{sc}}$, as depicted in Fig. 9c. This also implies, that the QTD is still required. Finally, the integration itself is then carried out over $\boldsymbol{\mathcal{R}}$ via Gaussian quadrature, analogous to the standard AIS, while maintaining the same accuracy. Consequently, the compression step can be straightforwardly implemented as an intermediate step between the QTD and the actual numerical integration, without the need for major modifications in the already existing codes. Note that the C-AIS is only concerned with the number and shape of the sub-cells, and it is independent of the family of shape functions or nodal distributions that are employed for the approximation of the solution field.

Shape of the sub-cells

Due to the rectangular shape of the sub-cells in $\boldsymbol{\mathcal{R}}$ (see for example Fig. 9c), the determinant of the Jacobian matrix $\boldsymbol{J}_{\boldsymbol{r}\to\boldsymbol{\xi}}$ of the geometry mapping $\boldsymbol{Q}_{\boldsymbol{r}\to\boldsymbol{\xi}}$ is still constant (see Eq. (13) for $h_\xi \neq h_\eta$), enabling a simple and exact integration over $\boldsymbol{\mathcal{R}}$ via Gaussian quadrature, if a polynomial Ansatz space is used. Note that an exact integration is also possible for sub-cells with curved edges, if and only if the mapping $\boldsymbol{Q}_{\boldsymbol{r}\to\boldsymbol{\xi}}$ is polynomial—otherwise an exact numerical integration is not possible. Nonetheless curved sub-cells require an increased number of IPs for the numerical integration, while for square- and rectangular-shaped sub-cells no additional IPs are needed. Furthermore, it is well known, that among other parameters, the aspect ratio of two-dimensional FEs has an effect on the convergence properties of the solution [3]. While allowing elongated rectangular sub-cells, they can have extreme aspect ratios (see for example Fig. 9c), we would like to emphasize, that the created sub-cells serve for integration purposes only and regardless of the aspect ratios, the parent cell, which holds actual DOFs, is not affected in its shape. In conclusion, when using rectangular sub-cells, an exact integration via Gaussian quadrature is still possible, without the need for any additional IPs and without affecting the convergence of the solution.

Accuracy

The compression should only affect the speed of the AIS but not its accuracy and therefore, the equality regarding the integral value I over a given cell

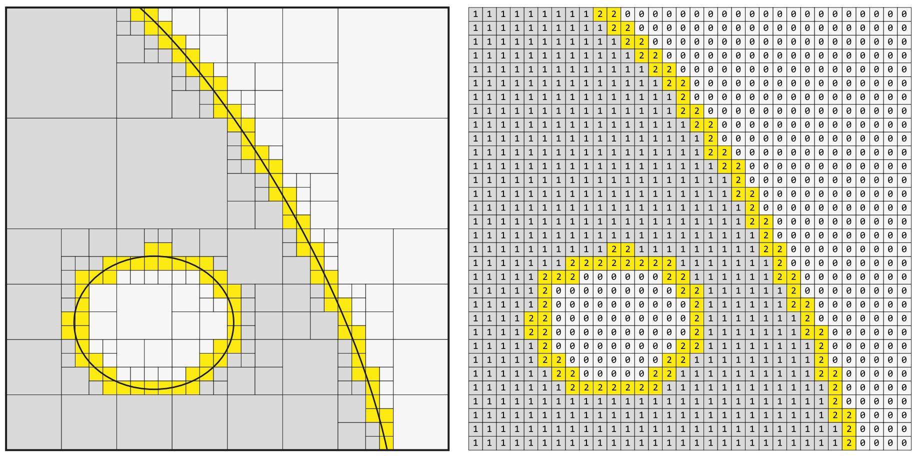
$$I = \sum_{i=1}^{n_{\mathrm{sc}}} \int_{(S_i)} \alpha(\boldsymbol{x}(\boldsymbol{\xi}))\mathcal{P}(\boldsymbol{\xi})\,\mathrm{d}A = \sum_{j=1}^{n_{\mathrm{R}}} \int_{(R_j)} \alpha(\boldsymbol{x}(\boldsymbol{\xi}))\mathcal{P}(\boldsymbol{\xi})\,\mathrm{d}A, \tag{14}$$

has to be demanded. The above equation holds, if (i) a *lossless* compression is used and if (ii) the compression targets only the physical and fictitious sub-cells, over which the integrand is constant, while the cut sub-cells are left unchanged. This can be explained by the following reasoning. The boundary is resolved by the IPs in the cut sub-cells, i.e., the IPs are partly located in the physical domain, while the rest is distributed in the fictitious one. Although the Gaussian quadrature is not suitable at all for the discontinuous integrals over the cut sub-cells, by using small enough sub-cells along the boundary the problematic area can be restricted to a negligible domain and the error can be kept below a certain threshold as already described in Sec. 5.1. The smaller the cut sub-cells along the boundary and the denser the distributed IPs are, the smaller the error becomes. Since for an accurate integration it is crucial to have enough IPs capturing the discontinuity consequently, it is not recommended to compress the cut sub-cells.

Image-compression techniques

Compressing the sub-cells in $\boldsymbol{\mathcal{S}}$ (see for example Fig. 10a) by merging adjacent sub-cells into larger blocks is a challenging task due to the non-uniform sizes of the sub-cells. Instead, the set $\boldsymbol{\mathcal{S}}$ will be transformed by the function T to an equivalent pixel-image $\mathcal{I}$, where the individual pixels covered by the fictitious, physical and cut sub-cells are assigned by the pixel-values 0, 1 und 2, respectively (see Fig. 10b). This and the following steps of the proposed approach are sketched in the flowchart in Fig. 11. There, also the used compression techniques are listed (RLE, IBR and MRP) that are discussed in "Compression techniques" section. Note that the QTD is not only a space partitioning method, but also an image-compression technique [98], thus, $\boldsymbol{\mathcal{S}}$ can be seen as a compressed version of $\mathcal{I}$ and applying T means no more, than *reconstructing* $\mathcal{I}$ from its compressed state. Clearly, T can be performed with negligible effort. The image should not contain more information than necessary and therefore its resolution $2^k \times 2^k$ is defined by the refinement level k of the QTD. Thus, the smallest sub-cell in $\boldsymbol{\mathcal{S}}$ corresponds to a single pixel in $\mathcal{I}$. Once $\mathcal{I}$ is created, other compression techniques can be applied to the pixel-regions defined by the values 1 and 0. Then, the results are transformed from the pixel-space back to the local-space of the cell by the function T^{-1}, in order to obtain the position of the sub-cells $\boldsymbol{\mathcal{R}}$ in $\boldsymbol{\xi}$. Since both the QTD and the used compression methods represent lossless compression techniques, the pixel-image can be restored and recompressed without any error, which is a major requirement for Eq. (14) to hold.

It is important to emphasize, that since we are dealing with image-compression, most likely all methods would significantly reduce the number of pixels in the image. However, our goal is not only to compress the image, but to apply a compression that outperforms the QTD in terms of the number of the resulting sub-cells.



a QTD ($k = 5$) of an exemplary cell with a discontinuity of complex topology.

b Pixel image corresponding to Fig. 10a.

Fig. 10 QTD of a cut cell and corresponding pixel-image. The pixel values 0, 1 and 2 denote physical, fictitious and cut sub-cells, respectively. The image consist of $2^k \times 2^k$ pixels, where k is the refinement level of the QTD ($k = 5$)

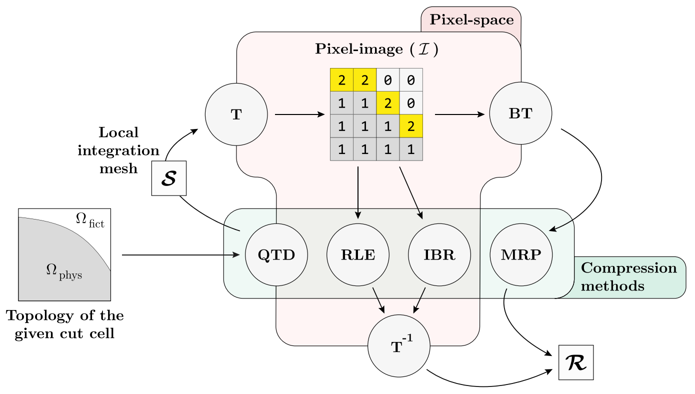


Fig. 11 Flowchart of the compression techniques applied to the integration sub-cells in the fictitious domain approach. Depending on the different image-compression techniques such as run-length encoding (RLE), image-block representation (IBR) and minimum rectangular partition (MPR), auxiliary procedures including a transformation (*T*) from the local- to the pixel-space and a boundary tracing (BT) procedure need to be introduced. The methods that are used for a transition from the local- to the pixel-space and vice versa are indicated at the boundary marking the pixel-space. Note that the QTD is also essentially an image-compression technique, thus it can operate both in the pixel- and local-space. Therefore, it is also located on the boundary

Alternative solutions for deriving the pixel-image

The question naturally arises, whether the need for the costly QTD could be eliminated by either (i) directly creating the pixel-image from the topology of the given cut cell or—which would be even better—(ii) by utilizing the other compression techniques for creating the local mesh directly from the topology of the cut cell, thus additionally eliminating the steps of the reconstruction as well as of the compression.

The first option is possible when we distribute $2^k \times 2^k$ equal-sized sub-cells in an isotropic manner in the cell and evaluate whether they belong to the physical, fictitious or cut domains. Note that in this case k is no longer the refinement level, only an integer defining the side-length of all the sub-cells by $l = 2/(2^k)$. In this case, there is a one-to-one correspondence between the sub-cells and the pixels of the image. Matter of fact, such a local mesh was already proposed by Parvizian et al. [31], however they have directly used the mesh for the numerical integration by the *trapezoidal rule*. According to our tests on implicitly defined geometries, it is much more time-efficient to use the QTD rather than an isotropic distribution of sub-cells, as it increases the density of sub-cells only along the boundary but not in the entire cell. Consequently, less time is spent on determining the status of the sub-cells, i.e., whether they are physical, fictitious or cut. Therefore, even if the QTD requires an additional step of reconstructing the pixel-image using T, it is still a computationally more efficient solution. Constructing a *bounding box* around the discontinuity shrinks the domain in which the sub-cells (pixels) have to be checked. However, the resulting reduction in computational effort is heavily dependent on the topology of the discontinuity. The method based on the use isotropic sub-cells only becomes efficient when the inside-outside tests have to be ran on a small portion of the sub-cells, which is for example the case when the boundary $\partial\Omega$ is defined by straight lines or by planar surfaces. Note that one may modify the QTD in such a way, that it already yields $\mathcal{I}$ instead of $\boldsymbol{\mathcal{S}}$. However, in this case, the transformation T is not eliminated, it is just integrated in the QTD.

Unfortunately there is no record or evidence for the second option to be computationally cheaper or even possible. Although other compression techniques such as the RLE, the IBR and the MRP ("Compression techniques" section) generally yield a higher compression than the QTD, they are rather unsuitable for the partitioning of the cell and for iteratively capturing the boundary.

In conclusion, for the construction of the initial mesh, the QTD seems to be the best candidate combining the abilities of space-partitioning and data-compression. While the first property is required for the creation of an initial local mesh independent from the geometry description of the boundary and from the connectivity of the physical and fictitious domains in the cell, the second feature enables the reconstruction of the pixel-image. Note that having an initial LIM, where the sub-cells can take any position and form within the cell, could not be transformed to a corresponding pixel-image $\mathcal{I}$ and would make the process of compressing the sub-cells significantly harder.

Performance

We have to stress again that the main goal is not only to achieve a high compression of $\mathcal{I}$, but to outperform the QTD in terms of the number IPs. Therefore, it is reasonable to evaluate the results with respect to the QTD. In the following, let us use the *compression ratio of the number of the IPs*

$$\lambda = n_{\mathrm{IP}}(\boldsymbol{\mathcal{R}})/n_{\mathrm{IP}}(\boldsymbol{\mathcal{S}}) > 0, \tag{15}$$

where the function $n_{\mathrm{IP}}(\cdot)$ expresses the number of IPs for a given set of sub-cells, being deployed for the numerical integration. In that sense, the smaller the ratio λ (for $\lambda < 1$), the better the compression, while $\lambda > 1$ indicates an undesired scenario, where the number IPs is increased compared the QTD (see "Drawback of the RLE compression technique" section for such a scenario). In the framework of the FCM combined with the AIS, the complexity of $\mathcal{I}$ depends (i) on the topological complexity of the physical boundary $\partial\Omega$ within a given cell and (ii) on the refinement level k of the QTD. Clearly, if the adjacent pixels are likely to have different values (= highly complex boundary) or if the image consists of a few pixels only (= low k), λ is less significant. Note that in the context of the FCM, an accurate and efficient solution can only be achieved if the discretization and integration errors are balanced. This generally requires FCs that are sufficiently small such that the boundary variation within one cell is not too complex (the smooth shape functions need to be capable of capturing the solution fields) and additionally the refinement level k has to be high enough to ensure an accurate numerical integration. Therefore, the number of integration sub-cells is typically rather high such that compression algorithms can produce meaningful results, i.e., a significantly decreased number of integration domains. Similar to λ, let us define the *time compression ratio*

$$\tau = t(C\text{-}AIS)/t(AIS), \tag{16}$$

where the function $t(\cdot) = t_{\mathrm{LIM}}(\cdot) + t_{\mathrm{I}}(\cdot)$ expresses the total time needed for the chosen algorithm, including both the time for creating the local mesh $t_{\mathrm{LIM}}(\cdot)$ as well as the time required for the integration $t_{\mathrm{I}}(\cdot)$ over the set of resulting sub-cells. Both the C-AIS and AIS are based on a QTD requiring t_{QTD} time, however, the C-AIS involves an additional step of compression requiring t_{C} time, i.e., for the C-AIS $t_{\mathrm{LIM}} = t_{\mathrm{QTD}} + t_{\mathrm{C}}$. Thus, it is obvious, that $t_{\mathrm{LIM}}(C\text{-}AIS) \geq t_{\mathrm{LIM}}(AIS)$. On the other hand, due to the compression a reduced integration time $t_{\mathrm{I}}(\boldsymbol{\mathcal{R}}) \leq t_{\mathrm{I}}(\boldsymbol{\mathcal{S}})$ is obtained. It is clear, that as long as the following inequality holds $t_{\mathrm{C}} < t_{\mathrm{I}}(\boldsymbol{\mathcal{S}}) - t_{\mathrm{I}}(\boldsymbol{\mathcal{R}})$, a desired overall time compression ratio $\tau < 1$ can be achieved. An undesired scenario is indicated by $\tau > 1$, i.e., the computational time invested in the compression is greater than the time saved when integrating over the compressed sub-cells.

Compression techniques

Run-length encoding (RLE)

The RLE (Fig. 12a) is one of the oldest and most straightforward approaches among the traditional image-compression techniques. It is based on scanning the pixel-image and replacing constant valued pixel bands of unit width by their start- and end-pixels as well as by the pixel-value being constant throughout the band. Considering image-compression, one may use different scanning schemes, such as row-wise, columns-wise or zig-zag. In any case, the algorithm checks every single pixel in $\mathcal{I}$ and thus its time complexity is proportional to $\mathcal{O}(n^2)$, where the resolution of the image is $n \times n$ pixels. Obviously, a low complexity of an image results in an improved compression [98,99]. In the framework of the FCM, the row- and column-wise scanning directions are of main concern, resulting in vertical or horizontal elongated rectangular shaped sub-cells $\boldsymbol{\mathcal{R}}$ after the transformation T^{-1}. The implemented code performs both a row- and column-wise RLE compression of $\mathcal{I}$ and chooses the one with the fewer number of resulting rectangles. Since the RLE performs pixel-checking operations only, it can easily compress images with multiply connected regions, i.e., domains that may enclose hole-regions (see for example physical pixels in Fig. 10b) and/or consist of several regions isolated from each other.

Image block representation (IBR)

The IBR (Fig. 12b) is a very simple yet powerful extension of the RLE. It is based on a primary RLE, which is then followed by merging adjacent rectangles into larger blocks, if they have the same start and end positions. In case of a column-wise RLE, a horizontal

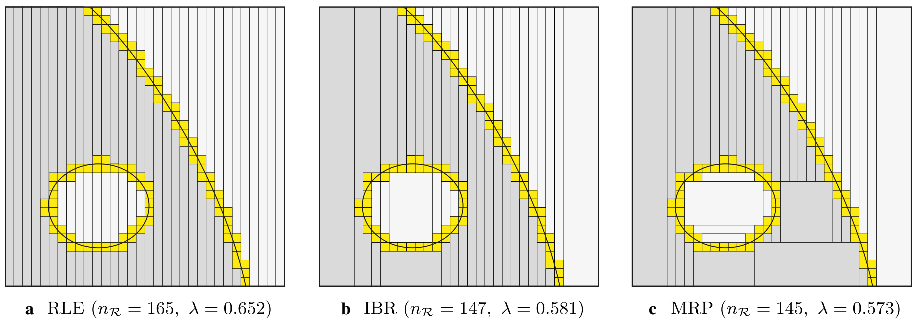
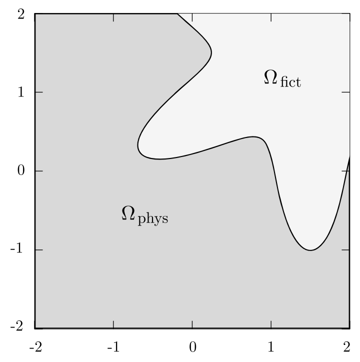

a RLE ($n_{\mathcal{R}} = 165,\ \lambda = 0.652$)
b IBR ($n_{\mathcal{R}} = 147,\ \lambda = 0.581$)
c MRP ($n_{\mathcal{R}} = 145,\ \lambda = 0.573$)

Fig. 12 Sub-cells resulting from the different compression techniques, when applied to the example depicted in Fig. 9b. Since the cut sub-cells (yellow) are not compressed, they are identical for all compression techniques

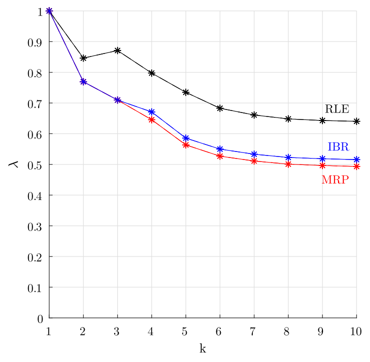
merging procedure is invoked during the IBR, while considering a row-wise RLE, a vertical merging technique is applied. Whether the results of the horizontal- or the row-wise RLE are passed on to the IBR, depends on which one of the both approaches yields less rectangular domains. As the RLE, the IBR is based solely on pixel checking operations and can be performed with marginal additional effort compared to the RLE, however, it yields significantly better results [98,100]. Furthermore, similar to the RLE, the IBR can easily handle multiply connected regions. As outlined in "Comparison" section, while the condition $n_{\mathrm{R}} \leq n_{\mathrm{sc}}$ is not always fulfilled in certain cases for the RLE, the IBR is not prone to such artifacts and reliably yields a compression rate of the IPs λ below 1.

Minimal rectangular partition (MRP)

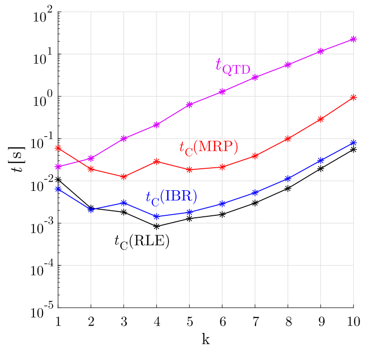
The methods presented in the previous two paragraphs significantly reduce the number of sub-cells and in most cases they perform considerably better than the QTD alone. Nevertheless, the fewest number of sub-cells results from the MRP, which *guarantees* a minimal partitioning into a set of non-overlapping rectangles for two-dimensional rectilinear polygons. Such a minimal partitioning can be achieved by translating the problem into the graph-theoretic realm and solving it on an equivalent bipartite graph, hence in some papers it is also referred to as a *graph-based decomposition* (GBD) [98,101]. In the following, we stick to the term MRP, as it better articulates the focus of this method. Finding the minimum partition is typically an optimization problem and as such it comes at a cost. The MRP algorithm generally has a much higher complexity and slower performance compared to the simple RLE and IBR algorithms [98]. Since for the MRP a rectilinear input polygon is needed, the procedure starts with a boundary tracing [102–104] of the physical and fictitious pixel domains, realizing the transition from the pixel-space to the local-space, where then the MRP can be performed (Fig. 11). Since the MRP performs the partition in the geometric space, multiply connected regions require an even more complex algorithm. Since a detailed introduction to the MRP is beyond the scope of this work, for general information about the method we recommend Refs. [98,101,105], while for its application to the compression of integration sub-cells in the framework of the FCM we refer to Petö et al. [106]. For further reading about the minimal partitioning/covering problems, the interested reader is referred to Refs. [107–111]. Regarding graph-theoretic tools, we recommend Ref. [112].



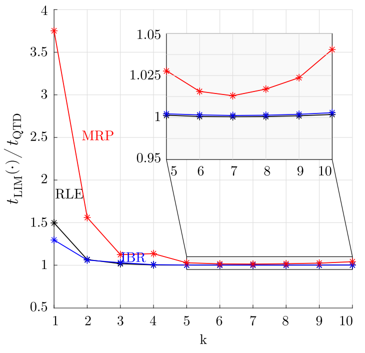
a Computational domain discretized by a single cell. The geometry is given by Eq. (17).



b Compression rate λ vs. refinement level k.



c Time to compress the initial QTD of the cut cell (t_{C}) by means of the RLE, IBR, or MRP methods in comparison to the time needed to set up the QTD (t_{QTD}).



d Time requirement for setting up a compressed LIM (t_{LIM}) compared to time needed for performing the QTD (t_{QTD}).

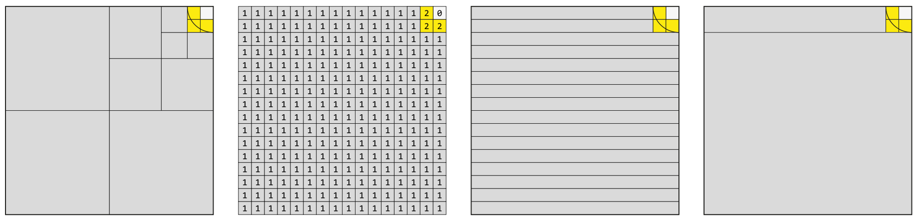
Fig. 13 Comparison of the performance and time requirements of the RLE, IBR and MRP algorithms for a given example

Comparison

A visual comparison of $\mathcal{R}$ obtained by the different methods when applied to Fig 10a is depicted in Fig. 12. A further investigation of the compression techniques regarding their performance as a function of the refinement level k is shown in Fig. 13. The domain for which a local integration mesh should be created is depicted in Fig. 13a, where Ω_{fict} is defined by the implicit function given in Eq. (17).

$$\sqrt{(x-1.5)^2+(y-1.5)^2} \leq \left[0.2 \cdot sin(6\varphi+\pi/2)+0.6\right]^{-1}, \quad \text{where} \quad \varphi = \mathtt{atan2}(y-1.5, x-1.5) \tag{17}$$

Figure 13b, depicts the $\lambda(k)$-curves of the examined methods, which all converge to stationary values, expressing the best compression reachable by the different methods. The curves are in a good agreement with Fig. 12 as well, although the RLE shows somewhat different results for the two different domains, due to the phenomenon discussed later in "Drawback of the RLE compression technique" section. Despite the fact, that the MRP yields a minimal partition, it comes with the cost that it requires approximately one order of magnitude more time for the compression t_{C} than the RLE and IBR, as shown in Fig. 13c. Nevertheless, t_{C} of the different methods for reasonable refinement levels ($k \geq 5$) is negligible compared to t_{QTD}, which is unconditionally included in the C-AIS



a QTD ($k = 4$) with $|\mathcal{S}| = 13$ sub-cells. **b** Corresponding pixel image with 16×16 pixels. **c** RLE compression with $|\mathcal{R}| = 20 > |\mathcal{S}|$ sub-cells **d** IBR compression with $|\mathcal{R}| = 6 < |\mathcal{S}|$ sub-cells.

Fig. 14 Artifact of the performance of the RLE algorithm. For this setting the RLE method performs worse than the QTD, i.e., the number of integration sub-cells is increased, while the IBR algorithm is not adversely affected

for all cases. This can be also seen in Fig. 13d, where the dominance of t_{QTD} is clear due to the convergence of the curves to the value 1. This is to say, the compression increases the time for the construction of the LIM only by a negligible amount.

The difference of the IBR and MRP methods becomes more present in case of highly complex domains. However, for reasonable discretizations in the context of the FCM the topology of discontinuity in the cut cells is less complex and typically it intersects only 2 edges of the cell. In such cases, the IBR and MRP both yield very similar or even identical results [98]. Furthermore, the difference in λ, if present, vanishes for almost all cases, if the IBR and MRP compression techniques are combined with other methods also aiming at a reduction of the IPs. However, it is important to note, that the MRP is based on a complex algorithm that is significantly more cumbersome to implement and time-consuming to execute than the one of the IBR. Since the superiority of the IBR over the RLE is also clearly demonstrated (see "Drawback of the RLE compression technique" section), we recommend the IBR approach for compressing the sub-cells $\mathcal{S}$ resulting from the QTD.

Drawback of the RLE compression technique

Finally, we discuss a special case, where instead of reducing the number of sub-cells of the QTD, the RLE actually increases their number. This is a great example for illustrating that it is not enough to compress the pixel-image, but the resulting rectangular sub-cells $\mathcal{R}$ should definitely have a lower cardinality[8] than the original set of sub-cells $\mathcal{S}$ of the QTD. Figure 14a depicts the QTD with $k = 4$ of a cell being cut in the vicinity of its top-right corner, resulting in a significant difference between the areas of the physical and fictitious domains within the cell. The corresponding pixel-image is shown in Fig. 14b. Although the row-wise RLE compression of the image significantly reduces the number of pixels, with its $|\mathcal{R}| = 20$ sub-cells compared to the QTD having only $|\mathcal{S}| = 13$ sub-cells,[9] it results in an unacceptable value of $\lambda = 20/13 = 1.538 > 1$. Applying the IBR to the same problem avoids this negative artifact and yields a reliable compression with $\lambda < 1$. Taking (i) this robustness of the IBR into account, (ii) its generally better compression (Fig. 13b) as well as (iii) its low additional time costs (Fig. 13c, d), it is definitely superior to the RLE.

[8]In mathematics, the word *cardinality* refers to the number of elements in a given set.
[9]In the remainder of this work, we use $|\cdot|$ to express the cardinality of a given input set.

Combination with other approaches

Several approaches have been published in the wide body of literature aiming at the reduction of the IPs during the AIS. It is straightforwardly possible to combine the compression techniques outlined above with these already existing approaches, in order to achieve an even better reduction of the IPs.

Combination with the reduction of fictitious integration points approach

In order to decrease the number of IPs in the numerical integration Abedian et al. [83,84] propose an approach consisting of two key features. The first feature is based on the idea of *decreasing the number of integration points* (DIP),[10], which exploits the fact, that the integrand over the smaller sub-cells can be seen as a lower order function for which a lower order Gaussian quadrature rule suffices. In other words, the number of IPs per sub-cell is not only chosen depending on the polynomial degree of the shape functions but also on the size of the sub-cell compared to the original cell. The second feature of their approach, which we directly apply to the integration over the compressed sub-cells $\boldsymbol{\mathcal{R}}$, is based on the *reduction of the fictitious integration points* (RFIP)[10]. Thus, the integration over a cut cell is executed in two distinct steps:

1. Integrating over the physical and cut sub-cells, while taking only the physical IPs in the cut sub-cells into account (see Fig. 15a).
2. The contribution of the fictitious domain needs to be considered, in order to avoid adverse effects resulting from ill-conditioning of the system matrices. To this end, a set of IPs is distributed according to the conventional Gaussian quadrature rule in the original cell $\Omega^{(c)}$ and only the points located in the fictitious domain are considered during the numerical integration (see Fig. 15b).

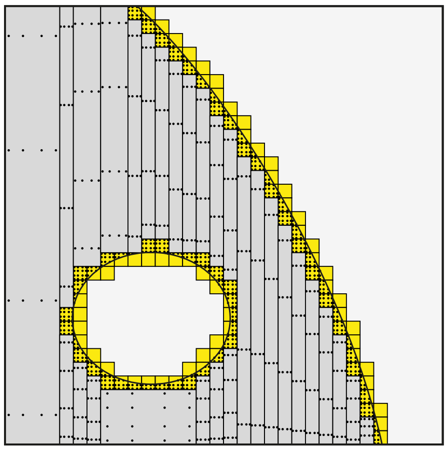
As a result of this algorithm, fictitious sub-cells can be discarded and hence, only the physical sub-cells are taken into account during the compression, slightly reducing the time t_{C}. In the remainder of this work, if the RFIP extension is used, it is indicated by an asterix $\square^{*}$. Although not discussed in this contribution, the RFIP can be easily employed in conjunction with the RLE and MRP techniques as well [106].

Combination with the B-FCM

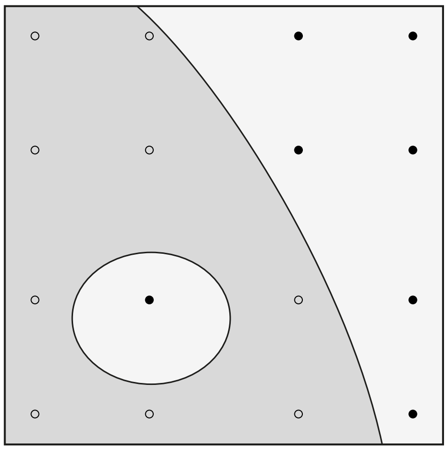
The B-FCM approach by Abedian et al. [45] extends the original FCM by Boolean operations for an improved integration efficiency. Although the B-FCM yields exact results for simple regions for which quadrature formulae exist, the practical use of it lies in its combination with a spacetree-decomposition, in order to handle cut cells with arbitrary topology. In this case, the B-FCM yields exactly the same results as the QTD-based FCM, however, with significantly fewer IPs. In the following, we focus on this latter approach of the B-FCM. The construction of the local mesh is governed by additional features[11] in such a way, that the overall IP-count is reduced. For achieving this, generally overlapping

[10] In the original contribution by Abedian et al. [83,84] these two features have not been given explicit names. However, for the sake of convenience, we have introduced the abbreviations DIP and RFIP to refer to these approaches.

[11] See the mentioned three features on page 7 (883) in [45]. The features 1 and 2 represent conditions (C1 and C2) that has to be evaluated for every parent sub-cell, while the third feature is the already introduced RFIP method. Note that the notations C1, C2 and RFIP do not appear in the original text, it is only our interpretation for referring to those features.



a The IPs in the physical domain are scaled according to Eq. (6), while the ones in the fictitious domain are completely ignored.

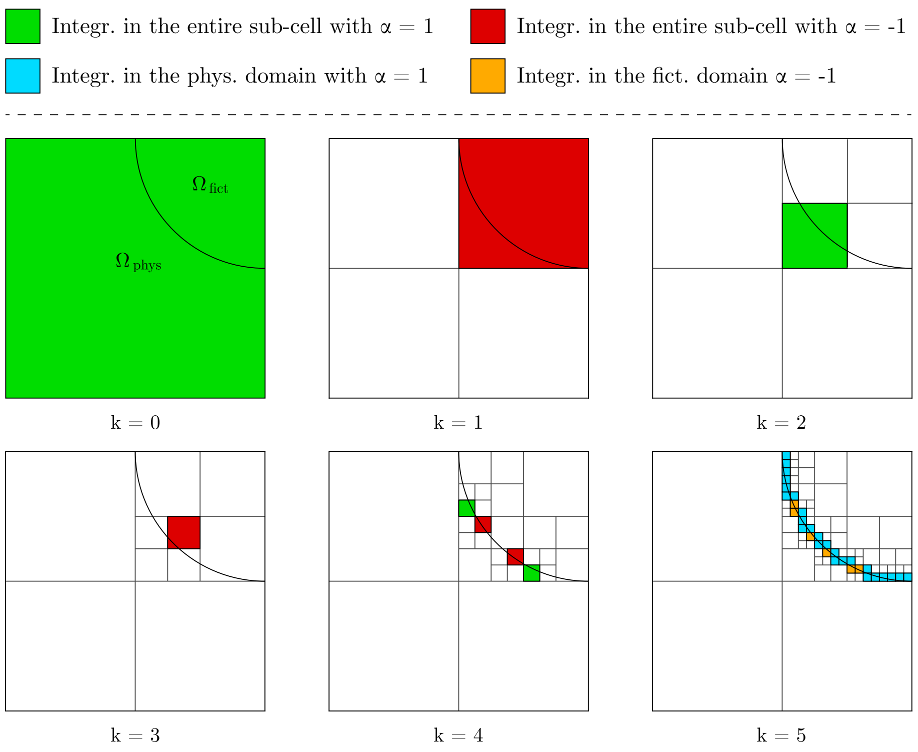


b For computing the contribution of $\Omega_{\text{fict}}^{(c)}$ we distribute $n \times n$ IPs in $\Omega^{(c)}$ and consider only the ones lying in the fictitious domain (filled black circles) according to Eq. (6).

Fig. 15 Combination of the RFIP approach with the IBR method (IBR*). This example is based on the compressed set of sub-cells $\mathcal{R}$ depicted in Fig. 12b where a 4×4-point Gaussian quadrature rule is assumed

sub-cells are created, whose Boolean meta information by the labels $\alpha_{\mathrm{B}} = 1$ and $\alpha_{\mathrm{B}} = -1$ are utilized during numerical integration. For more in depth introduction please see the original work [45]. Throughout the current contribution we refer to such a way of constructing the LIM as *B-QTD*, while the term B-FCM is used for the entire simulation, including both the B-QTD and the corresponding numerical integration. The set of sub-cells resulting from the B-QTD is denoted by $\mathcal{B}$. Due to the fact that the B-FCM yields the same results as the FCM but with a reduced number of IPs, the B-FCM can be directly compared in terms of λ and τ to the C-AIS, although its not based on any compression technique.

It turns out, that the B-FCM performs equally well compared to the C-AIS, however, its performance can be further improved when combined with the compression techniques. Before outlining the compression in the B-FCM, let us briefly recall the basic ideas of the B-QTD based on a simple example depicted in Fig. 16, where $k_{\max} = 5$ is used. On the levels $k < k_{\max}$ we use the green and red colors to depict the *positive* and *negative* sub-cells, corresponding to the labels $\alpha_{\mathrm{B}} = 1$ and $\alpha_{\mathrm{B}} = -1$, respectively. Let us denote the sets of these sub-cells by $\mathcal{B}_{\text{full}}^{+}$ and $\mathcal{B}_{\text{full}}^{-}$. Note that these types of sub-cells are all cut by the boundary, however, during the numerical integration, they will be treated as intact ones free of any discontinuity. The cut sub-cells, for which the discontinuity will be taken into account during the numerical integration, are located at the final level $k = k_{\max}$ of the B-QTD. Here, the blue color represents an integration with $\alpha = 1$ in the physical domain, while the orange color means an integration with $\alpha = -1$ in the fictitious domain. In both cases, the IPs in the other domain are ignored. For these sets we use the labels $\mathcal{B}_{\text{cut}}^{+}$ and $\mathcal{B}_{\text{cut}}^{-}$, respectively. Note that the set of cut sub-cells $\mathcal{B}_{\text{cut}}^{+} \cup \mathcal{B}_{\text{cut}}^{-}$ of the B-QTD is exactly the same as the one of the QTD, however, in case of the B-QTD, the domain of interest of the integration as well as the α-values of the IPs vary from sub-cell to sub-cell. Generally,



🟩 Integr. in the entire sub-cell with α = 1
🟥 Integr. in the entire sub-cell with α = -1
🟦 Integr. in the phys. domain with α = 1
🟧 Integr. in the fict. domain α = -1
Ω fict
Ω phys
k = 0
k = 1
k = 2
k = 3
k = 4
k = 5

Fig. 16 Illustration of the integration sub-cells resulting from the B-QTD. On each refinement level a certain number of sub-cells is created and different labels are assigned indicated by the color-coding: The sets of green ($\boldsymbol{\mathcal{B}}^{+}_{\text{full}}$) and red sub-cells ($\boldsymbol{\mathcal{B}}^{-}_{\text{full}}$) represent integration domains where the discontinuity is not taken into account. The blue ($\boldsymbol{\mathcal{B}}^{+}_{\text{cut}}$) and orange ($\boldsymbol{\mathcal{B}}^{-}_{\text{cut}}$) sub-cells are located at $k = k_{\text{max}} = 5$ exclusively, and they denote integration domains, where the discontinuity is actually taken into account, but only IPs in the physical (blue) or fictitious (orange) domain are considered

it is clear that

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{B}}^{+}_{\text{full}} \cup \boldsymbol{\mathcal{B}}^{-}_{\text{full}} \cup \boldsymbol{\mathcal{B}}^{+}_{\text{cut}} \cup \boldsymbol{\mathcal{B}}^{-}_{\text{cut}}, \tag{18}$$

and that none of these sets have any sub-cells in common. For the color-coding in the above equation, please see Fig. 16. Finally, note that the B-FCM is constructed in such way, that for an integration over a given cell that does not lead to an ill-conditioned stiffness matrix, the contribution of the fictitious domain in the cell still has to be taken into account. This is realized in the B-FCM by utilizing the RFIP approach, discussed in “Combination with the reduction of fictitious integration points approach” section.

Compression steps

Now, a compression can be performed on $\boldsymbol{\mathcal{B}}^{+}_{\text{full}}$ and $\boldsymbol{\mathcal{B}}^{-}_{\text{full}}$, since they both represent sub-domains with continuous integrands.[12] On the other hand, just like before, we do not compress the cut sub-cells, in order to preserve the accuracy of the numerical integration. Compressing the sub-cells on every refinement level is not reasonable or sometimes not even possible, since they often do not form coherent regions (see for example Figs. 16 and 18). Instead, we execute the followings steps, also depicted Fig. 17:

[12] Recall, in these sub-cells the discontinuity is not taken into account.

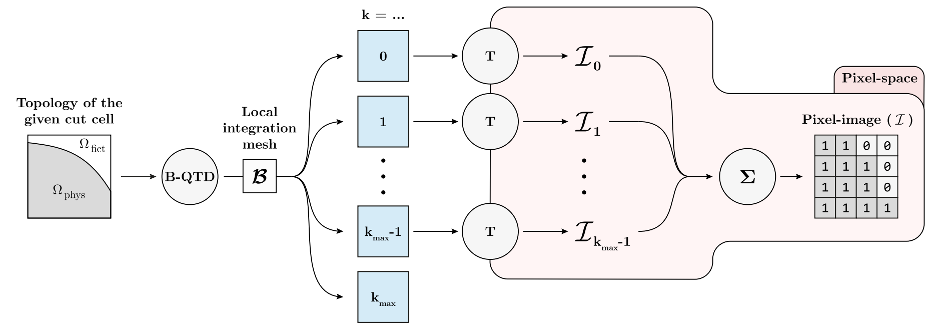


Fig. 17 Procedure of compressing the sub-cells on the layers $k = 0, 1, ..., k_{\max} - 1$ resulting from the B-QTD. The layers are transformed to the corresponding pixel-images $\mathcal{I}_0, \mathcal{I}_1, ..., \mathcal{I}_{k_{\max}-1}$, all having the same resolution $2^{k_{\max}-1} \times 2^{k_{\max}-1}$. After the final pixel image is created, the procedure is identical to the one depicted in Fig. 11

1. Transformation of the individual layers $k_0, k_1, ..., k_{\max} - 1$ by T to the pixel-images $\mathcal{I}_0, \mathcal{I}_1, ..., \mathcal{I}_{k_{\max}-1}$. While the sub-cells of $\boldsymbol{\mathcal{B}}^+_{\text{full}}$ mark pixel regions with the value 1, the sub-cells of $\boldsymbol{\mathcal{B}}^-_{\text{full}}$ result in regions with the value -1. It is important, that each image has the same resolution, which should be defined by the size of the smallest sub-cell participating in the compression, just as before. Consequently, the images should have a resolution of $2^{k_{\max}-1} \times 2^{k_{\max}-1}$ pixels.
2. All images are superimposed and the overlaying pixels-values $(1, 0, -1)$ are added resulting in the image $\mathcal{I} = \sum_{i=0}^{k_{\max}-1} \mathcal{I}_i$. Since (i) no two sub-cells of the same kind can be directly located above each other and (ii) a negative sub-cell can be created only if there is a positive sub-cell already located directly beneath it, the number of positive sub-cells minus the number of negative sub-cells over an arbitrary point is always either 1 or 0. Consequently, the image $\mathcal{I}$ consists of pixels with the values 1 and 0 only.
3. The discussed algorithms can be applied to $\mathcal{I}$, while compressing only the pixel region(s) indicated by the value 1. Since the B-FCM also utilizes the RFIP approach, an additional integration region is introduced in the form of the entire cell, in which we integrate over the fictitious domain with $\alpha << 1$ in order to avoid the possible ill-conditioning of the system matrices.

An example for compressing the sub-cells of the B-QTD with $k_{\max} = 6$ is shown in Fig. 18, where for the physical domain a superellipse-shaped region is chosen. The corresponding implicit function defining this domain is given by Eq. (19) [41].

$$\left[\frac{(x+y)^2}{2}\right]^6 + \left[\frac{2(y-x)^2}{9}\right]^6 \leq 1 \tag{19}$$

Due to the compression, for which in the case of the current example the IBR was used, the originally 60 sub-cells on the levels $k < k_{\max}$ have been significantly reduced to 26 and the overall sub-cell-count has decreased from 273 to 239. As a visualization of the flowchart in Fig. 17, the images created in case of the current example on the refinement levels as well as their sum, are depicted in Fig. 19. Note that for the current case there are no sub-cells created at the levels $k = 0$ and 1. Therefore, only the layers $k = 2 - 5$ are participating in the compression, resulting in the images $\mathcal{I}_2$, $\mathcal{I}_3$, $\mathcal{I}_4$, and $\mathcal{I}_5$, while $\mathcal{I}_0$ and $\mathcal{I}_1$ are empty images. One can see, that the final image $\mathcal{I}$ (Fig. 19e) on which the

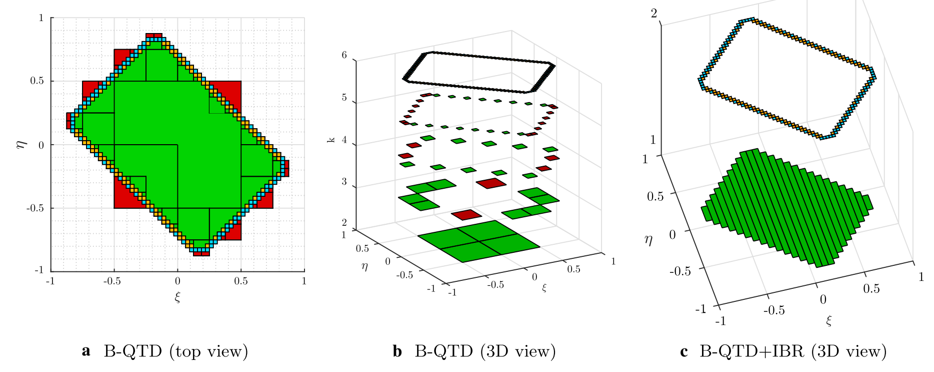


Fig. 18 Compression of the sub-cells resulting from the B-QTD, based on the procedure depicted in Fig. 17. For the given example the IBR compression technique was used

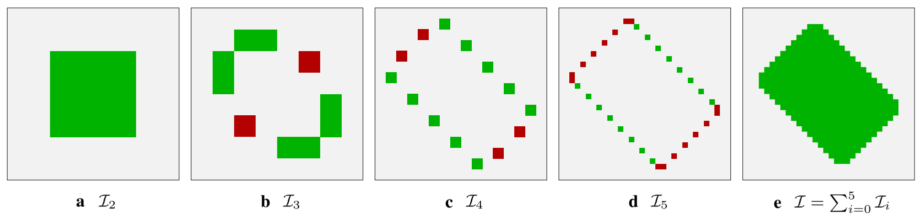


Fig. 19 Creating the resulting image $\mathcal{I}$, on which the compression techniques can be applied. The current figure is based on the example depicted in Fig. 18 and on the procedure illustrated by the flowchart in Fig. 17. In the current case, all images have a resolution of $2^5 \times 2^5$ of pixels, meaning, that in $\mathcal{I}_5$ the single squares correspond to individual pixels. While the green and red pixels mark pixel values of 1 and -1, respectively, the gray background represents pixels with the value 0

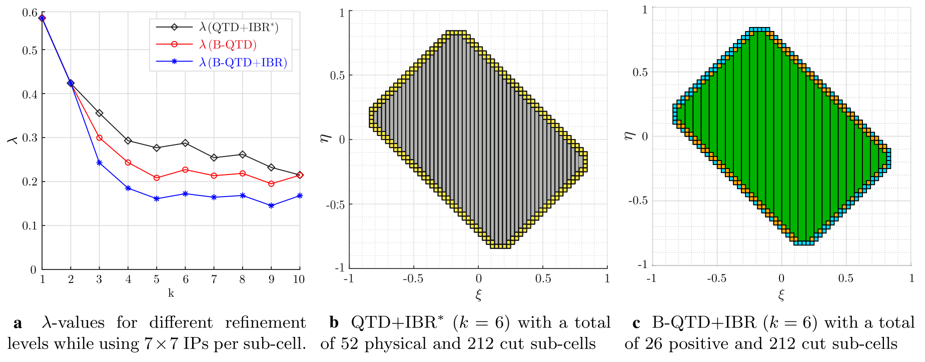
compression algorithms can be applied, indeed contains pixels with values 1 (green) and 0 (grey) exclusively.

Comparison

Sticking to the example presented in Fig. 18, we plot the λ-values for different refinement levels while comparing the QTD-based IBR*, the B-QTD[13] and the B-QTD + IBR[13] approaches in Fig. 20a. According to the results, it is clear, that all methods yield a significant compression by reducing the number of the IPs resulting from the plain QTD to 15–30%. The B-QTD actually outperforms the QTD + IBR* approach, however, its efficiency can be further improved, when combined with the IBR.

Although the QTD + IBR* (Fig. 20b) and B-QTD + IBR (Fig. 20c) yield visually similar results, they are not the same: The B-QTD + IBR results in a smaller number of sub-cells and while in Fig. 20b the physical sub-cells (gray) do not overlap with the cut sub-cells (yellow), in case of Fig. 20c, the positive sub-cells (dark green) are covered by the negative cut sub-cells (red). Furthermore, while in case of the QTD + IBR* the domain of interest is the same for all cut sub-cells during the numerical integration, for the cut sub-cells in

[13]Note that since the B-QTD already incorporates the RFIP approach, indicating this with the asterisk $\square^*$ and writing B-QTD* and B-QTD + IBR* is therefore redundant.



a λ-values for different refinement levels while using 7×7 IPs per sub-cell. **b** QTD+IBR* ($k = 6$) with a total of 52 physical and 212 cut sub-cells **c** B-QTD+IBR ($k = 6$) with a total of 26 positive and 212 cut sub-cells

Fig. 20 Comparison of the different reduction techniques the for the example of the superellipse shaped physical domain defined by Eq. (19)

Fig. 20c, the domain of interest and the sign of the integration changes, depending on which is more effective in a computational sense, resulting in $\mathcal{B}^+_{\text{cut}}$ and $\mathcal{B}^-_{\text{cut}}$.

Although the B-QTD + IBR approach generally yields fewer IPs, it has two disadvantages when compared to the QTD: (i) performing a QTD is computationally less expensive than the B-QTD, since the latter one involves the checking of extra conditions when a parent sub-cell produces four children and (ii) the QTD has a somewhat simpler algorithm, being easier to implement and is already widely used in the field of computational mechanics. Therefore, the best option would be to perform a traditional QTD and transform $\mathcal{S}$ by an extra step to $\mathcal{B}$, i.e., Fig. 20b–c. However, according to our tests, such a direct transition is not possible.

Numerical examples

In this section, we further investigate the approaches IBR*, B-FCM and B-FCM + IBR, that proved to be the most efficient ones, while fully including them into the FCM framework for the computation of problems in linear elasticity. The simulations are performed using our in-house SCM-code written in MATLAB using Lagrangian shape functions based on the Gauss–Lobatto–Legendre (GLL) nodal distribution [67,76,86]. In our investigations, we will vary the cell size, the refinement level k as well as the polynomial degree $p = p_\xi = p_\eta$. For the integration over the (sub-)cells $(p + 1) \times (p + 1)$ IPs are used based on a Gauss–Legendre quadrature rule. If the extended domain is square-shaped, we use n_C to define the number of cells both in x- and y-directions, and thus the total number of IPs grows with $\mathcal{O}(n_c^2 p^2 2^k)$.

Perforated plate

As a first example, we apply the chosen methods to a statically loaded perforated plate depicted in Fig. 21, already investigated in the context of the FCM as a benchmark problem by numerous authors [31,33,39–41,45,77]. The material with a Young's modulus $E = 206.9$ GPa and Poisson's ratio $\nu = 0.29$ is assumed to be linear-elastic and isotropic. Furthermore, the plate is assumed to be in a plane stress state.

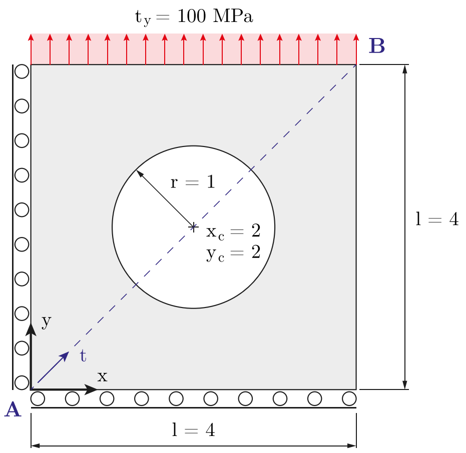


Fig. 21 Setup of the perforated plate under investigation. The given dimensions are understood in (mm)

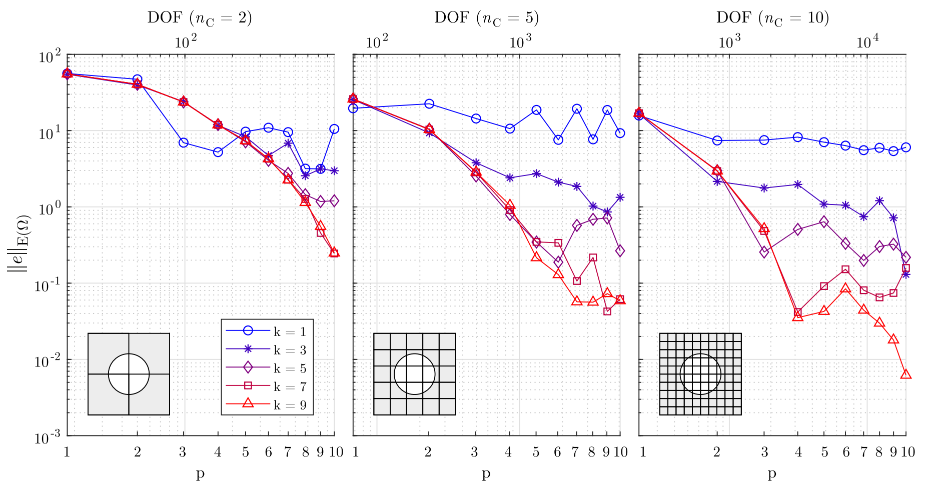


Fig. 22 Error in the energy norm for different settings using the B-FCM + IBR integration approach. Using the other compression techniques as well as using no compression results in the same depicted curves

Quality of the results

First, we evaluate the quality and reliability of the results obtained when using the three methods investigated in this section. In Fig. 22, the errors in the energy norm

$$||e||_{E(\Omega_e)} = \sqrt{\left|\frac{\mathcal{B}(\boldsymbol{u}_{\text{ref}}, \boldsymbol{u}_{\text{ref}}) - \mathcal{B}(\boldsymbol{u}, \boldsymbol{u})}{\mathcal{B}(\boldsymbol{u}_{\text{ref}}, \boldsymbol{u}_{\text{ref}})}\right|} \cdot 100[\%], \tag{20}$$

for various input parameters are presented, which should be minimized by the FCM solution on the *energy space* $E(\Omega_e)$ over the domain Ω_e [3,33]. In Eq. (20), $\boldsymbol{u}$ is the displacement field obtained by the FCM solution and $\boldsymbol{u}_{\text{ref}}$ is the reference solution, obtained by p-FEM using blending functions [113] for an exact geometry mapping, resulting in a strain energy of $1/2 \cdot \mathcal{B}(\boldsymbol{u}_{\text{ref}}, \boldsymbol{u}_{\text{ref}}) = 0.7021812127$ [31]. Besides investigating the global quality of the results based on $||e||_{E(\Omega_e)}$, we also evaluate the solution based on point-wise values of the stress-fields σ_{vM} and σ_{yy} along the diagonal $\overline{AB}$ in Fig. 21, where σ_{vM} is the von Mises stress and σ_{yy} the stress in the y-direction.

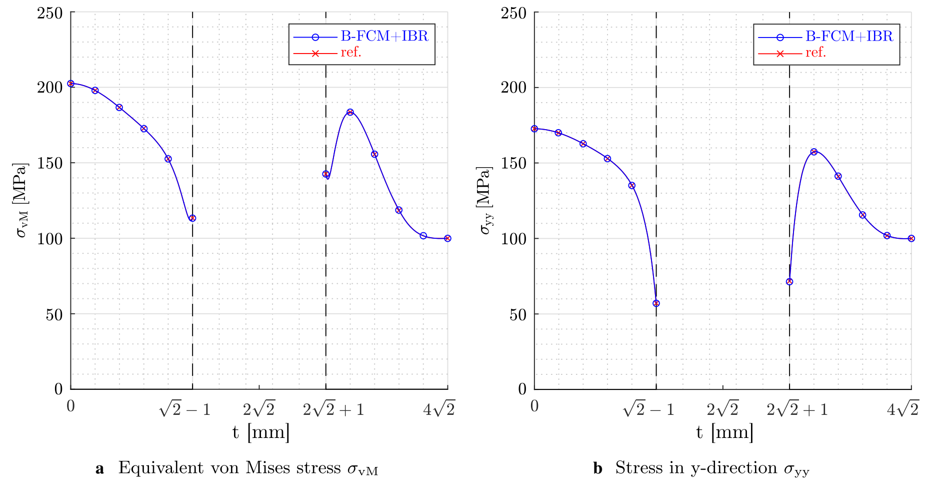


Fig. 23 Stresses along the diagonal cut-line $\overline{AB}$ depicted in Fig. 21 for the B-FCM + IBR approach and for the reference solution. The dashed lines indicate the start and end of the hole along the cut-line. Using the other compression techniques as well as using no compression results in the same depicted curves

The following observations regarding the results are made: First, the depicted global (Fig. 22) and local (Fig. 23) results are the same for all three methods. This indicates that the methods being used indeed only differ in their computational complexities and costs. Thus, we come to the conclusion, that no compromise has to be made between the accuracy and computational effort, when choosing one of the introduced compression techniques. Second, as a verification, the obtained results have been compared to the literature (see convergence-curves for $k = 3$ and $k = 7$ in Refs. [33,41] and stress-curves depicted in Refs. [31,45]) and to a scientific code developed at the Technical University of Munich (FCMLab [114]). In both cases, identical results have been obtained.

Time consumption

Although the investigated methods yield identical solutions, their time requirements need further investigation. The time spent on the integration t_I is in direct correlation with λ, since it is influenced by the number of deployed IPs, i.e., for a given number of sub-cells and IPs, the reduction of t_I depends on the performance of the chosen compression algorithm. In Fig. 24, the λ-values regarding the investigated methods are depicted for different settings in the case of the current example. Similar to what has been depicted in Fig. 20a, it is evident, that the different compression techniques are capable of reaching different stationary λ-values, outlining the highest performance reachable by the compression techniques. The results depicted in Fig. 24 have been computed for $p = 10$ and therefore $(p + 1) \times (p + 1) = 121$ IPs are deployed per sub-cell. Please note that using a lower value of p does not affect the overall trend of the shown curves significantly. If, however, a low polynomial degree ($p = 1, 2$) is employed in conjunction with the RFIP approach, minor deviations might occur due to the fact that only a small number of IPs is distributed within the cut sub-cell.

Note that performing a compression increases the time spent on setting up the local integration mesh t_{LIM}, leading to a decrease of the positive effect of the reduction in compression time t_C. Therefore, for investigating the overall effect of the compression

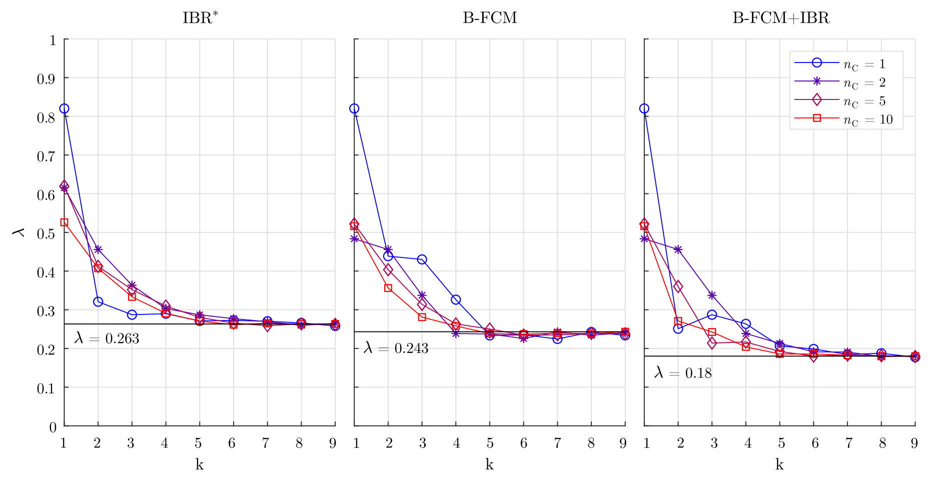


Fig. 24 Compression ratio of the IPs λ of the investigated methods when applied to the perforated plate depicted in Fig. 21. For all curves $p = 10$ was used, resulting in $(p + 1) \times (p + 1) = 121$ IPs per sub-cell

on the integration, we measure the time spent on the cut cells and compute τ according to "Performance" section. The corresponding results are depicted in Fig. 25, where in general, following two domains regarding the compression can be observed:

1. $\tau > 1$: For lower polynomial degrees undesired results are obtained, where the application of the different techniques increases the computation time. This is due to the fact, that when only a few IPs are distributed, their reduction is less significant and therefore, the corresponding reduction in t_I does not outweigh the increase of t_{LIM} caused by the compression procedure.
2. $\tau < 1$: Increasing p results in higher number of distributed IPs per sub-cell, where the effect of the compression becomes more evident, resulting in a significant reduction in the integration time t_I, i.e., the time saved during the integration outweighs the time invested in the compression t_C. The compression ratio τ improves exponentially with p, until the curves reach a stationary value which is specific to the chosen compression algorithm. Note that the refinement level k has a similar effect on τ as p, since it also directly influences λ, as it was demonstrated in Fig. 24. Therefore, regardless of p, lower levels of k result in higher stationary values of τ.

The transition point $\tau = 1$, below which actual time-savings occur, is located for all three methods between $p = 3$ and $p = 4$ (Fig. 25). Therefore, for problems where polynomial degrees $p \geq 4$ are used, a successful reduction in the computational time can be safely expected. Although low refinement levels k may weaken the effect of the reduction, for obtaining optimal convergence in $||e||_{E(\Omega_e)}$, it is reasonable to avoid poor refinement levels in the first place. Note that even if different discretizations are used, both the transition point and the general trend of the τ curves remains unchanged, as demonstrated in Fig. 26 for the case of the B-FCM + IBR. For the perforated plate, the most significant reduction was achieved by the B-FCM combined with the IBR ($\tau = 0.169$), requiring only 69% the computational time needed for the B-FCM ($\tau = 0.245$).

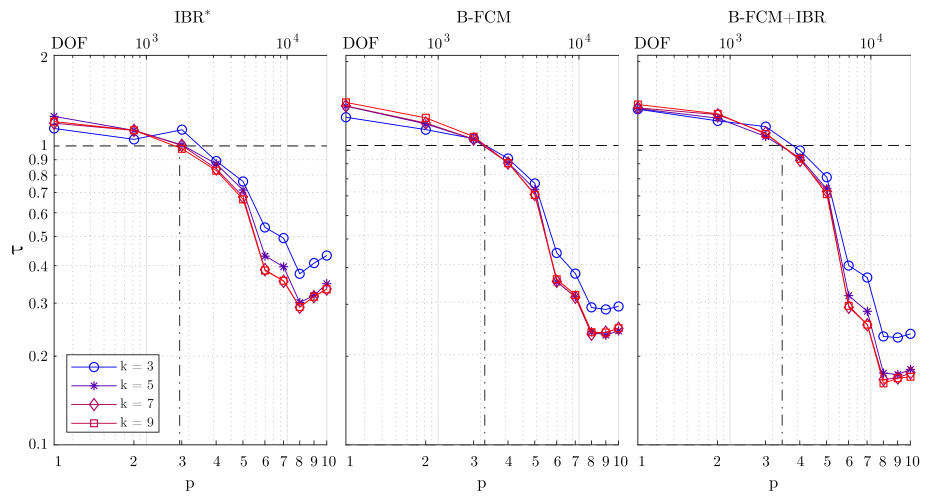


Fig. 25 τ-Values of the investigated methods. The depicted curves are obtained by a discretization of $n_C = 10$ of the problem depicted in Fig. 21

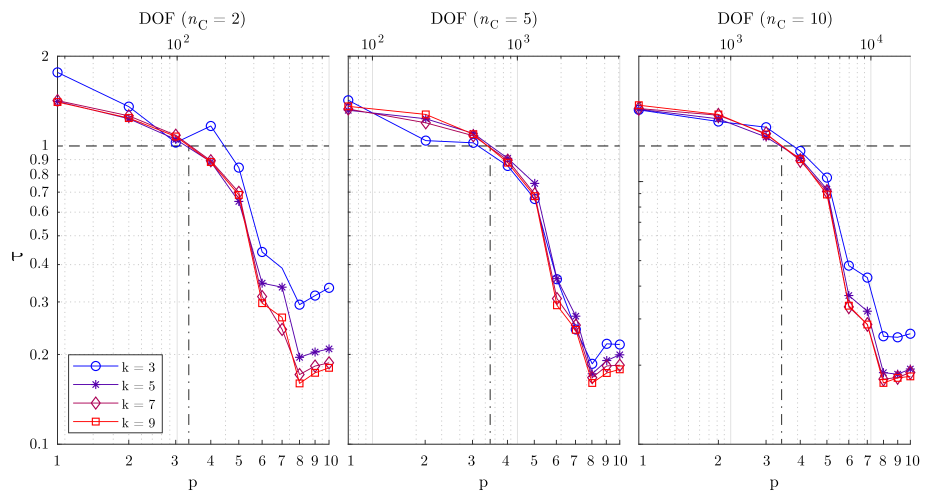


Fig. 26 Time compression ratios τ for the B-FCM + IBR method for various combinations of p, k and n_C

Violin-shaped region

As a second example, we investigate the reduction of the computational effort for the computation of the mode shapes of the violin-shaped domain depicted in Fig 27a. Since the given problem is a dynamic one, the global EQS presented in Eq. (8) is modified to

$$\boldsymbol{M}\ddot{\boldsymbol{U}} + \boldsymbol{K}\boldsymbol{U} = \boldsymbol{F}, \tag{21}$$

where $\boldsymbol{M}$ is the global mass matrix and $\ddot{\boldsymbol{U}}$ is the second derivative of the displacement vector with respect to the time, corresponding to the nodal accelerations. The mass matrices of the individual cells $\boldsymbol{M}^{(c)}$ are computed as

$$\boldsymbol{M}^{(c)} = \int_{-1}^{1}\int_{-1}^{1} \alpha\rho\boldsymbol{N}^T\boldsymbol{N}\det\big(\boldsymbol{J}^{(c)}_{\boldsymbol{\xi}\to\boldsymbol{x}}\big)\,\mathrm{d}\xi\,\mathrm{d}\eta, \tag{22}$$

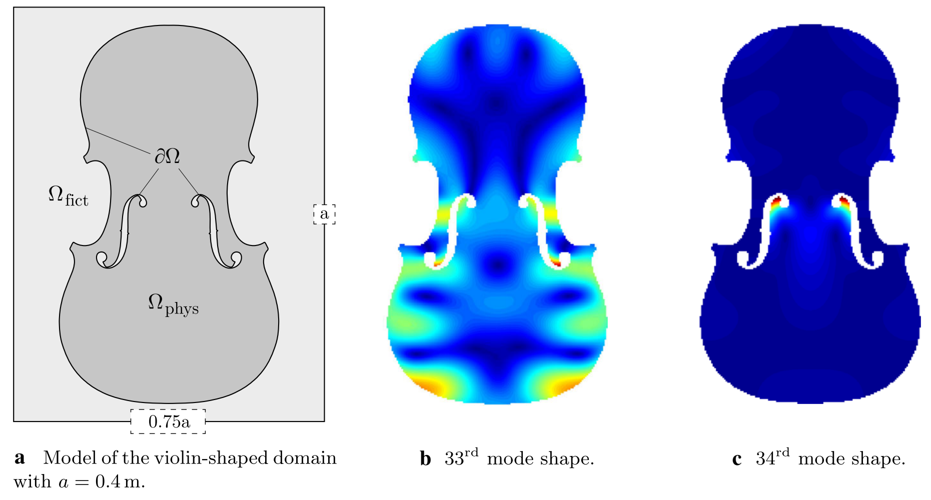


Fig. 27 Violin-shaped domain with two of its mode shapes. The visualization is based on the magnitude of the normalized ($u_{\text{max}} = 1$) displacement vector

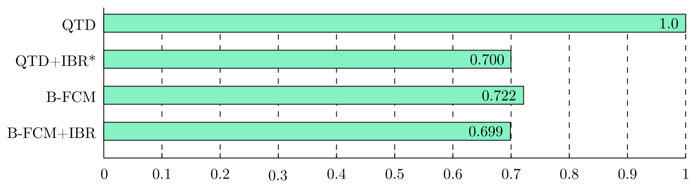


Fig. 28 Normalized computational time needed for the numerical integration of the cut cells (violin-shaped domain)

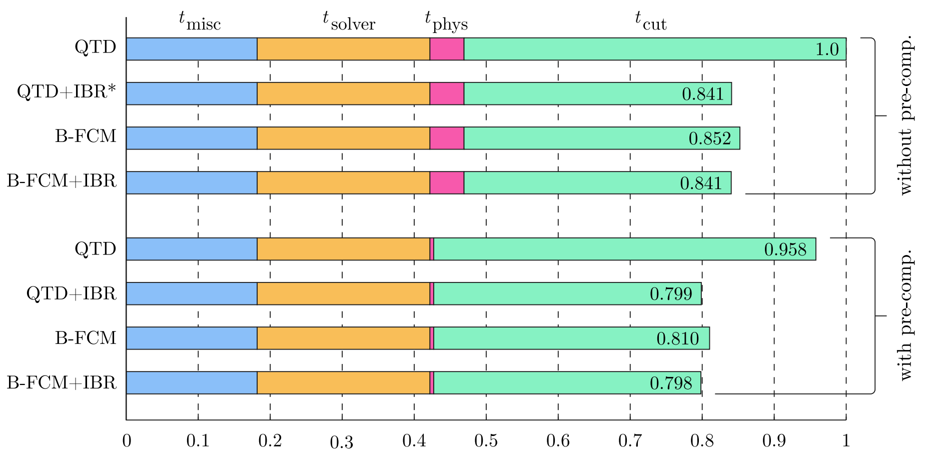


Fig. 29 Time compression ratios τ for the entire processing step. The times t_{phys} and t_{cut} refer to the times spent on the computation of the physical and cut cells, while t_{solver} denotes the time needed for the solution of the eigenvalue problem. Furthermore, the time t_{misc} includes miscellaneous tasks, such as the initialization of the code and assembling of the global matrices, etc

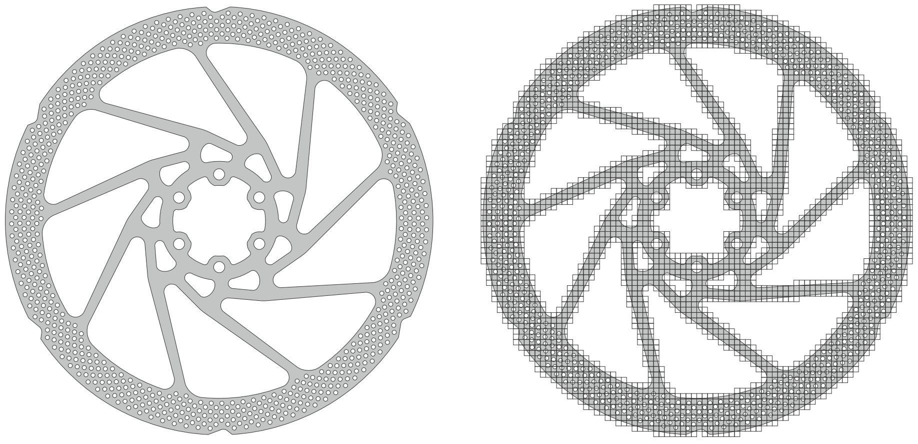


Fig. 30 Disc brake with a diameter of 200 mm and computational mesh used for the FC simulation

where ρ is the density of the physical material. For obtaining the solution, the general eigenvalue problem of Eq. (21) has to be solved, while free boundary conditions are assumed. For comparison, we use the results published by Gravenkamp and Duczek [115]. According to their simulation, we assume a linear isotropic material with the following properties: $E = 12.048$ GPa, $\nu = 0.4$ and $\rho = 470$ kg/m^3 as well as the plane stress state. Furthermore, the embedding domain is discretized with $192 \times 256 = 49,152$ cells, and the polynomial degree of the shape function is set to $p = 4$. For capturing the boundary, a refinement level of $k = 5$ is used. Using our FC-simulation, with and without compression, identical results (see Fig. 27b, c) were obtained as the ones given in Ref. [115].

The τ-values obtained for this example are depicted in Fig. 28. Considering the chosen set up of the numerical model, the computational time that is dominantly spent on the integration of the cut cells was successfully reduced by approximately 30% compared to the QTD-based AIS, while maintaining the same accuracy. It is already a significant reduction, even though the applied compression methods are more efficient for higher polynomial degrees, as discussed in the previous section (see Fig. 25). This is due to the already mentioned fact, that for lower polynomial degrees fewer IPs are distributed in each cell and therefore, the effect of the compression is less pronounced. Note that this phenomenon does not only affect the C-AIS, but also the B-FCM.

For getting a clearer insight regarding the global effect of the compression, let us now evaluate τ for the entire processing step composed of the computation and assembly of the cell matrices as well as the solution of the eigenvalue problem. Since the free vibrations of the violin are investigated, no Dirichlet BCs are applied to the system. The corresponding results are depicted in Fig. 29, where also the effect of the pre-computation of the stiffness and mass matrices of the physical cells, made possible by using a Cartesian mesh, is included. It is clear, that the pre-computation of the physical cell matrices and the reduction of the IPs in the cut cells affect exclusively the times t_{phys} and t_{cut}, respectively, the time requirement of the rest of simulation remains unchanged. Although there are significantly more physical cells (22,863) than cut ones (1245) during the simulation, t_{phys} is only 8.9% of t_{cut}. This result can be explained by two facts: (i) the cut cells contain a total of

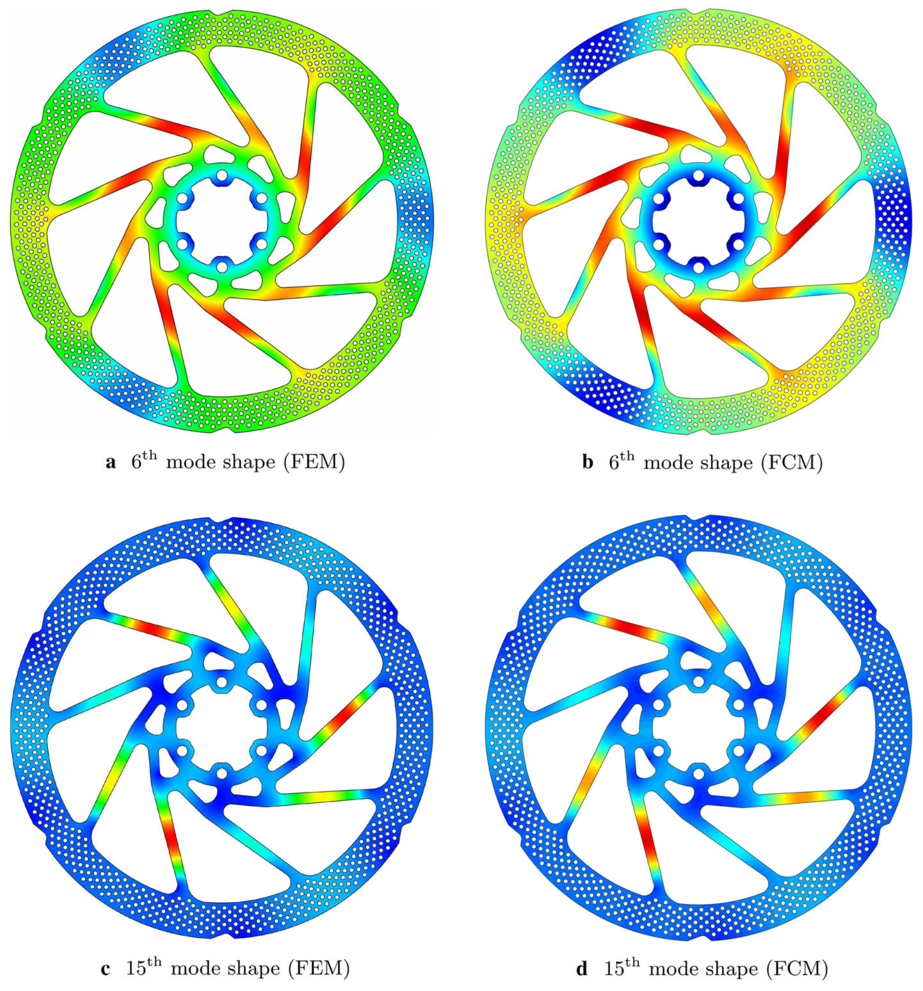

a 6th mode shape (FEM)
b 6th mode shape (FCM)
c 15th mode shape (FEM)
d 15th mode shape (FCM)

Fig. 31 Visualization of the 6th (**a**, **b**) and 15th mode shapes (**c**, **d**) with the eigenfrequencies 16,796 Hz and 26,651 Hz, respectively. The color indicates the magnitude of the normalized displacement vector ($u_{\max} = 1$)

123,975 sub-cells in the uncompressed state, each of which having the same computational cost during the numerical integration as a single physical cell and (ii) t_{cut} includes both t_{I}, and t_{LIM}. As a result, the pre-computation improved the overall run-time only by 4%. Furthermore, although t_{cut} is significantly larger than t_{phys}, for the chosen combination of p and k, it represents only 53% of the total processing time. Consequently, the approximately 30% reduction in t_{cut} (see Fig. 28) results without pre-computation roughly in 15% and with pre-computation in approximately 20% reduction of the processing-time, which is still a significant number. Especially, if the straightforward implementation of the IBR* and the lossless compression which results in the same accuracy is taken into account. Comparing the methods to each other, the QTD + IBR* has performed better than the B-FCM and did just as well as the B-FCM + IBR.

Disc brake

Finally, we investigate the performance of the compression techniques for the modal analysis of a disc brake depicted in Fig. 30a. The geometry is freely in the internet [116].

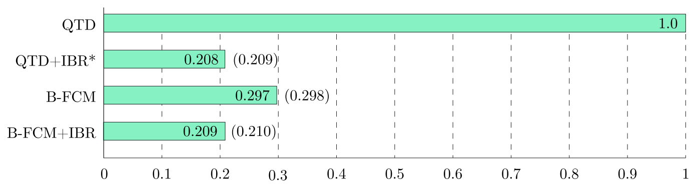


Fig. 32 Time compression ratios for the computation of the cell matrices (values inside the bars) and for the entire processing step (values outside the bars)

The original model is a three-dimensional thin-walled plate, from which the top surface is extracted for our two-dimensional simulation. For the simulation, a material with a Young's modulus of $E = 200$ GPa, a Poisson's ratio of $\nu = 0.29$ and density of $\rho = 7800$ kg/m^3 is used. A reference solution of the problem was obtained by the commercial FE-software Abaqus using an unstructured mesh consisting of 278,610 quadratic quadrilateral elements,[14] resulting in a total of 1,007,147 DOFs. For the FC simulation, the extended domain was discretized by $80 \times 80 = 6400$ cells, from which 344 were purely physical and 2694 are cut (see Fig. 30b). To ensure highly accurate results and thus to compensate for the rather coarse mesh, a polynomial degree $p = 9$ and a refinement level $k = 8$ was used during the simulation, resulting in a total of 503,574 DOFs. Both the computed eigenfrequencies and the mode shapes were in a good agreement with the reference solution. A visual comparison of the 6th and 15th mode shapes are depicted in Fig. 31.

The compression ratio τ for the different approaches is depicted in Fig. 32, where τ is evaluated both for the computation of the cut cell matrices (value written within the bars) and for the total processing (value in parenthesis outside of the bars). Unlike in case of the previous example, due to the high value of p in combination with a sufficient refinement level ($k \geq 5$), a significant compression was possible, requiring only 20–30% of the original time. While in case of the previous example the QTD + IBR* was only slightly better than the B-FCM, for the current example, the QTD + IBR* ($\tau = 0.209$) was 30% faster than the B-FCM ($\tau = 0.298$). Although the B-FCM + IBR has lead to the same time saving, its implementation is definitely more cumbersome than the one of the QTD + IBR*. Furthermore, due to the significantly larger number of cut cells than physical ones (again, unlike in the case of the previous example), the computation of the cut cells is entirely dominating the processing time. Consequently, reducing t_{cut} has a major influence on the processing time, so much so, that τ for the cut cells and τ for the entire processing step are basically equal.

Conclusion

In this contribution, the compressed adaptive integration scheme (C-AIS) was introduced and investigated, which extends the traditional AIS by compressing the sub-cells resulting from the quadtree-decomposition (QTD). The additional compression step results in significantly decreased computational costs as the number of integration sub-cells is notably

[14]In Abaqus these elements are referred to as CPS8 elements indicating that conventional continuum elements under plane stress conditions with 8 nodes are used. The shape functions of an 8-node quadrilateral finite element are based on the Serendipity family.

reduced. Furthermore, if only polynomial shape functions are considered in conjunction with Cartesian meshes, the accuracy is identical to the conventional AIS.

In "Compression techniques", "Comparison", "Drawback of the RLE compression technique" section, different compression techniques were introduced and tested. The image-block representation (IBR) approach turned out to be the most suitable candidate due to its efficiency, simplicity and robustness. It was shown in "Combination with other approaches" section, that the C-AIS can be coupled with the reduction of fictitious integration points (RFIP) and Boolean finite cell method (B-FCM) approaches as well, leading to even fewer IPs and consequently, decreased computational times. Then, in "Numerical examples" section, the C-AIS was successfully embedded in the FCM framework and applied to two-dimensional problems of elastostatics and modal analysis. By means of several numerical examples, the IBR*, B-FCM, and B-FCM + IBR approaches have been investigated in detail. All of these methods yielded accurate results with significantly less computational effort. Since the accuracy of these methods is basically identical, the focus was placed on the comparison of their computational time which is influenced by two factors:

1. Time investment for setting up an appropriate local integration mesh, which depends on the speed of the partitioning and compression algorithms.
2. Time saved during integration over the resulting sub-cells, which depends on the total number of integration points in the broken cells, determined by the number of sub-cells and distributed integration points within them.

In "Time consumption" section, it was shown, that the polynomial order of the shape functions heavily influences whether the compression algorithm renders a meaningful reduction of computational time. A low polynomial degree (in our code $p < 4$) may increase the computational time. However, for higher order shape functions and reasonable refinement levels, which are typically used in the FCM, a significant reduction can be achieved. In less ideal cases ("Violin-shaped region" section), the time spent on the cut sub-cells can be compressed down to 70% and the time of the entire simulation to 80%, while in ideal cases that are more suitable for the FCM ("Disc brake" section), a significant reduction was achieved requiring only 20% of the computational time for the entire simulation.

Since the investigated methods all yield the same accuracy, no compromise between the quality of the solution and the computational time has to be made. Therefore, one may choose the simplest and most efficient approach. Although the B-FCM methodology, which can be further enhanced by the combination with the IBR compression technique, saves a significant amount of time, it requires major modifications in the already existing codes regarding the construction of the local integration mesh (LIM) as well as the compression and integration over the Boolean sub-cells. On the other hand, the IBR*, which in 2 of the 3 numerical examples turned out to be just as efficient as the B-FCM + IBR, requires a QTD only, which is already widely used and implemented in the context of the FCM and other fictitious domain approaches. Also the integration over the compressed sub-cells can be carried out without modification in the pre-existing code. Therefore, we recommend the C-AIS with the IBR* approach, both for the FCM and for other numerical methods as well, where a QTD is performed for integration purposes.

Outlook

In this work, we have investigated the image-compression techniques for two-dimensional problems. Consequently, future work could include the extension of the introduced compression techniques to three-dimensional problems, where instead of a pixel image we deal with a sequence of images resulting in a voxelated data set which can be derived from an initial OTD. This should be fairly straightforward for the RLE and IBR methods, since they only require an additional scanning and merging directions. The RLE would scan the run-lengths along the x-, y- or z-direction, increasing its time complexity to $\mathcal{O}(n^3)$. Then, the IBR can be carried out in two steps, assuming the primary RLE was carried out in i-direction: it first merges the line segments in the ij-plane, then further unites the merged blocks, if possible, by comparing the adjacent jk-planes, each of which being in a two-dimensional IBR-compressed state. Although the three-dimensional compression requires more time t_C than a two-dimensional one, the number of IPs also grows at a cubic-rate and therefore, a significant portion of the time t_I can be saved when a meaningful compression of the sub-cells is achieved. The extension of the MRP to three-dimension is arguable: Not only that in this case the intersection graph (if it can be constructed in the first place) is not bipartite any more and thus König's theorem does not apply, it is also questionable, whether it is worth the effort, based on the good performance and easy implementation of the IBR we have seen in two-dimensions. Finally, the B-FCM is expected to perform similar compared to the C-AIS in thee-dimensions as well. However, it is yet to be investigated, how these different approaches perform against each other in that case.

Future work might also include the investigation of the C-AIS in conjunction with other fictitious domain approaches, where the integrand in Eq. (1) is not a polynomial function. This is the case, e.g., in the poly-FCM [89], where due to the rational shape functions and non-affine geometry mapping of the polygonal elements, the integrand in Eq. (1) is a rational function. Applying Gaussian quadrature rules to rational functions inevitably leads to integration errors, which further increase when due to the compression of the sub-cells a reduced number of integrations points is used for the numerical integration. Further research might reveal whether the typical convergence rates can be maintained in the poly-FCM when combined with the C-AIS for reducing the computational time. Another interesting area of application for the C-AIS is seen in unfitted triangular and tetrahedral meshes, e.g., CutFEM [24] and TetFCM [11,69,70]. In this case, as long as a sub-parametric geometry mapping preserving the straight edges and faces of the elements is used in conjunction with a polynomial approximation of the field variables [70] the integrand in Eq. (1) remains polynomial. Hence, the C-AIS can be applied without loss of accuracy compared to the conventional AIS.

The shown advantages of the C-AIS become even more significant, when applied to non-linear problems or p-refinement strategies. In these cases, following the reasoning put forward in Ref. [61] for the moment fitting approach, the compression has to be performed only once for each cut cell, while the compressed sub-cells $\mathcal{R}$ can be reused in every iteration loop, making the computational overhead t_C vanish for even lower polynomial degrees (cf. Fig. 25).

Abbreviations

AIS: Adaptive integration scheme; BC: Boundary condition; B-FCM: Boolean finite cell method; C-AIS: Compressed adaptive integration scheme; DIP: Decreasing integration points; DOF: Degrees of freedom; DT: Divergence theorem; EP: Equivalent polynomial; EQS: Equation system; FC: Finite cell; FCM: Finite cell method; FE : Finite element; FEM: Finite element method; IBR: Image block representation; IBR*: IBR combined with the RFIP; IP: Integration point; LIM: Local integration mesh; MF: Moment fitting; MRP: Minimal rectangular partition; OTD: Octree-decomposition; QTD: Quadtree-decomposition; RFIP: Reduction of fictitious integration points; RLE: Run-length encoding.

Acknowledgements

The authors thank the German Research Foundation (DFG) for its financial support under Grant DU 1904/2-1. The authors also acknowledge the many fruitful discussions with Prof. Juhre (Otto von Guericke University Magdeburg) and Prof. Düster (Hamburg University of Technology) regarding the interpretation of the numerical results.

Authors' contributions

The in-house Matlab code which is based on the Spectral Cell Method was developed by SE and extended to image-compression based numerical integration, as proposed in this article, by MP. The application of this approach to the presented numerical examples was jointly investigated by MP and FD. The article is mainly written by MP, while for the theoretical background as well as for the writing, discussions and comments provided by SE and FD were essential for the quality of the work. All authors read and approved the final manuscript.

Funding

The first author is financed by the German Research Foundation (DFG) under Grant DU 1904/2-1.

Availability of data and materials

The datasets used and/or analysed during the current study are available from the corresponding author on reasonable request.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Institute of Mechanics, Otto von Guericke University, Magdeburg, Germany, [2]School of Civil and Environmental Engineering, University of New South Wales, Sydney, Australia.

Received: 11 November 2019 Accepted: 16 April 2020
Published online: 06 May 2020

References

1. Hutton DV. Fundamentals of finite element analysis. New York: McGraw-Hill Science/Engineering/Math; 2003.
2. Zienkiewicz OC, Taylor RL, Zhu JZ. The finite element method: its basis and fundamentals. Oxford: Butterworth-Heinemann; 2005.
3. Szabó B, Babuška I. Introduction to finite element analysis. Hoboken: Wiley; 2011.
4. Del Pino S, Pironneau O. A fictitious domain based general PDE solver. Numerical methods for scientific computing variational problems and applications. 2003.
5. Ramière I, Angot P, Belliard M. A fictitious domain approach with spread interface for elliptic problems with general boundary conditions. Comput Methods Appl Mech Eng. 2007;196(4–6):766–81.
6. Parussini L. Fictitious domain approach via lagrange multipliers with least squares spectral element method. J Sci Comput. 2008;37(3):316–35.
7. Parussini L, Pediroda V. Fictitious domain approach with hp-finite element approximation for incompressible fluid flow. J Comput Phys. 2009;228(10):3891–910.
8. Hansbo A, Hansbo P. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. Comput Methods Appl Mech Eng. 2002;191(47–48):5537–52.
9. Roma AM, Peskin CS, Berger MJ. An adaptive version of the immersed boundary method. J Comput Phys. 1999;153(2):509–34.
10. Dauge M, Düster A, Rank E. Theoretical and numerical investigation of the finite cell method. J Sci Comput. 2015;65(3):1039–64.
11. Duczek S, Duvigneau F, Gabbert U. The finite cell method for tetrahedral meshes. Finite Elem Anal Des. 2016;121:18–32.
12. Sukumar N, Moës N, Moran B, Belytschko T. Extended finite element method for three-dimensional crack modelling. Int J Numer Methods Eng. 2000;48(11):1549–1570, 8.
13. Sukumar N, Chopp DL, Moës N, Belytschko T. Modeling holes and inclusions by level sets in the extended finite-element method. Comput Methods Appl Mech Eng. 2001;190(46–47):6183–200.
14. Cheng KW, Fries T-P. Higher-order XFEM for curved strong and weak discontinuities. Int J Numer Methods Eng. 2009;82:564–90.
15. Fries T-P, Belytschko T. The extended/generalized finite element method: an overview of the method and its applications. Int J Numer Methods Eng. 2010;84:253–304.

16. Strouboulis T, Copps K, Babuška I. The generalized finite element method. Comput Methods Appl Mech Eng. 2001;190(32–33):4081–193.
17. Strouboulis T, Zhang L, Babuška I. Assessment of the cost and accuracy of the generalized FEM. Int J Numer Methods Eng. 2006;69(2):250–83.
18. Babuška I, Banerjee U. Stable generalized finite element method (SGFEM). Comput Methods Appl Mech Eng. 2012;201–204:91–111.
19. Melenk JM, Babuška I. The partition of unity finite element method: basic theory and applications. Comput Methods Appl Mech Eng. 1996;139(1–4):289–314.
20. Babuška I, Melenk JM. The partition of unity method. Int J Numer Methods Eng. 1997;40(4):727–58.
21. Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: I. A stabilized lagrange multiplier method. Comput Methods Appl Mech Eng. 2010;199(41–44):2680–6.
22. Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: II. A stabilized nitsche method. Appl Numer Math. 2012;62(4):328–41.
23. Burman E, Hansbo P. Fictitious domain methods using cut elements: III. A stabilized nitsche method for stokes' problem. ESAIM: Math Model Numer Anal. 2014;48(3):859–74.
24. Burman E, Claus S, Hansbo P, Larson MG, Massing A. CutFEM: Discretizing geometry and partial differential equations. Int J Numer Methods Eng. 2014;104(7):472–501.
25. García-Ruíz MJ, Steven GP. Fixed grid finite elements in elasticity problems. Eng Comput. 1999;16(2):145–64.
26. Maan FS, Querin OM, Barton DC. Extension of the fixed grid finite element method to eigenvalue problems. Adv Eng Softw. 2007;38(8–9):607–17.
27. Daneshmand F, Kazemzadeh-Parsi MJ. Static and dynamic analysis of 2d and 3d elastic solids using the modified FGFEM. Finite Elem Anal Des. 2009;45(11):755–65.
28. Kim H, García MJ, Querin OM, Steven GP, Xie YM. Introduction of fixed grid in evolutionary structural optimisation. Eng Comput. 2000;17(4):427–39.
29. Woon SY, Querin OM, Steven GP. On improving the GA step-wise shape optimization method through the application of the fixed grid FEA paradigm. Struct Multidiscip Optim. 2003;25(4):270–8.
30. Nadal E, Ródenas JJ, Albelda J, Tur M, Tarancón JE, Fuenmayor FJ. Efficient finite element methodology based on cartesian grids: application to structural shape optimization. Abstract Appl Anal. 2013;1–19:2013.
31. Parvizian J, Düster A, Rank E. Finite cell method: *h*- and *p*-extension for embedded domain problems in solid mechanics. Comput Mech. 2007;41(1):121–33.
32. Düster A, Parvizian J, Yang Z, Rank E. The finite cell method for three-dimensional problems of solid mechanics. Comput Methods Appl Mech Eng. 2008;197(45–48):3768–82.
33. Düster A, Rank E, Szabó B. The *p*-version of the finite element and finite cell methods. Encyclopedia of Computational Mechanics. 2017. p. 1–35.
34. Schillinger D, Cai Q, Mundani R-P, Rank E. A review of the finite cell method for nonlinear structural analysis of complex CAD and image-based geometric models., Lecture notes in computational science and engineeringBerlin: Springer; 2013. p. 1–23.
35. Schillinger D, Ruess M. 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. 2014;22(3):391–455.
36. Ruess M, Tal D, Trabelsi N, Yosibash Z, Rank E. The finite cell method for bone simulations: verification and validation. Biomech Model Mechanobiol. 2011;11(3–4):425–37.
37. Zander N, Kollmannsberger S, Ruess M, Yosibash Z, Rank E. The finite cell method for linear thermoelasticity. Comput Math Appl. 2012;64(11):3527–41.
38. Ruess M, Schillinger D, Bazilevs Y, Varduhn V, Rank E. Weakly enforced essential boundary conditions for NURBS-embedded and trimmed NURBS geometries on the basis of the finite cell method. Int J Numer Methods Eng. 2013;95(10):811–46.
39. Abedian A, Düster A. Equivalent legendre polynomials: numerical integration of discontinuous functions in the finite element methods. Comput Methods Appl Mech Eng. 2019;343:690–720.
40. Abedian A, Parvizian J, Düster A, Khademyzadeh H, Rank E. Performance of different integration schemes in facing discontinuities in the finite cell method. Int J Comput Methods. 2013;10(03):1350002.
41. Joulaian M. The hierarchical finite cell method for problems in structural mechanics. Ph.D. thesis, Hamburg Technical University. 2017.
42. Yang Z. The finite cell method for geometry-based structural simulation. Ph.D. thesis, Technical University of Munich. 2011.
43. Yang Z, Kollmannsberger S, Düster A, Ruess M, Garcia EG, Burgkart R, Rank E. Non-standard bone simulation: interactive numerical analysis by computational steering. Comput Vis Sci. 2011;14(5):207–16.
44. Yang Z, Ruess M, Kollmannsberger S, Düster A, Rank E. An efficient integration technique for the voxel-based finite cell method. Int J Numer Methods Eng. 2012;91(5):457–71.
45. Abedian A, Düster A. An extension of the finite cell method using boolean operations. Comput Mech. 2017;59(5):877–86.
46. Legrain G, Moës N. Adaptive anisotropic integration scheme for high-order fictitious domain methods: application to thin structures. Int J Numer Methods Eng. 2018;114(8):882–904.
47. Fries T-P, Omerović S. Higher-order accurate integration of implicit geometries. Int J Numer Methods Eng. 2015;106(5):323–71.
48. Hughes TJR. The finite element method: linear static and dynamic finite element analysis. New York: Dover Pubn Inc.; 2000.
49. Kudela L. Highly accurate subcell integration in the context of the finite cell method. Master's thesis, Technical University of Munich. 2013.
50. Kudela L, Zander N, Bog T, Kollmannsberger S, Rank E. Efficient and accurate numerical quadrature for immersed boundary methods. Adv Model Simul Eng Sci. 2015;2(1):10.

51. Királyfalvi Gy, Szabó B. Quasi-regional mapping for the *p*-version of the finite element method. Finite Elem Anal Des. 1997;27(1):85–97.
52. Kudela L, Zander N, Kollmannsberger S, Rank E. Smart octrees: accurately integrating discontinuous functions in 3D. Comput Methods Appl Mech Eng. 2016;306:406–26.
53. Guichard D, et al. Single and multivariable calculus: early transcendentals. In: Guichard D. Creative commons attribution—Noncommercial—Share Alike 3.0. 2018.
54. Riley KF, Hobson MP, Bence SJ. Mathematical methods for physics and engineering: a comprehensive guide. Cambridge: Cambridge University Press; 2006.
55. Dasgupta G. Integration within polygonal finite elements. J Aerosp Eng. 2003;16(1):9–18.
56. Sudhakar Y, Moitinho de Almeida JP, Wall WA. An accurate, robust, and easy-to-implement method for integration over arbitrary polyhedra: application to embedded interface methods. J Comput Phys. 2014;273:393–415.
57. Duczek S, Gabbert U. Efficient integration method for fictitious domain approaches. Comput Mech. 2015;56(4):725–38.
58. Mousavi SE, Xiao H, Sukumar N. Generalized Gaussian quadrature rules on arbitrary polygons. Int J Numer Methods Eng. 2009;82:99–113.
59. Xiao H, Gimbutas Z. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. Comput Math Appl. 2010;59(2):663–76.
60. Müller B, Kummer F, Oberlack M. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. Int J Numer Methods Eng. 2013;96(8):512–28.
61. Joulaian M, Hubrich S, Düster A. Numerical integration of discontinuities on arbitrary domains based on moment fitting. Comput Mech. 2016;57(6):979–99.
62. Mousavi SE, Sukumar N. Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons. Comput Mech. 2010;47(5):535–54.
63. Sudhakar Y, Wall WA. Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods. Comput Methods Appl Mech Eng. 2013;258:39–54.
64. Hubrich S, Joulaian M, Düster A. Numerical integration in the finite cell method based on moment-fitting. In: Proceedings of 3rd ECCOMAS young investigators conference, 6th GACM Colloquium, Aachen, Germany. 2015. p. 1–4.
65. Ventura G, Benvenuti E. Equivalent polynomials for quadrature in heaviside function enriched elements. Int J Numer Methods Eng. 2014;102(3–4):688–710.
66. Ventura G. On the elimination of quadrature subcells for discontinuous functions in the eXtended finite-element method. Int J Numer Methods Eng. 2006;66(5):761–95.
67. Duczek S. Higher order finite elements and the fictitious domain concept for wave propagation analysis. VDI Fortschritt-Berichte Reihe 20 Nr. 458. 2014. https://opendata.uni-halle.de/handle/1981185920/11873.
68. Mossaiby F, Joulaian M, Düster A. The spectral cell method for wave propagation in heterogeneous materials simulated on multiple GPUs and CPUs. Comput Mech. 2018;63(5):805–19.
69. Xu F, Schillinger D, Kamensky D, Varduhn V, Wang C, Hsu M-C. The tetrahedral finite cell method for fluids: immersogeometric analysis of turbulent flow around complex geometries. Comput Fluids. 2016;141:135–54.
70. Varduhn V, Hsu M-C, Ruess M, Schillinger D. The tetrahedral finite cell method: Higher-order immersogeometric analysis on adaptive non-boundary-fitted meshes. Int J Numer Methods Eng. 2016;107(12):1054–79.
71. Nagaraja S, Elhaddad M, Ambati M, Kollmannsberger S, De Lorenzis L, Rank E. Phase-field modeling of brittle fracture with multi-level *hp*-FEM and the finite cell method. Comput Mech. 2018;63(6):1283–300.
72. Sehlhorst H-G. Numerical homogenization strategies for cellular materials with applications in structural mechanics. Ph.D. thesis. Technical University Hamburg-Harburg. 2011.
73. Düster A, Sehlhorst H-G, Rank E. Numerical homogenization of heterogeneous and cellular materials utilizing the finite cell method. Comput Mech. 2012;50(4):413–31.
74. Heinze S, Joulaian M, Düster A. Numerical homogenization of hybrid metal foams using the finite cell method. Comput Math Appl. 2015;70(7):1501–17.
75. Elhaddad M, Zander N, Bog T, Kudela L, Kollmannsberger S, Kirschke J, Baum T, Ruess M, Rank E. Multi-level *hp*-finite cell method for embedded interface problems with application in biomechanics. Int J Numer Methods Biomed Eng. 2018;34(4):e2951.
76. Duczek S, Liefold S, Gabbert U. The finite and spectral cell methods for smart structure applications: transient analysis. Acta Mech. 2014;226(3):845–69.
77. Parvizian J, Düster A, Rank E. Topology optimization using the finite cell method. Optim Eng. 2011;13(1):57–78.
78. Schillinger D, Kollmannsberger S, Mundani RP, Rank E. The finite cell method for geometrically nonlinear problems of solid mechanics. In: IOP conference series: materials science and engineering, vol. 10. 2010. p. 012170.
79. Schillinger D, Dedè L, Scott MA, Evans JA, Borden MJ, Rank E, Hughes TJR. 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 Eng. 2012;249–252:116–50.
80. Konyukhov A, Lorenz C, Schweizerhof K. Various contact approaches for the finite cell method. Comput Mech. 2015;56:331–51.
81. Bog T, Zander N, Kollmannsberger S, Rank E. Weak imposition of frictionless contact constraints on automatically recovered high-order, embedded interfaces using the finite cell finite-element-method. Comput Mech. 2017;61:385–407.
82. Bog T. Frictionless contact simulation using the finite cell method. Ph.D thesis, Technical University of Munich. 2017.
83. Abedian A, Parvizian J, Düster A, Rank E. The finite cell method for the J_2 flow theory of plasticity. Finite Elem Anal Des. 2013;69:37–47.
84. Abedian A, Parvizian J, Düster A, Rank E. Finite cell method compared to *h*-version finite element method for elasto-plastic problems. Appl Math Mech. 2014;35(10):1239–48.
85. Kollmannsberger S, D'Angella D, Rank E, Garhuom W, Hubrich S, Düster A, Stolfo P. Di, Schröder A. Spline- and hp-basis functions of higher differentiability in the finite cell method. GAMM-Mitteilungen. 2019.
86. Duczek S, Joulaian M, Düster A, Gabbert U. Numerical analysis of Lamb waves using the finite and spectral cell methods. Int J Numer Methods Eng. 2014;99(1):26–53.

87. Elhaddad M, Zander N, Kollmannsberger S, Shadavakhsh A, Nübel V, Rank E. Finite cell method: high-order structural dynamics for complex geometries. Int J Struct Stab Dyn. 2015;15(07):1540018.
88. Duczek S, Gabbert U. The finite cell method: a higher order fictitious domain approach for wave propagation analysis in heterogeneous structures. In: Lammering R, Gabbert U, Sinapius M, Schuster T, Wierach P, editors. Lamb-wave based structural health monitoring in polymer composites. Cham: Springer International Publishing; 2018. p. 217–39.
89. Duczek S, Gabbert U. The finite cell method for polygonal meshes: poly-FCM. Comput Mech. 2016;58(4):587–618.
90. Lee NS, Bathe K-J. Effects of element distortions on the performance of isoparametric elements. Int J Numer Methods Eng. 1993;36(20):3553–76.
91. Duczek S, Berger H, Gabbert U. The finite pore method: a new approach to evaluate gas pores in cast parts by combining computed tomography and the finite cell method. Int J Cast Metals Res. 2015;28(4):221–8.
92. Joulaian M, Düster A. Local enrichment of the finite cell method for problems with material interfaces. Comput Mech. 2013;52(4):741–62.
93. Joulaian M, Zander N, Bog T, Kollmannsberger S, Rank E, Düster A. A high-order enrichment strategy for the finite cell method. PAMM. 2015;15(1):207–8.
94. Zander N, Bog T, Kollmannsberger S, Schillinger D, Rank E. Multi-level *hp*-adaptivity: high-order mesh adaptivity without the difficulties of constraining hanging nodes. Comput Mech. 2015;55(3):499–517.
95. Düster A, Bröker H, Rank E. The *p*-version of the finite element method for three-dimensional curved thin walled structures. Int J Numer Methods Eng. 2001;52(7):673–703.
96. Schillinger D, Ruess M, Zander N, Bazilevs Y, Düster A, Rank E. Small and large deformation analysis with the *p*− and B-spline versions of the finite cell method. Comput Mech. 2012;50(4):445–78.
97. de Berg M, Cheong O, van Kreveld M, Overmars M. Computational geometry. Berlin: Springer; 2008.
98. Suk T, Höschl C, Flusser J. Rectangular decomposition of binary images. In: Blanc-Talon J, Distante C, Philips W, Popescu D, Scheunders P, editors. Advanced concepts for intelligent vision systems. Berlin: Springer; 2012. p. 213–24.
99. Salomon D, Motta G. Handbook of data compression. London: Springer; 2010.
100. Spiliotis IM, Mertzios BG. Real-time computation of two-dimensional moments on binary images using image block representation. IEEE Trans Image Process. 1998;7(11):1609–15.
101. Eppstein D. Graph-theoretic solutions to computational geometry problems. In: Graph-theoretic concepts in computer science. Springer Berlin Heidelberg. 2010. p. 1–16.
102. Seo J, Chae S, Shim J, Kim D, Cheong C, Han TD. Fast contour-tracing algorithm based on a pixel-following method for image sensors. Sensors. 2016;16(3):353.
103. Pavlidis T. Algorithms for graphics and image processing. Berlin: Springer; 1982.
104. Miyatake T, Matsushima H, Ejiri M. Contour representation of binary images using run-type direction codes. Mach Vis Appl. 1997;9(4):193–200.
105. Ferrari L, Sankar PV, Sklansky J. Minimal rectangular partitions of digitized blobs. Comput Vis Graph Image Process. 1984;28(1):58–71.
106. Pető M. Improving the eciency of the numerical integration in the finite cell method. Master's thesis, Otto von Guericke University Magdeburg. 2019.
107. Wu SY, Sahni S. Covering rectilinear polygons by rectangles. IEEE Trans Comput-Aided Des Integr Circuits Syst. 1990;9:377–388, 05.
108. Wu SY, Sahni S. Fast algorithms to partition simple rectilinear polygons. VLSI Des. 1994;1(3):193–215.
109. Levcopoulos C, Gudmundsson J. Approximation algorithms for covering polygons with squares and similar problems. In: Randomization and approximation techniques in computer science. Springer Berlin Heidelberg. 1997. pp. 27–41.
110. Franzblau DS, Kleitman DJ. An algorithm for covering polygons with rectangles. Inf Control. 1984;63(3):164–89.
111. Gunther O. Minimum k-partitioning of rectilinear polygons. J Symb Comput. 1990;9(4):457–83.
112. Bondy JA, Murty USR. Graph theory. London: Springer; 2008.
113. Szabó B, Düster A, Rank E. The *p*-version of the finite element method. Chapter 5. Encyclopedia of computational mechanics. 2004.
114. Zander N, Bog T, Elhaddad M, Espinoza R, Hu H, Joly A, Wu C, Zerbe P, Düster A, Kollmannsberger S, Parvizian J, Ruess M, Schillinger D, Rank E. FCMLab: a finite cell research toolbox for MATLAB. Adv Eng Softw. 2014;74:49–63.
115. Gravenkamp H, Duczek S. Automatic image-based analyses using a coupled quadtree-SBFEM/SCM approach. Comput Mech. 2017;60(4):559–84.
116. Online source of the disc brake model. https://grabcad.com/library/brake-stuff-punch-disc-1. Accessed 16 Oct 2019.

Publisher's Note