Kudela *et al. Adv. Model. and Simul. in Eng. Sci.* (2015) 2:10
DOI 10.1186/s40323-015-0031-y

Advanced Modeling and Simulation in Engineering Sciences
a SpringerOpen Journal

RESEARCH ARTICLE Open Access

Efficient and accurate numerical quadrature for immersed boundary methods

László Kudela*, Nils Zander, Tino Bog, Stefan Kollmannsberger and Ernst Rank

*Correspondence:
laszlo.kudela@tum.de
Chair for Computation in Engineering, Technische Universität München, Arcisstrasse 21, 80333 Munich, Germany

Abstract

One question in the context of immersed boundary or fictitious domain methods is how to compute discontinuous integrands in cut elements accurately. A frequently used method is to apply a composed Gaussian quadrature based on a spacetree subdivision. Although this approach works robustly on any geometry, the resulting integration mesh yields a low order representation of the boundary. If high order shape functions are employed to approximate the solution, this lack of geometric approximation power prevents exponential convergence in the asymptotic range. In this paper we present an algorithmic subdivision approach that aims to be as robust as the spacetree decomposition even for close-to-degenerate cases—but remains geometrically accurate at the same time. Based on 2D numerical examples, we will show that optimal convergence rates can be obtained with a nearly optimal number of integration points.

Keywords: Immersed boundary methods, Finite Cell Method, Numerical quadrature

Background

One of the essential steps of performing a finite element analysis is the discretisation of the geometric domain into an analysis-suitable mesh. In case of complex geometries this process may become a severe bottleneck in the complete analysis pipeline. Studies show that the time required for creating an analysis-suitable geometry and its computational mesh accounts for about 80% of the overall analysis time [1].

In recent years, efforts to fill the gap between geometry and simulation brought forth many promising approaches, most of them related to the isogeometric analysis [1]. The idea of IGA is to use exactly the same shape functions for approximating the solution that are used for the geometry description. This way, the calculation can be performed on the geometric model directly, without any explicit mesh generation.

Other approaches—meshless methods, for example—aim to approximate the solution entirely in terms of nodal values [2]. Further approaches to avoid expensive mesh generation of complex geometries include the immersed boundary and fictitious domain methods [3, 4]. These techniques extend domains of complex shapes to a larger embedding domain, the geometry of which is simpler and can thus be meshed easily with a structured grid.

The Finite Cell Method (FCM) [5, 6] combines the basic idea of immersed boundary methods with higher order finite elements (p-FEM) [7, 8]. The physical domain is extended by the fictitious domain, such that their union results in a simple geometry

that can be meshed easily. The influence of the fictitious domain is marginalized by scaling down its material parameters by a small factor α. Due to this scaling factor the fictitious material is numerically extremely soft. Therefore, the strain energy of the solution in the original domain of interest and in the extended domain remains the same. For smooth problems, the method shows the same exponential convergence characteristics in the energy norm known from p-FEM [9], and allows accurate numerical computations without having to mesh complex geometries.

The FCM has been proven to work well in a number of contexts, such as shell analysis [10], large deformation analysis [11], voxel-based analysis on geometric models from CT-scans [12, 13] and for wave propagation problems [14]. The biggest advantage of FCM lies in high convergence rates with almost no meshing costs. An overview of the method together with a summary of recent developments can be found in [15].

One major difficulty of the FCM (as well as other fictitious domain methods) is posed by the introduction of a discontinuity in the cells that are intersected by the geometric boundary of the physical domain. Throughout this paper, these cells are referred to as cut cells. A direct Gaussian quadrature is inappropriate for discontinuous integrands, therefore the cut cells require special integration formulae (see, for example, [16]), or the application of composed integration schemes. The standard approach in the context of FCM is to use composed Gaussian quadrature in combination with a recursive spacetree-based refinement.

Although the spacetree-based approach is easy to implement and works robustly on any geometry, it has some disadvantages. Because the spacetree decomposition yields a low order approximation of the boundary, exponential convergence can only be achieved up to the point where the error of integration starts to dominate over the discretization error of the field variables. In order to reduce the integration error, more levels of spacetree subdivision have to be introduced. However, the number of integration points increases exponentially with every new level, thus making the analysis computationally expensive. Although this extra effort is bearable in linear computations, it may become prohibitively large if the amount of work per quadrature point is higher—as in nonlinear calculations, for example. These drawbacks indicate the need of another approach toward the quadrature point distribution to ensure a better geometric description of the boundary.

Finding the right quadrature rules for integrating through discontinuities is not an unknown question in the finite element community. In the context of the extended finite element methods (XFEM) [17], numerous approaches have been proposed to deal with this problem. One possibility is to fix the location of the quadrature points *a priori* and compute the relevant weights by solving the moment-fitting equations, as described in [18] or [19].

Another possibility in the context of XFEM is based on the idea of replacing the discontinuous integrand by an equivalent polynomial that is defined such that the evaluated stiffness matrix is exact. The original function and its polynomial substitute are equivalent in the integral sense, which is why the integral can be computed by standard quadrature rules without having to perform any subdivision. Examples of the method of equivalent polynomials can be found in [20] and [21].

Another technique to resolve the discontinuous integrand is to decompose the cut cells into Lagrangian elements in such a way that the edges of the resulting subcells align with the interface [22]. The use of triangular NURBS-Enhanced Finite Elements [23] in different XFEM settings has also been investigated in [24], based on an approach that is extended in this contribution.

The possibility of using the exact geometrical description of the boundary directly for the cut cell integration in the context of IGA and FCM is still an open question. Therefore, it shall be addressed in this paper—according to the following structure: "Theoretical background" briefly introduces the main concept of the Finite Cell Method and addresses the challenge of integration in the context of cut cells. "High order subcell integration" introduces a novel algorithmic approach aiming to overcome the integration problems in a robust, yet accurate way. In "Decomposition examples", some 2D example problems serve to demonstrate the proposed algorithm. The convergence properties of the approach in the context of FCM are discussed in "Finite Cell Method examples" along with an example of a wave propagation problem on a complex domain. The conclusions and an outlook on possibilities of further development are presented in "Conclusion and outlook".

Theoretical background

In the following, the essential ideas of the Finite Cell Method for steady linear elastic problems are discussed. For further details, see [5, 6, 11].

The essential ideas of the Finite Cell Method

As mentioned in the introduction, the FCM circumvents the task of mesh generation by extending the boundaries of the physical domain of interest Ω_{phy} by a fictitious part Ω_{fic}. Their union $\Omega_{phy} \cup \Omega_{fic}$ forms a simply shaped embedding domain $\Omega_{\cup}$ that can be meshed easily. The concept is depicted in Figure 1.

The derivation of FCM is based on the principle of virtual work [25]:

$$\delta W(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega} \boldsymbol{\sigma} : (\nabla_{sym}\delta\boldsymbol{u})dV - \int_{\Omega_{phy}} \delta\boldsymbol{u} \cdot \boldsymbol{b}dV - \int_{\Gamma_N} \delta\boldsymbol{u} \cdot \boldsymbol{t}dA = 0, \tag{1}$$

where $\boldsymbol{\sigma}$, $\boldsymbol{b}$, $\boldsymbol{u}$, $\delta\boldsymbol{u}$ and ∇_{sym} denote the Cauchy stress tensor, the body forces, the displacement vector, the test function and the symmetric part of the gradient, respectively.

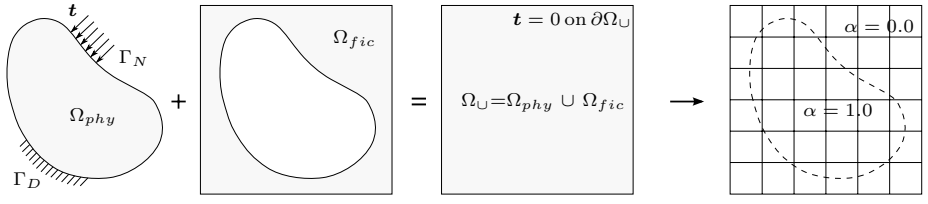


Figure 1 The core concept of the fictitious domain approach: the physical domain Ω_{phy} is extended by the fictitious domain Ω_{fic} such that their union forms the embedding domain $\Omega_{\cup}$. This allows the creation of a structured grid. To extinguish the influence of the fictitious domain, the material parameters of Ω_{fic} are penalized by α [11].

On Γ_N of the physical domain, the traction vector $\boldsymbol{t}$ specifies the Neumann boundary conditions.

Stresses and strains are related through the constitutive tensor $\boldsymbol{C}$:

$$\boldsymbol{\sigma} = \alpha \boldsymbol{C} : \boldsymbol{\varepsilon}, \tag{2}$$

where α is an indicator function defined as:

$$\alpha(\boldsymbol{x}) = \begin{cases} 1 & \forall \boldsymbol{x} \in \Omega_{phy} \\ 10^{-q} & \forall \boldsymbol{x} \in \Omega_{fict}. \end{cases} \tag{3}$$

Neumann boundary conditions of zero traction on the boundary of the physical domain are automatically satisfed. Inhomogeneous Neumann boundary conditions can be applied by simply integrating over the boundary Γ_N, regardless of whether the cell boundaries coincide with the geometric boundaries or not. Essential boundary conditions are generally imposed in the weak sense using variational techniques, such as the penalty method [26, 27], the Lagrange multiplier method [27], or Nitsche's method [28].

The unknown quantities $\delta \boldsymbol{u}$ and $\boldsymbol{u}$ are discretized by a linear combination of N_i shape functions with unknown coefficients $\boldsymbol{u}_i$:

$$\boldsymbol{u} = \sum_{i=1}^{n} N_i \boldsymbol{u}_i \tag{4}$$

$$\delta \boldsymbol{u} = \sum_{i=1}^{n} N_i \delta \boldsymbol{u}_i. \tag{5}$$

So far, different types of shapes functions have been used in the context of FCM, such as integrated Legendre polynomials [7], B-Splines [11] and NURBS [28].

Substituting (4) and (5) into (1) yields the discrete finite cell representation:

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}. \tag{6}$$

The stiffness matrix $\boldsymbol{K}$ results from a proper assembly of the element stiffness matrices $\boldsymbol{k}^e$ calculated as:

$$\boldsymbol{k}^{\boldsymbol{e}} = \int_{-1}^{1} \int_{-1}^{1} (\boldsymbol{LN})^T \boldsymbol{C} (\boldsymbol{LN}) \|\boldsymbol{J}\| d\xi d\eta, \tag{7}$$

where $\boldsymbol{L}$ is the standard strain-displacement operator, $\boldsymbol{N}$ is the matrix of shape functions, $\boldsymbol{C}$ is the constitutive matrix, and $\|\boldsymbol{J}\|$ is the Jacobian determinant of the mapping $\boldsymbol{Q}(\xi, \eta)$ that maps the local coordinates (ξ, η) of the cell to the global coordinate system (x, y) [25, 29, 30].

The challenge of integration

In conventional finite element approaches, the numerical evaluation of the integral in (7) is performed by the Gaussian quadrature [25, 31]:

$$\boldsymbol{k}^{\boldsymbol{e}} \approx \sum_{j}^{n_j} \sum_{i}^{n_i} (\boldsymbol{LN})^T \boldsymbol{C} (\boldsymbol{LN}) \|\boldsymbol{J}\|_{\xi_i,\eta_i} w_i w_j, \tag{8}$$

where ξ_i and η_i are the locations of the quadrature points in local coordinates of the cell, w_i and w_j are their corresponding weights.

Due to the discontinuity that is introduced by penalizing the constitutive matrix $\boldsymbol{C}$ (Eq. 3), the Gaussian quadrature loses its accuracy in cut cells [5, 6, 32]. In order to improve the precision of the numerical integration, the FCM uses a composed Gaussian quadrature that is based on a spacetree decomposition of the cells that are cut by the domain boundaries. In two dimensions, this means that every cut cell is recursively subdivided into 4 equal integration subcells until a predefined depth is reached (Figure 2). The quadrature points are distributed in the parameter space of each resulting integration cell and then mapped to the parameter space of the finite cell. Then, the Jacobian term in (8) is the product of the mappings $\boldsymbol{Q}(\xi, \eta)$ and $\tilde{\boldsymbol{Q}}(\tilde{\xi}, \tilde{\eta})$, where the terms with $\sim$ denote the local coordinates of the integration cell and the mapping from the parameter space of the integration cell to the parameter space of the finite cell. The concept of the mappings is depicted in Figure 3.

The advantage of the spacetree-based integration lies in its simplicity. As it merely relies on point membership classification, it is easy to implement and works robustly on any geometry. The drawback of this approach is that the number of integration points increases exponentially with every new level of spacetree decomposition, rendering the analysis computationally expensive. Moreover, the spacetree decomposition yields a low

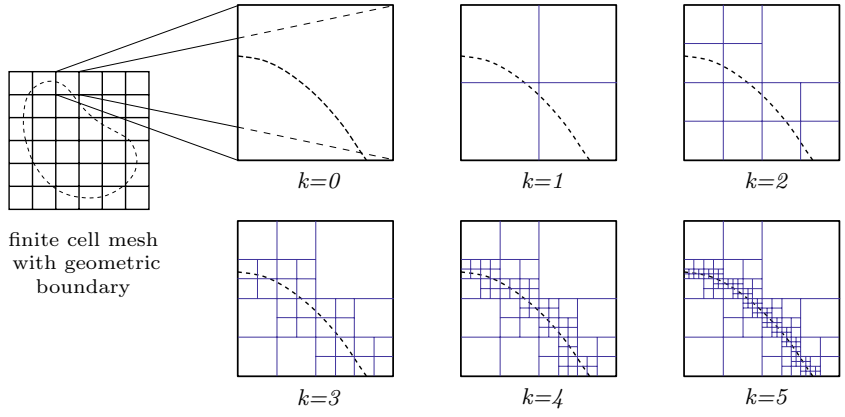


Figure 2 The method of quadtree partitioning. Each cut cell is recursively subdivided until a predefined subdivision depth is reached [11].

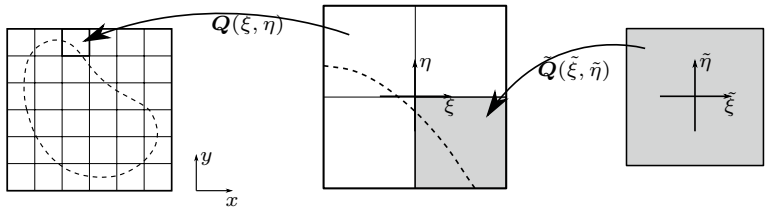


Figure 3 The concept of mappings for quadtree depth $k = 1$.

order approximation of the boundary and leaves the higher order geometric information (if available) unexploited.

Investigations in the field of XFEM show that it is beneficial to subdivide elements that are cut by the interface into a set of curved geometric entities. In [22], the cut elements are approximated by a set of Lagrangian elements with curved sides. The problem has also been addressed in [24], where an algorithmic approach to decompose cut elements into a set of curved triangles was presented. These triangles, based on NURBS-Enhanced Finite Elements recover the boundary in an exact sense.

The next section introduces an algorithm aiming to combine the robustness of the spacetree approach with the previously mentioned idea of exact geometry representation. The resulting set of triangles and quadrilaterals uses the blending function interpolation [33] in order to yield an exact representation of the boundary.

High order subcell integration

The simplicity of the spacetree decomposition results in robustness. From an algorithmic point of view, it is irrelevant how the boundary cuts through the cell: the resulting integration subcells are always created in the same way. In other words, the spacetree decomposition is completely insensitive to the topological configuration that the cutting boundary creates in a cut cell. In order to be robust, the enhanced decomposition method has to inherit this topological insensitivity. This means that it is beneficial to define a generic way of decomposition that can be applied to any kind of cut configuration, regardless of what kind of topology a cutting boundary creates inside one cell.

The standard method to identify in what way rectangular domains are cut is based on the method of marching squares, known from computer graphics [34]. Although this method works well to determine contours of scalar fields, it is a method designed for boundary recovery instead of domain decomposition. Moreover, instead of one generic procedure, it requires different treatment for different intersection patterns. To maintain robustness, the algorithm in this paper follows a different approach, that does not require the identification of such intersection cases.

Triangulated cut cell approach

Our algorithm follows the idea described in [22]. In this work, the authors state that a cut cell can always be decomposed into a set of quadrilaterals and triangles if the cell is cut in two triangles by a diagonal line.

The idea of the proposed algorithm is the following: if the diagonal line is drawn appropriately, the boundary always cuts the resulting two triangles into a triangle and a quadrilateral, regardless of how the boundary cuts through the cell. This idea is depicted in Figure 4. To identify how the diagonal line has to be created, the algorithm performs an inside-outside test on the four corner vertices of the cell. If two opposite vertices have opposite states, the diagonal line is created between them. After the line is created, the intersection points between the boundary and the diagonal line as well as the cell edges are computed. The boundary is then trimmed at these points of intersection. Details of implementation of the algorithm can be found in [35].

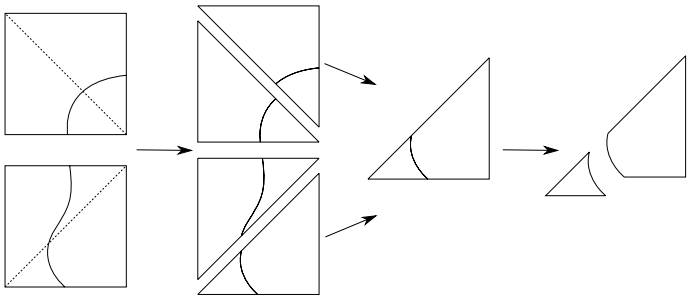


Figure 4 The general flow of the proposed algorithm. Every cut cell is subdivided into two triangles along a diagonal line. The triangles are further cut by the boundary into a triangle and a quadrilateral.

Mapping of curved regions

The resulting trimmed curves are used as the boundaries of the curved triangles and quadrilaterals. The proposed approach employs the blending function method introduced in [33]. This method uses the parametric description of the boundary to define a mapping between the standard bi-unit square and the curved triangle or quadrilateral representation. The definition of the blended mapping for a generic curved quadrilateral reads [7]:

$$\begin{aligned}\boldsymbol{Q}(\xi,\eta) = \frac{1}{2}\Big((1-\eta)\boldsymbol{E}_1(\xi) + (1+\xi)\boldsymbol{E}_2(\eta) + (1+\eta)\boldsymbol{E}_3(\xi) + (1-\xi)\boldsymbol{E}_4(\eta)\Big) \\ - \sum_{i=1}^{4} \boldsymbol{N}_i(\xi,\eta)\boldsymbol{X}_i,\end{aligned} \tag{9}$$

where $\boldsymbol{E}_i$, $\boldsymbol{N}_i$, $\boldsymbol{X}_i$ denote the parametric equations of the bounding curves, the standard bi-linear shape functions and the corners of the quadrilateral, respectively. Figure 5 shows the example of a quadrilateral with one high order boundary. The blended mapping can be extended to triangles with curved parametric boundaries by collapsing one of its edges into one point, as explained e.g. in [25].

Special cases

There are cases in which the previously outlined method is not able to provide an exact decomposition of the cut cell. The following points focus on these special cases and on the ways they can be treated.

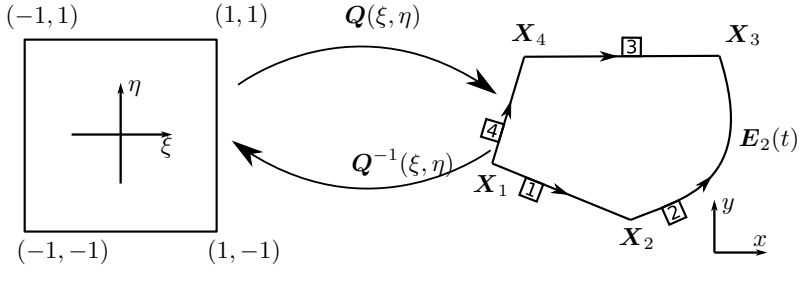


Figure 5 Blended quadrilateral with one curved side.

Degenerate cuts

If a non-convex boundary cuts the cell, there can be several intersection points between this boundary and the diagonal line (Figure 6). Therefore, no triangular decomposition can be made. Likewise, if the boundary has no intersection with the diagonal line, there is no straightforward decomposition that matches the procedure described before (Figure 7). This second kind of special configuration is detected by evaluating the inside-outside state of dedicated seed points on the domain of the cell. Thus, the special case in Figure 7 may be missed if the resolution of these seed points is not fine enough. If a special configuration is detected, the algorithm performs a quadtree subdivision and runs the triangulated decomposition on the leaf cells of the tree. This is done recursively, until the generic case of Figure 4 can be constructed. This quadtree based fallback option is depicted in Figures 6 and 7.

Kinks and corners

In many cases, the boundary of the domain is not composed of one continous curve, but is a set of connected curve segments. These points usually represent discontinous jumps in the curve derivatives and have to be taken into account by the decomposition algorithm in order to maintain the precision of the integration.

Therefore, if more than one curve is detected in a cell, the diagonal line is replaced by two linear segments. The point in which these segments are connected is the location of the kink in the cell. This case is depicted in Figure 8.

Piecewise definition of the boundary

As Eq. 9 suggests, the nature of the parametric description of the bounding curves has a strong influence on the mapping $\tilde{\boldsymbol{Q}}(\tilde{\xi}, \tilde{\eta})$ and thus on the Jacobian determinant of the

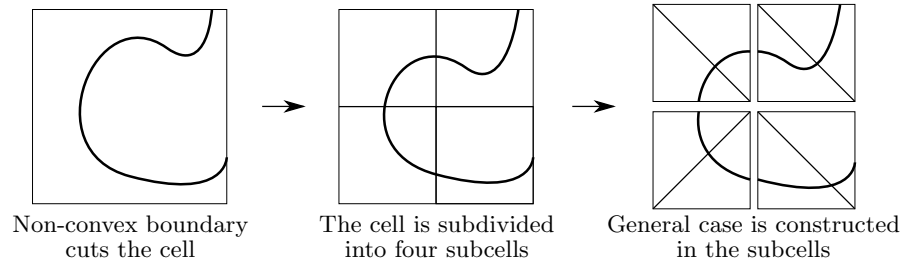


Figure 6 Non-convex boundary cuts the cell.

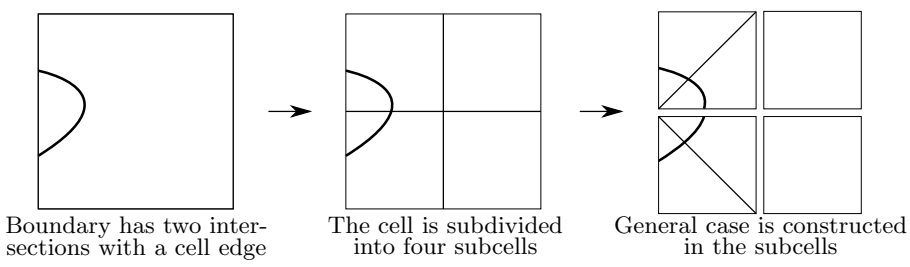


Figure 7 A boundary intersects a cell edge twice.

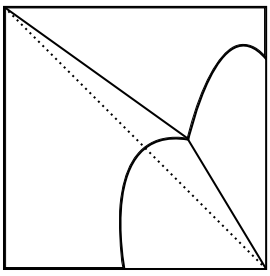


Figure 8 Resolution strategy for kinks: the diagonal line is replaced by two connected segments.

integration cell. If any of the bounding curves of the integration cell is defined piecewise, the Jacobian determinant in the integrand of the element stiffness matrix (Eq. 7) becomes non-smooth as well. Because the Gaussian quadrature is able to cope with polynomials but not with functions that are defined piecewise, this has to be taken into account by the decomposition algorithm. Having piecewise defined curves as boundary description is a very frequent case, as many geometries in engineering computations use B-Splines or NURBS. The parameter space of these curves is divided into a set of subintervals and the parametric equation of the curve changes at the breakpoint between neighboring subintervals. In the context of B-Splines or NURBS, these breakpoints are often referred to as knots.

In 1D, integrating piecewise polynomials numerically is performed by employing composed integration, based on the sum of the integrals on the separate intervals the curve is defined on. In order to perform exact numerical integration, this concept has to be applied to 2D: instead of integrating subintervals, the integration has to be carried out on subregions, where the boundaries of the regions are defined by the locations of the breakpoints. As an example, consider a blended integration cell with $e_b(\eta)$ being a piecewise-defined, curved boundary (Figure 9) that has one breakpoint at $\eta = -0.5$.

Because the definition of $e_b(\eta)$ changes at $\eta = -0.5$, the Jacobian determinant of the blended mapping changes too. Thus, the integration cell has to be further subdivided along the $\eta = -0.5$ isoparametric line. Integration then takes place on these subcells separately, and the complete integral is computed by the sum of the integrals on the subcells. In general, the triangulated subdivision algorithm is followed by an additional

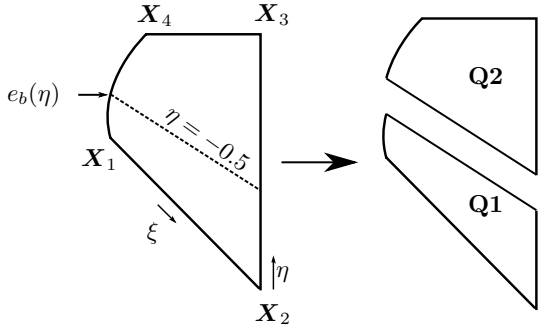


Figure 9 Example of breakpoint subdivision.

decomposition: all cells that are bounded by piecewise defined curves are further subdivided along the breakpoints of the curves.

As it will be pointed out later, this breakpoint-wise subdivision is a necessary step in order to be able to compute highly accurate integrals of subcells bounded by piecewise curves.

Decomposition examples

This section demonstrates the proposed decomposition algorithm on a few examples.

Moving circle in rectangular domain

We define a square-shaped domain with a circular hole inside, the center of which moves on a circular path (Figure 10). The boundary of the circular hole is composed of four arcs. In every time step, a different geometrical setting has to be partitioned. This way, it can be assessed, how the algorithm copes with non-regular configurations. As Figure 11 depicts, every cut cell is decomposed into two pairs of quadrilaterals and triangles. In cells where neighboring arcs join, the diagonal cutting line is drawn according to the meeting point of the curves. It is worth observing how the degenerate case is resolved by the quadtree decomposition in Figure 11c.

Integration test

The precision of the entire subdivision algorithm is assessed on a geometric setup that contains all the special cases of "Special cases".

We introduce an "integration patch test" with the following idea: the value of the scaling factor is chosen to be $\alpha = 1$ both on Ω_{phy} (with a possibly complex geometry) and Ω_{fict}. Because the scaling factor is the same in both domains, there is no discontinuity in the cells anymore and the numerical problem simplifies to a 2D finite element

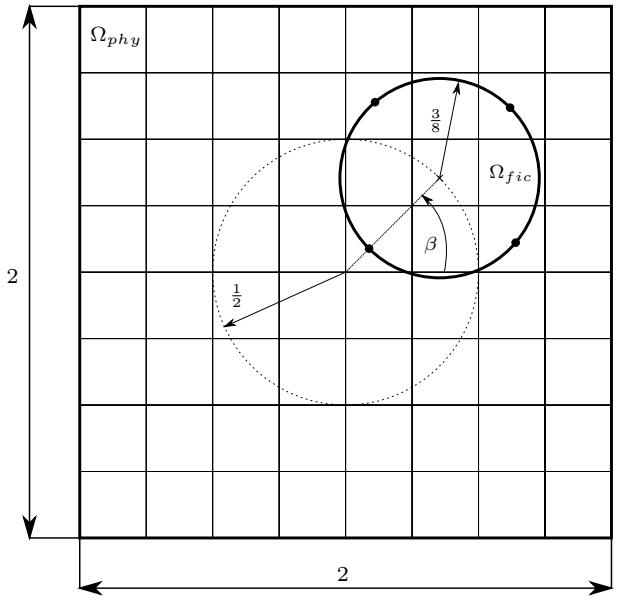


Figure 10 Setting of the moving circle example.

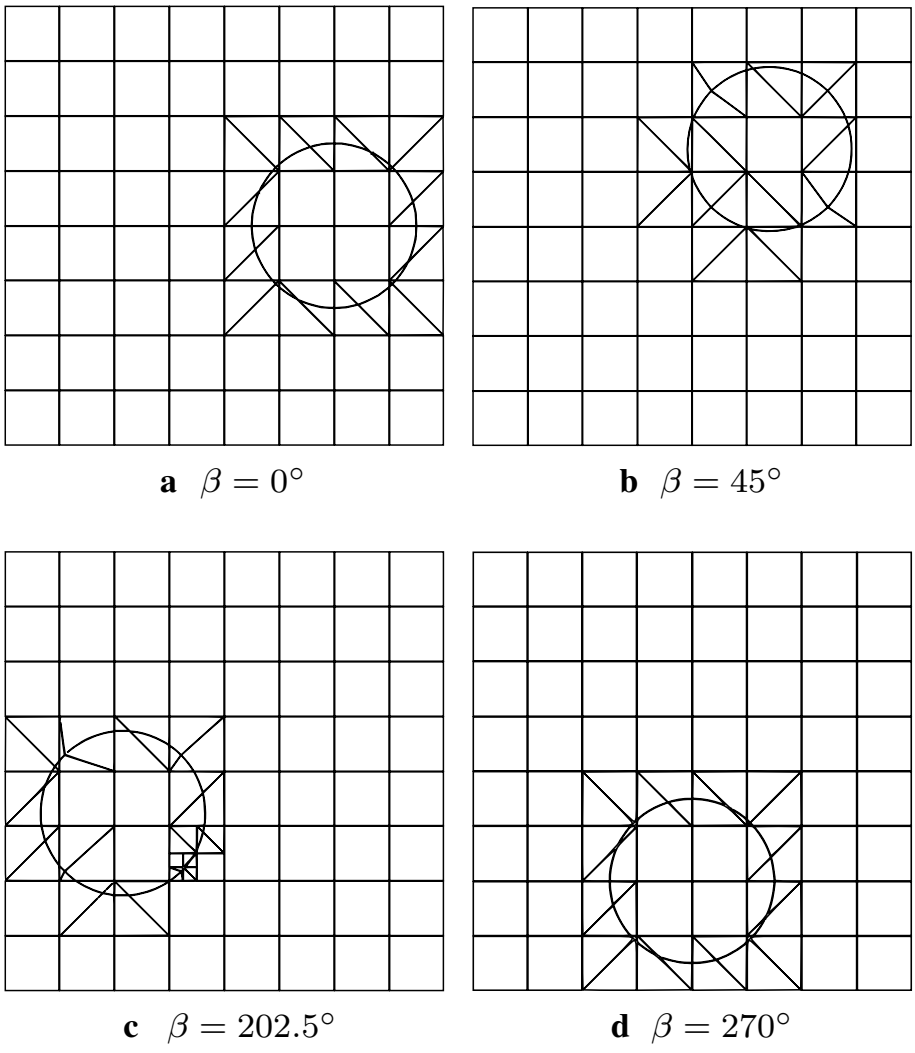


Figure 11 Decomposition example: moving circle.

computation on a rectangular domain with a quadrilateral mesh. If this domain is subjected to constant strains, the linear shape functions spanned on the quadrilateral mesh have to be able to represent the solution exactly, because the completeness condition is satisfied [25, 29]. Note that the integrands of the element stiffness matrices (Eq. 7) are still computed on the subcells resulting from the decomposition algorithm. Therefore, any possible difference between the numerical and the analytical solution is a sign that there is an error in the integration. To quantify these differences in a global sense, the numerical and analytical strain energy are compared using the following error measure:

$$e = \sqrt{\frac{|u_{ex} - u_{num}|}{u_{ex}}}, \tag{10}$$

where u_{ex} and u_{num} are the exact and numerical values of the strain energy, respectively.

As an example, consider the geometric setting depicted in Figure 12. The boundaries of Ω_{phy} represent the cover plate of a violin, including the f-holes (Figure 13). The boundary of the violin—including the f-holes—are represented by B-Spline curves of 3rd order. On this domain, the Laplace equation

$$\Delta \Phi = 0 \tag{11}$$

is solved with Dirichlet boundary conditions applied on the top and bottom of the mesh grid, such that the field value $\Phi = 0$ on Γ_{D_1} and $\Phi = h$ on Γ_{D_2}, where h is the height of Ω_{fic}. As a result, the gradient of the field value $\frac{\partial \Phi}{\partial x}$ is equal to one throughout the whole domain. The boundaries denoted by Γ_N are defined as free Neumann boundaries. Because the solution is linear, the polynomial order of the shape functions is chosen to be $p = 1$.

Figure 13a illustrates the results of the decomposition without applying breakpoint-wise subdivision. The resulting integration mesh with breakpoint subdivision is depicted in Figure 13b. Figure 14 shows how the algorithm copes with the small geometric features of the domain.

The analytical value of the strain energy is

$$u_{ex} = \frac{1}{2} \int_{\Omega} \left(\frac{\partial \Phi}{\partial x} \right)^2 d\Omega = \frac{1}{2} wh, \tag{12}$$

where w denotes the width of the domain (Figure 12). Figure 15 shows the error in the strain energy depending on the number of quadrature points distributed per parametric

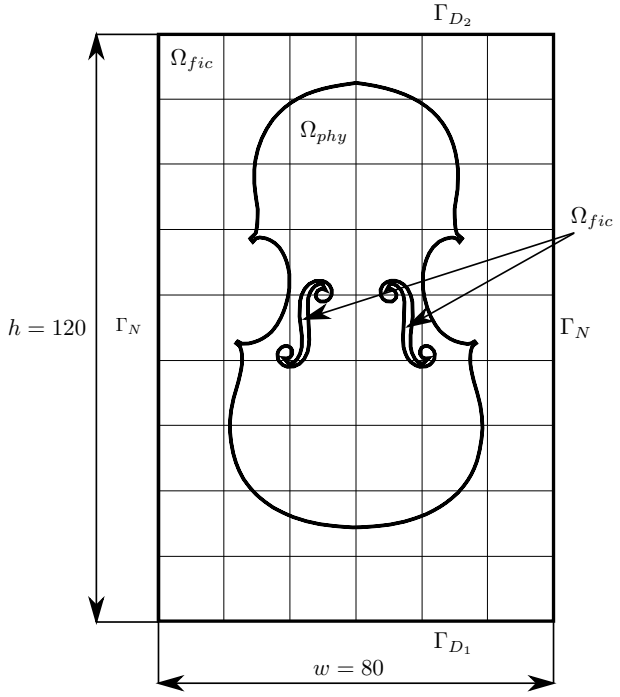
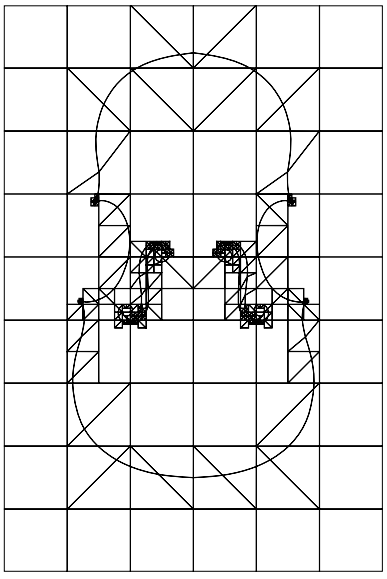
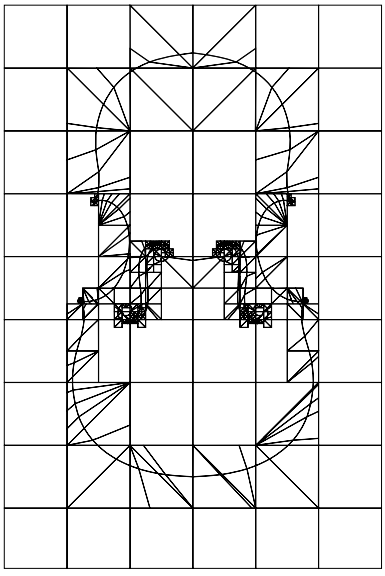


Figure 12 Geometry setup for the violin partitioning example.



a Without breakpoint subdivision



b With breakpoint subdivision

Figure 13 Decomposition examples on the violin domain.

direction in each cell. If the breakpoints are not taken into account as explained in "Piecewise definition of the boundary", the error remains higher regardless of how many integration points are distributed in the cells. However, if the integration cells are subdivided along the breakpoints of the bounding curves, the error in the energy norm converges to machine precision.

Here, it is worth noting that the total number of integration points is influenced by the parametric definition of the curves to a great extent. This means that if there are many breakpoints present in the boundary curves, there will also be a lot of breakpoint-wise

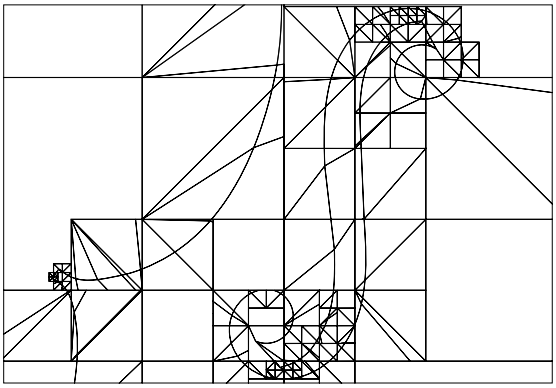


Figure 14 A detailed view of the f-hole.

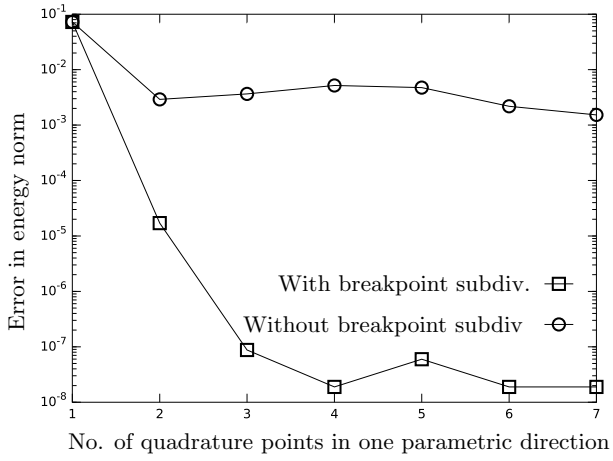


Figure 15 Error in energy norm with and without breakpoint subdivision.

subdivision performed. Therefore a high number of quadrature points is generated which results in high overall runtime.

In order to determine whether a cell is cut, both the blended and the quadtree based approaches require the evaluation of the inside-outside state of dedicated seed points. However, due to its recursive nature, the quadtree based technique has to evaluate the point membership in every cell of every level of subdivision. Thus, for higher subdivision levels the quadtree based integration mesh generation becomes significantly more expensive than the proposed algorithm. This is also confirmed by the comparison in Figure 16.

Finite Cell Method examples

This section demonstrates the capabilities of the proposed integration method in the context of FCM. First, the accuracy and the performance is compared to the quadtree based approach by solving a linear elastic problem. This is followed by the simulation of a compression wave propagating through a complex domain.

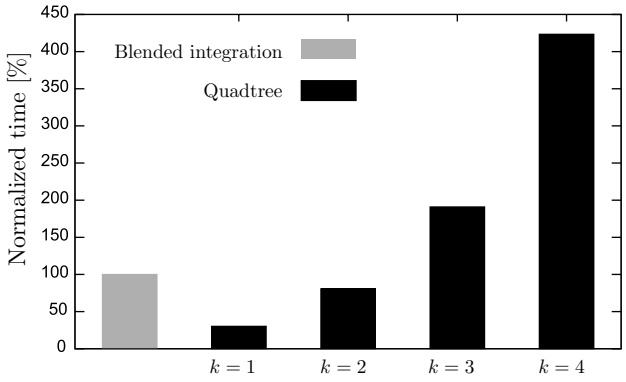


Figure 16 Comparison of the time required for generating different integration meshes for the violin example.

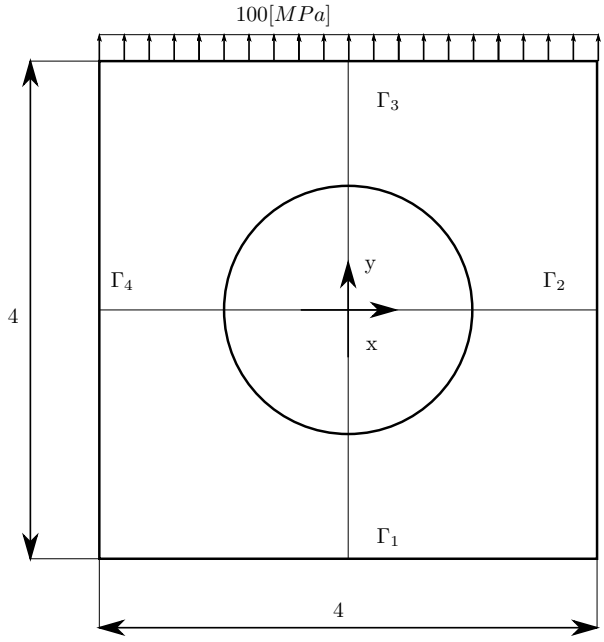


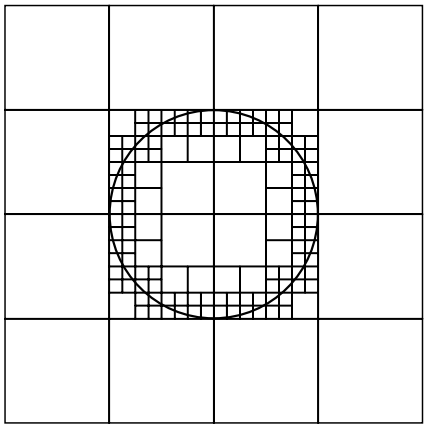
Figure 17 Perforated plate problem.

Perforated plate

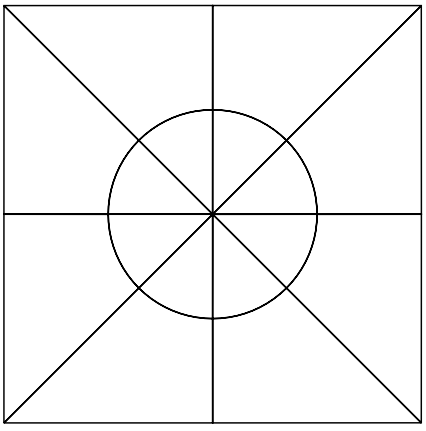
The first example is a plane stress problem that was already analysed in the context of FCM [6], with the geometric setting depicted in Figure 17. The material of the perforated plate is steel, with the properties $E = 2.069 \cdot 10^5$[MPa], $\nu = 0.29[-]$. The plate is vertically loaded by 100 [MPa]. Symmetry conditions are applied on Γ_1 and Γ_4. The boundaries of the hole and Γ_2 are treated as free boundaries. The domain is discretized into 2×2 finite cells. The polynomial degree of the shape functions is increased from $p = 1$ to $p = 15$. The reference strain energy of the problem is $U = 0.7021812127$, obtained by an "overkill" p-FEM solution from [6].

The integration is performed on blended integration cells and on quadtree cells with a depth of $k = 4$ and $k = 5$. For comparison, the same problem is solved by means of linear finite elements (h-FEM) with different element sizes. For the quadtree integration cells and the linear finite elements the number of quadrature points is chosen to be $(p + 1)^2$. To account for the high order boundaries of the blended integration mesh, $(p + 4)^2$ integration points are distributed in the curved integration cells. Figure 18 depicts the integration cell meshes and a mesh of linear finite elements. The error in the strain energy (Eq. 12) is plotted in Figure 19a.

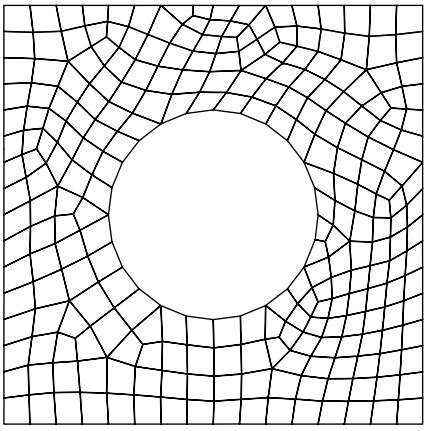
Both the quadtree and the blended integration show exponential convergence, similar to p-FEM. However, the curve representing the quadtree integration levels off at an error of approximately $10^{-2}[-]$. At this point the integration error dominates over the discretization error which renders a further increase of the polynomial degree pointless. The integration error can be reduced by adding more levels of refinement to the spacetree subdivision—however, the low approximation of the integration does not allow for



a Quadtree integration mesh, $k = 4$

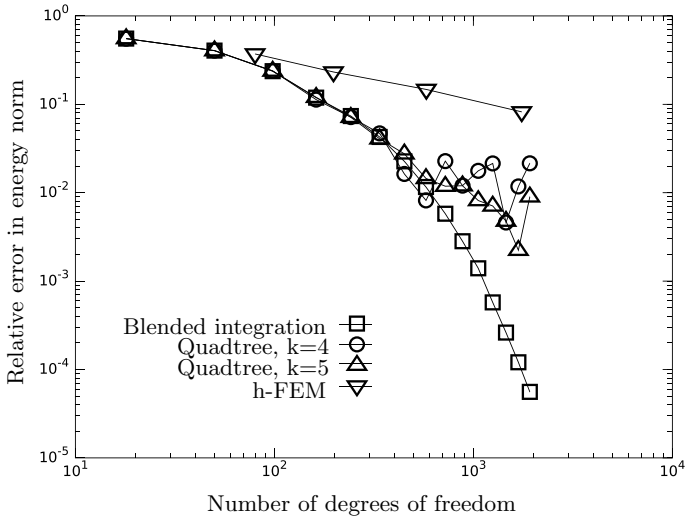


b Blended integration mesh

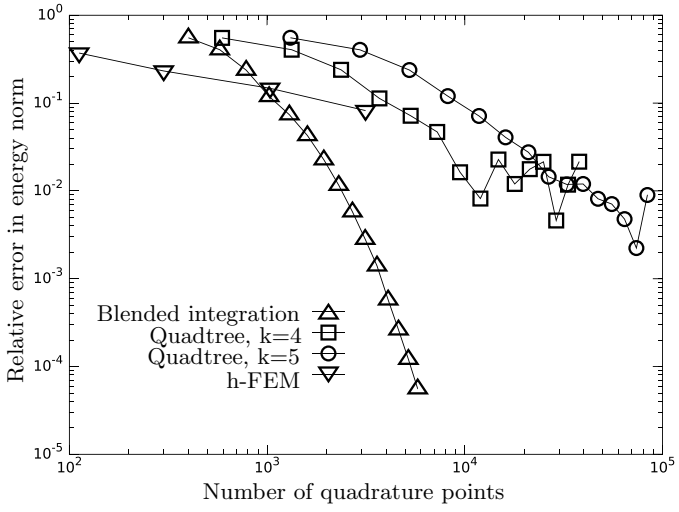


c Mesh of 246 linear quadrilaterals

Figure 18 Integration meshes and a finite element mesh for the perforated plate example.



a Convergence w.r.t. no. of degrees of freedom



b Convergence w.r.t. no. of quadrature points for $p = 1..15$

Figure 19 Comparison of the convergence characteristics of different methods.

exponential convergence in the asymptotic range. In comparison, the blended integration that uses the parametric description of the boundaries shows exponential convergence also in the asymptotic sense.

The number of quadrature points has a major influence on the overall computational cost of a numerical simulation. Therefore, the relationship between the number of integration points and the relative error in strain energy is an important aspect when it comes to assessing the performance of the different approaches. This relationship is depicted in Figure 6. Apart from the better convergence characteristics, the other advantage of the blended subcell integration lies in the total number of Gauss points distributed on the domain. For the same error in the strain energy, the blended integration cells require approximately one order of magnitude less Gauss points than a quadtree

integration mesh with a depth of $k = 4$. If the depth of the quadtree is increased to $k = 5$, the point at which the convergence curve levels off is slightly shifted to a lower value. However, this gain in precision comes with the cost of high computational overhead. As an example, consider a relative error of 10^{-4} in the energy norm, where the $k = 5$ quadtree integration mesh needs approximately 40,000 quadrature points. In contrast, the blended mesh needs approximately 2,500 integration points to reach the same error.

A comparison in terms of the time required to compute the stiffness matrix—including the generation of the integration mesh—is plotted in Figure 20. The quadtree based computation becomes more expensive than the blended one for two reasons. The first reason is due to the extra inside-outside tests required by the quadtree with high levels of k (refer to "Integration test"). Secondly, the high number of quadrature points leads to an excessive number of matrix-matrix product evaluations in Eq. 8.

Both the blended and quadtree methods show better convergence characteristics in comparison to the standard h-FEM on the basis of the number of degrees of freedom (Figure 19a). Comparing the number of integration points of the different approaches reveals that up to approximately 1,000 integration points the error in the strain energy of the h-FEM solution is smaller than the error of the blended integration. This point is located where the curve of the h-FEM error intersects the curve of the blended integration error on Figure 19b. For the quadtree methods, this intersection with the h-FEM error curve lies in regions of higher number of Gauss points.

Here, it should be noted that although the h-FEM is at least as precise as the blended subcell integration up to this intersection point, it requires considerably more degrees of freedom than the FCM approach.

Compression wave on the violin-shaped domain

Consider the geometric setting with the violin-shaped domain of "Integration test". We assume linear homogenous isotropic material properties on the physical domain Ω_{phys},

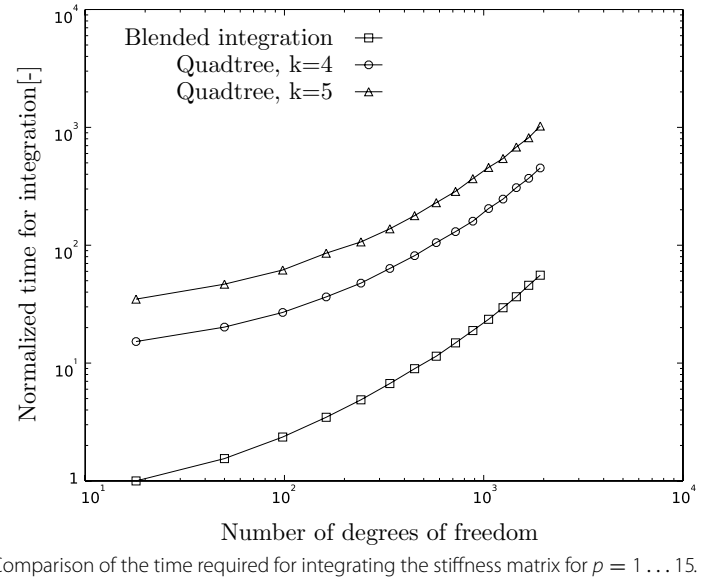


Figure 20 Comparison of the time required for integrating the stiffness matrix for $p = 1 \ldots 15$.

and simulate a pressure wave advancing through the domain by solving the wave equation [25, 29]:

$$\frac{\partial \Phi}{\partial t^2} - \alpha^2 \nabla^2 \Phi = 0. \tag{13}$$

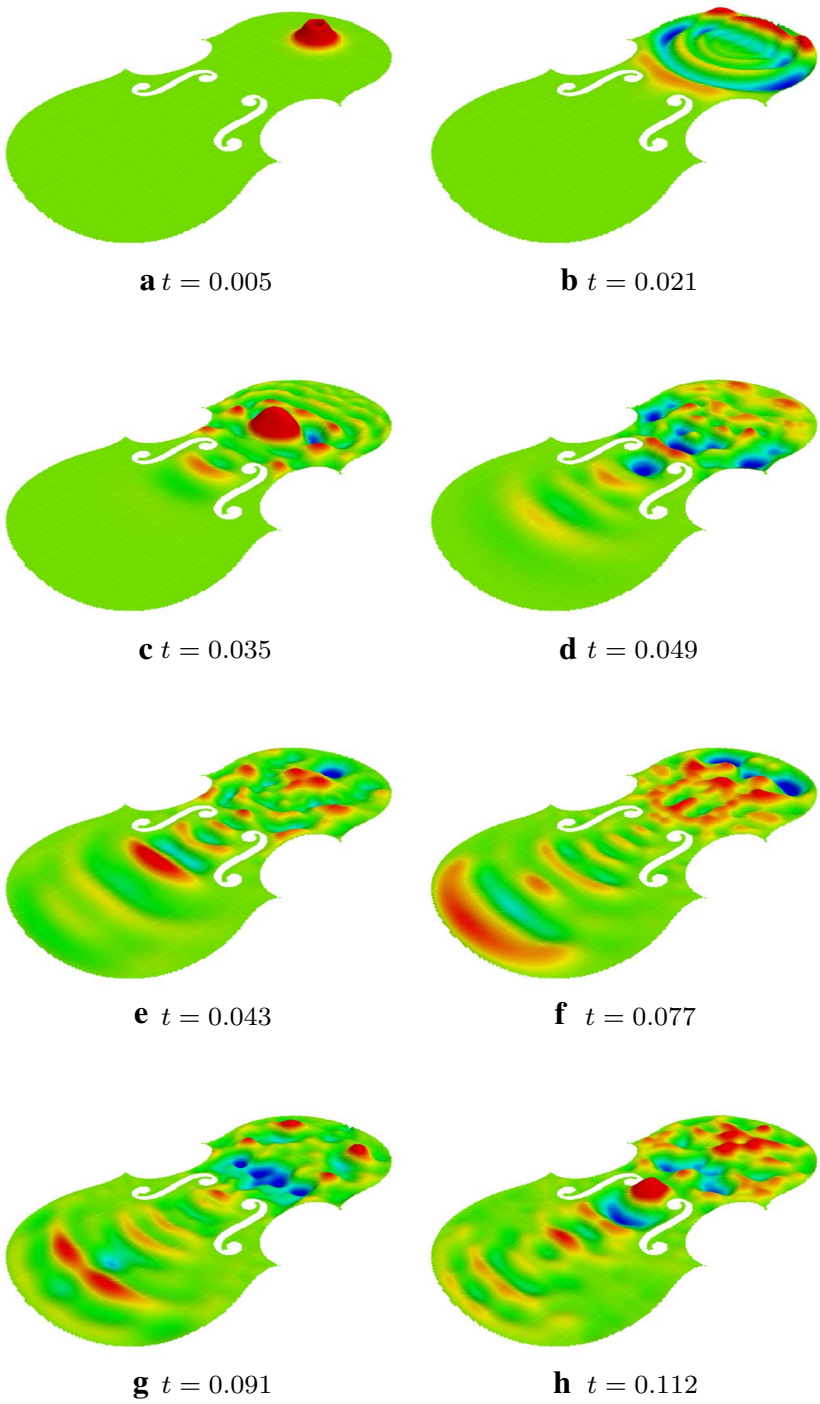


Figure 21 Values of Φ plotted for selected t values.

Homogenous Dirichlet boundary conditions are specified both on the outer boundary and the inner holes of the physical domain. The system is excited by a Gaussian wavelet in the upper region of Ω_{phys}. The finite cell mesh is composed of 8×14 elements with a polynomial order $p = 6$. The time interval of the simulation $[t_{min}, t_{max}] = [0, 0.120]$ is discretized into 120 time steps and the Newmark-beta method is used for time integration [25, 29]. It is worth noting here that an integration mesh with blended elements including breakpoint subdivision and 7×7 quadrature points per integration cell yields approximately 3.5 times less integration points in comparison to the quadtree based integration with a refinement level of $k = 4$. Figure 21 shows the results of the simulation over time.

Conclusion and outlook

This paper presented a novel approach to overcome the integration challenges in the context of the Finite Cell Method. The standard integration in FCM is performed by a spacetree based composed Gaussian quadrature, which works robustly on any geometry, but lacks the geometric approximation power that balances well with the high order shape functions of p-FEM.

The presented algorithm decomposes the cut finite cells into a set of curved quadrilaterals and triangles. Knowing the parametric description of the boundary, these triangles and quadrilaterals use blending functions to map the quadrature points.

The algorithm aims to be generic in such a way that different topological cutting situations can be handled in a similar manner. Thus, similarly to the quadtree decomposition, the method is able to cope with complex geometries in a robust way. The error in integration with blended integration cells was investigated on a complex geometry by setting up a "pseudo-FCM" problem that has an exact solution but is numerically computed on the blended integration mesh. It was found that in case of curves that follow a piecewise parametric definition (e.g. B-Splines), the integration can be highly accurate only if the blended cells are subdivided along the breakpoint locations of the bounding curves.

The blended subcell integration shows better convergence characteristics in comparison to the classical spacetree decomposition with at least one order of magnitude less quadrature points for the same error. To demonstrate that the algorithm is capable of working with complex geometries, a cover plate model of a violin was partitioned and the propagation of a compression wave was simulated on its domain. Thanks to its robustness and high accuracy, the algorithm is a good fit into the design-through-analysis pipeline.

Further research should address an extension of the algorithm to 3D cases. As the blending function concept extends to 3D naturally, the prior question would be how to find a generic decomposition-approach that is able to handle different cutting situations in a robust manner.

Authors' contributions
All authors have prepared the manuscript. All authors read and approved the final manuscript.

Compliance with ethical guidelines

Competing interests
The authors declare that they have no competing interests.

Received: 11 February 2015 Accepted: 23 May 2015
Published online: 18 June 2015

References

1. Cottrell JA, Hughes TJR, Bazilevs Y (2009) Isogeometric analysis: towards integration of CAD and FEM. Wiley, New York
2. Nguyen N-Q, Ladd AJC (2008) Meshless methods: a review and computer implementation aspects 79:763–813. doi:10.1016/j.matcom.2008.01.003
3. Löhner R, Cebral JR, Camelli FF, Baum JD, Mestreau EL, Soto OA (2007) Adaptive embedded/immersed unstructured grid techniques. Arch Comput Methods Eng 14:279–301
4. Peskin C (2002) The immersed boundary method. Acta Numerica 11:1–39
5. Düster A, Parvizian J, Yang Z, Rank E (2008) The finite cell method for three-dimensional problems of solid mechanics. Comput Methods Appl Mech Eng 197:3768–3782
6. Parvizian J, Düster A, Rank E (2007) Finite cell method—h- and p-extension for embedded domain problems in solid mechanics. Comput Mech 41:121–133
7. Szabó BA, Babuška I (1991) Finite element a nalysis. Wiley, New York
8. Szabó BA., Düster A., Rank E (2004) The p-version of the Finite Element Method. In: Stein E, de Borst R, Hughes TJR (eds) Encyclopedia of computational mechanics, vol 1, Chap 5. Wiley, Chichester, pp 119–139
9. Dauge M, Düster A, Rank E (2013) Theoretical and numerical investigation of the finite cell method. Technical Report hal-00850602, CCSD. http://hal.archives-ouvertes.fr/hal-00850602
10. Rank E, Ruess M, Kollmannsberger S, Schillinger D, Düster A (2012) Geometric modeling, isogeometric analysis and the Finite Cell Method. Comput Methods Appl Mech Eng 249–252:104–115. doi:10.1016/j.cma.2012.05.022
11. Schillinger D, Ruess M, Zander N, Bazilevs Y, Düster A, Rank E (2012) Small and large deformation analysis with the p- and B-spline versions of the finite cell method. Comput Mech 50:445–478. doi:10.1007/s00466-012-0684-z
12. Ruess M, Tal D, Trabelsi N, Yosibash Z, Rank E (2012) The finite cell method for bone simulations: verification and validation. Biomech Model Mechanobiol 11:425–437
13. Schillinger D, Düster A, Rank E (2012) The *hp-d*-adaptive finite cell method for geometrically nonlinear problems of solid mechanics. Int J Numer Methods Eng 89:1171–1202. doi:10.1002/nme.3289
14. Joulaian M, Duczek S, Gabbert U, Düster A (2014) Finite and spectral cell method for wave propagation in heterogeneous materials. Comput Mech 54(3):661–675
15. Schillinger D, Ruess M (2014) 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 1–65. doi:10.1007/s11831-014-9115-y
16. Müller B, Kummer F, Oberlack M (2013) Highly accurate surface and volume integration on implicit domains by means of moment-fitting. Int J Numer Methods Eng 96:512–528. doi:10.1002/nme.4569
17. Belytschko T, Gracie R, Ventura G (2009) A review of extended/generalized finite element methods for material modeling. Modell Simulat Mater Sci Eng 17:043001
18. Mousavi SE, Sukumar N (2011) Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons. Comput Mech 47:535–554
19. Sudhakar Y, Wall WA (2013) Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods. Comput Methods Appl Mech Eng 158:39–54
20. Ventura G (2006) On the elimination of quadrature subcells for discontinuous functions in the extended finite-element method. Int J Numer Methods Eng 66:761–795
21. Ventura G, Benvenuti E (2014) Equivalent polynomials for quadrature in heaviside function enriched elements. Int J Numer Methods Eng. doi:10.1002/nme.4679. Accessed 2014-08-03
22. Cheng KW, Fries T-P (2009) Higher-order XFEM for curved strong and weak discontinuities. Int J Numer Methods Eng 82:564–590
23. Sevilla R, Fernández-Méndez S, Huerta A (2008) NURBS-enhanced finite element method (NEFEM). Int J Numer Methods Eng 76(1):56–83. doi:10.1002/nme.2311. Accessed 2014-02-27
24. Legrain G (2013) A nurbs enhanced extended finite element approach for unfitted cad analysis. Comput Mech 52(4):913–929
25. Hughes TJR (2000) The Finite Element Method: linear static and dynamic finite element analysis. Dover Publications, Mineola, New York
26. Babuška I (1973) The finite element method with penalty. Math Comput 27:221–228
27. Fernández-Méndez S, Huerta A (2004) Imposing essential boundary conditions in mesh-free methods. Comput Methods Appl Mech Eng 193(12–14):1257–1275
28. Ruess M, Schillinger D, Bazilevs Y, Varduhn V, Rank E (2013) 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 95(10):811–846. doi:10.1002/nme.4522
29. Bathe KJ (2002) Finite-Elemente-Methoden. Springer, Berlin
30. Zienkiewicz OC, Taylor RL (2005) The Finite Element Method—its basis and fundamentals, vol 1, 6th edn. Butterworth-Heinemann, Oxford
31. Davis PJ, Rabinowitz P (1975) Methods of Numerical Integration by Philip J. Davis and Philip Rabinowitz. Academic Press, New York
32. Abedian A, Parvizian J, Düster A, Khademyzadeh H, Rank E (2013) Performance of different integration schemes in facing discontinuities in the finite cell method. Int J Comput Methods 10(3):1350002/1–24. doi:10.1142/S0219876213500023
33. Gordon WJ, Hall CA (1973) Transfinite element methods: blending function interpolation over arbitrary curved element domains. Numerische Mathematik 21:109–129

34. Lorensen WE, Cline HE (1987) Marching cubes: a high resolution 3d surface construction algorithm. Comput Graph 21(4):163–169
35. Kudela L (2013) Highly accurate subcell integration in the context of the finite cell method. Master's thesis, Technische Universität München