Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Efficient finite element quadrature on immersed voxel data

Matti Christmann

matti.christmann@uni-weimar.de

Bauhaus-Universität Weimar

Philipp Kopp

Bauhaus-Universität Weimar

Stefan Kollmannsberger

Bauhaus-Universität Weimar

Research Article

Keywords: moment fitting, numerical quadrature, finite cell method, voxel-based finite elements, immersed boundary methods, adaptive refinement

Posted Date: July 1st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10093795/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Engineering with Computers on October 12th, 2026. See the published version at https://doi.org/10.1007/s00366-026-02436-9.

Efficient finite element quadrature on immersed voxel data

Matti Christmann[1*], Philipp Kopp[1] and Stefan Kollmannsberger[1]

[1]Bauhaus-Universität Weimar, Chair of Data Science in Civil Engineering, Weimar, Germany.

*Corresponding author(s). E-mail(s): matti.christmann@uni-weimar.de;

Abstract

High-resolution computed tomography (CT) scans provide voxel-wise information on geometry and material properties. Direct numerical simulations on CT data in which each voxel is considered a finite element can yield billions of unknowns, a resolution that is often excessive. Coarsening the finite element mesh reduces the number of unknowns, but accurate quadrature then becomes a challenge. Composed integration by distributing Gauss-Legendre points on each voxel is computationally expensive due to the resulting large number of integration points. The standard remedy is to pre-integrate the stiffness contribution of each voxel in reference coordinates. However, this approach does not directly extend to local enrichment of the finite element space. For example, under hierarchical octree-based refinement the basis functions vary from element to element. For such cases, we develop an efficient voxel quadrature scheme based on a modified moment-fitting approach. The method computes a custom integration rule for each element which absorbs voxel information directly into the integration weights. This enables exact integration of the stiffness matrix of finite elements with any polynomial degree while limiting the number of quadrature points during assembly. In contrast to pre-integration, our approach decouples the material resolution from the finite element basis and is even faster for coarse finite element resolutions, while remaining far cheaper than composed Gauss-Legendre quadrature. We demonstrate voxel-based moment fitting on three examples: a scaling analysis for heterogeneous voxel data, a voxelized L-shaped domain with adaptivity driven by a residual-based error estimator, and a locally refined phase-field fracture simulation.

Keywords: moment fitting, numerical quadrature, finite cell method, voxel-based finite elements, immersed boundary methods, adaptive refinement

Highlights

- Voxel moment fitting absorbs per-voxel material into custom quadrature weights.
- Exact stiffness integration with far fewer points than composed Gauss quadrature.
- Basis-independent: enables multi-level *hp*-adaptivity where pre-integration cannot.

1 Introduction

Computed tomography (CT) is one of the main diagnostic tools able to capture internal structures of parts in great detail. CT-scans are not only used for visualization, but are also the starting point for a mechanical analysis of scanned parts. The most popular numerical techniques in this field are based on the Fast Fourier Transform (FFT) [1, 2] or finite elements. FFT-based methods are most favorably employed in situations where periodic boundary conditions arise naturally. This is the case, e.g., in homogenization tasks, but uncommon for general-purpose computations in solid mechanics of entire structures. In such cases, the finite element method is commonly employed. For historic

reasons, applications are mainly found in the field of mechanobiology [3, 4]. However, since CT-scan-based mechanical analysis is generally useful for the investigation of inhomogeneous structures, it is also used in the material science community (see, e.g., [5]).

The finite element method requires an analysis-suitable mesh. The generation of such meshes may be accomplished by segmentation and boundary reconstruction; see, e.g., [3, 6] for domain-specific approaches in mechanobiology. Although the technology is widely applicable [7], a fully automated procedure in general settings remains a challenge and is prone to loss of detail. Alternatively, voxel-based finite element approaches exist. These directly utilize the CT-scan as a computational grid by converting every voxel into a finite element. This approach is, therefore, readily automatable and avoids loss of detail, but naturally leads to a very high number of degrees of freedom [8]. A remedy that keeps the number of degrees of freedom at bay is provided by the voxel-based finite cell method (FCM) [9]. It allows a single finite element to contain multiple voxels of the CT-scan [4, 10–12]. At the same time, it respects the distinct material properties of each voxel through composed integration of the stiffness matrix and uses high-order shape functions that can potentially also be locally refined. It thereby incorporates all detail provided by the CT scan and remains automatable. However, the cost of computing the stiffness matrices is increased compared to a merging of voxels, since composed integration must be used to remain accurate. The element stiffness matrix $\mathbf{K}_e$ involves the evaluation on the element domain Ω_e of an integral usually denoted as

$$\mathbf{K}_e = \int_{\Omega_e} \mathbf{B}^T \, \mathbf{C} \, \mathbf{B} |\mathbf{J}_e| \, d\Omega_e \,, \tag{1}$$

where $\mathbf{B}$ is the strain-displacement matrix that relates displacements to strains, and $\mathbf{C}$ is the material matrix. For a Cartesian grid with a constant mapping from the local to the global coordinate system, the determinant of the Jacobian matrix $|\mathbf{J}_e|$ is constant within Ω_e and can be extracted from the integral. Commonly, the integral in Equation (1) is evaluated numerically using Gauss-Legendre quadrature, which assumes the integrand to be an infinitely smooth polynomial. A discontinuous integrand leads to integration errors that are commonly larger than the approximation error of the finite element method itself. This is the case when $\mathbf{C}$ is defined by a CT-scan that provides heterogeneously distributed material constants and the voxels of the CT-scan do not coincide with the boundaries of the finite elements. Such situations naturally arise when several voxels with different values are subsumed into a finite element to gain efficiency without spending extra effort.

A simple remedy is to subdivide the integral at the discontinuities, i.e., the boundaries of the voxels, into subdomains defined by the voxels Ω_v within which the integrand is smooth. The integral over Ω_e can then be treated as a sum of sub-integrals with smooth integrands, i.e., $\int \cdot \, d\Omega_e = \sum_{n_v} \int \cdot \, d\Omega_v$. This keeps the number of degrees of freedom constant, but increases the cost of integration. The associated loss of efficiency may be reduced substantially as presented in [13], where the stiffness contribution of each subdomain defined by a voxel within Ω_e is pre-computed and stored as a voxel-specific sub-stiffness matrix $\mathbf{K}_v$. Upon assembly of the global stiffness matrix, each element stiffness matrix $\mathbf{K}_e$ is then itself assembled from the voxel-specific sub-stiffness matrices $\mathbf{K}_v$. Despite this additional assembly step, the method is very efficient since the pre-integration avoids evaluating the triple matrix product $\mathbf{B}^T \, \mathbf{C} \, \mathbf{B}$ at each Gauss point and replaces it by a sum of $\mathbf{K}_v$ contributions, each weighted by a scalar.

However, the pre-integration scheme of [13] has limitations. It becomes impractical if different sub-summations of $\mathbf{K}_v$ are required, i.e., when different numbers of voxels are subsumed into one element stiffness matrix or when the polynomial degree is varied. More importantly, pre-integration [13] cannot be applied in a straightforward manner when the finite element mesh is refined, thereby precluding the use of adaptivity.

To address these challenges, the paper at hand presents a method to efficiently integrate Equation (1) on immersed voxel geometries. Therein each finite element contains many sub-element voxels, whose values encode either a heterogeneous material distribution that is derived from a CT-scan grey value, if interpreted geometrically, simply defines a voxelized domain. The goal of the method is not only to reduce the cost of numerical integration, but also to enable adaptive refinement while preserving voxel-level material information. To this end, the moment fitting scheme presented in [14–16] is extended and adapted for voxel-based geometries. The moment fitting method significantly reduces the number of required quadrature points during the assembly by computing a

custom integration rule for each finite element. The method is presented in Section 2 and its advantages are demonstrated on a representative set of numerical benchmarks in Section 3. Section 4 summarizes the main conclusions of the paper.

2 Methodology

2.1 Voxel-based moment fitting

Let us consider a physical domain Ω governed by the problem of linear elasticity in strong form, denoted by:

$$\operatorname{div}(\boldsymbol{\sigma}) + \mathbf{b} = 0 \qquad \text{in } \Omega, \tag{2}$$

$$\mathbf{u} = \bar{\mathbf{u}}_n \qquad \text{on } \Gamma_D, \tag{3}$$

$$\boldsymbol{\sigma} \cdot \mathbf{n} = \bar{\mathbf{t}}_n \qquad \text{on } \Gamma_N, \tag{4}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor, $\mathbf{b}$ is the body force, and $\bar{\mathbf{u}}_n$ and $\bar{\mathbf{t}}_n$ are the prescribed displacements and tractions on the Dirichlet (Γ_D) and Neumann (Γ_N) boundaries, respectively. For a heterogeneous material, the constitutive equation relates strains to stresses at each point through the relation $\boldsymbol{\sigma} = \mathbf{C}(\mathbf{x}) : \boldsymbol{\varepsilon}(\mathbf{u})$, where $\mathbf{C}(\mathbf{x})$ is the material tensor which embodies the material variation, e.g., from CT data. In the context of the Finite Cell Method or the traditional voxel-based Finite Element Method, the domain Ω is discretized using finite elements Ω_e, where the bilinear form for a single finite element is $a_e(\mathbf{u}, \mathbf{v}) = \int_{\Omega_e} \boldsymbol{\varepsilon}(\mathbf{v}) : \mathbf{C}(\mathbf{x}) : \boldsymbol{\varepsilon}(\mathbf{u})\, d\Omega_e$. To compute this numerically, we split the material heterogeneity and the displacement approximation such that, for the 1D case, the integral can be written as

$$\int_{\Omega_e} \alpha(x)\, h(x)\, d\Omega_e \tag{5}$$

where Ω_e is the domain of one finite element, $\alpha(x)$ is a discontinuous function representing the scalar-valued material distribution, and $h(x)$ is the remaining smooth part of the integrand of the element stiffness matrix $\mathbf{K}_e$. The main idea of the moment fitting method is to derive customized quadrature rules for each finite element by solving a system of moment-fitting equations [14–16]. In the paper at hand, the original moment fitting is modified such that voxel-level information is included directly in the weights of the integration points. This eliminates the need to access material properties during the integration of element stiffness matrices. Specifically, the scaling contribution of $\alpha(x)$ is absorbed into the numerical integration weights w_i. In 1D, the general concept of the method reads:

$$\int_{\Omega_e} \alpha(x)\, h(x)\, d\Omega_e = \sum_{i=1}^{n} h(x_i)\, w_i, \tag{6}$$

where the objective is to determine the equivalent weights, or *moments*, w_i, such that the numerical evaluation of $h(x)$ corresponds to the original integral. Now, $h(x)$ can be represented by a set of linearly independent basis functions $P = \{f_j\}_{j=1}^{m}$ and their respective coefficients β_j:

$$h(x) = \sum_{j=1}^{m} \beta_j\, f_j(x) \tag{7}$$

Combining Eq. (7) with the left hand side of Eq. (6) yields

$$\int_{\Omega_e} \alpha(x)\, h(x)\, d\Omega_e = \sum_{j=1}^{m} \beta_j \int_{\Omega_e} \alpha(x)\, f_j(x)\, d\Omega_e = \sum_{j=1}^{m} \beta_j \sum_{i=1}^{n} f_j(x_i)\, w_i, \tag{8}$$

where the second equality uses a quadrature rule with appropriate weights w_i to evaluate the integral exactly on the linearly independent basis. Since this expression must hold for any coefficient β_j, the

moment fitting equations can be formulated as:

$$\begin{bmatrix} f_1(x_1) & \cdots & f_1(x_n) \\ \vdots & \ddots & \vdots \\ f_m(x_1) & \cdots & f_m(x_n) \end{bmatrix} \begin{Bmatrix} w_1 \\ \vdots \\ w_n \end{Bmatrix} = \begin{Bmatrix} \int_{\Omega_e} \alpha(x) f_1(x)\, d\Omega_e \\ \vdots \\ \int_{\Omega_e} \alpha(x) f_m(x)\, d\Omega_e \end{Bmatrix} \tag{9}$$

or, written per row,

$$\sum_{i=1}^{n} f_j(x_i)\, w_i = \int_{\Omega_e} \alpha(x)\, f_j(x)\, d\Omega_e, \quad j = 1, \ldots, m. \tag{10}$$

This expression states that the evaluation of the integral of $h(x)$ over the whole domain is equivalent to the sum of linearly independent basis functions evaluated at the integration points x_i and multiplied by their corresponding integration weights w_i. In general, this problem is nonlinear, since both the positions of the integration points and the integration weights must be determined. Therefore, the solution of Eq. (10) is commonly treated as an optimization problem with respect to the geometry of the finite element under consideration [10, 17]. In the present work, however, we focus on finite elements for which the positions of the quadrature points x_i are known. Under this assumption, Eq. (10) is linear [18, 19]. The objective is then to determine the custom quadrature weights w_i for each quadrature grid point x_i. To accelerate the solution process even further, the matrix on the left-hand side of Eq. (10) can be diagonalized by choosing the basis functions $f_j(x)$ to be the Lagrange functions [16] defined at the nodes x_i as:

$$l_j(x) = \prod_{\substack{m=1 \\ m \neq j}}^{n} \frac{x - x_m}{x_j - x_m}. \tag{11}$$

With this choice, Eq. (10) becomes

$$\sum_{i=1}^{n} l_j(x_i)\, w_i = \int_{\Omega_e} \alpha(x)\, l_j(x)\, d\Omega_e, \quad j = 1, \ldots, m. \tag{12}$$

With the property $l_j(x_i) = \delta_{ji}$ and the choice $m = n$, Eq. (12) reduces to (using $\sum_{i=1}^{n} \delta_{ji} w_i = w_j$)

$$w_j = \int_{\Omega_e} \alpha(x)\, l_j(x)\, d\Omega_e, \quad j = 1, \ldots, m, \tag{13}$$

where w_j are called the *moments*. Computing the moments yields a custom integration rule for the exact integration of a function $h(x)$ on Ω_e. The right-hand side of Eq. (13) can be evaluated using different integration methods (see, e.g., [17, 18, 20]), but should be computed with respect to the properties of $\alpha(x)$. When $\alpha(x)$ is defined in a piecewise manner, e.g., from CT-scan data, composed quadrature over smaller subdomains can be used such that these subdomains coincide with the voxel resolution. Equation (13) is then evaluated on each subdomain as

$$w_j = \sum_{k=1}^{n_{vox}} \sum_{g=1}^{n_{gauss}} \alpha(x_{kg})\, l_j(x_{kg})\, w_{kg}, \quad j = 1, \ldots, m, \tag{14}$$

where x_{kg} and w_{kg} are the evaluation points and weights for Gauss-Legendre quadrature on each voxel, respectively.

To illustrate this method, Fig. 1 shows a 1D domain discretized with three elements and eight voxels. The boundaries of the elements are depicted by black circles. The boundaries of the voxels are depicted by vertical lines, whereby the lines are dashed if they are located within the element. A quadratic ($p = 2$) function $h(x)$ multiplied with a discontinuous but piecewise constant function $\alpha(x)$ is to be integrated exactly on every Ω_e. The center of the blue boxes located on the x-axis mark the locations of the integration points on the element level (used for the evaluation of the element stiffness matrix $\mathbf{K}_e$), resulting from the relationship $n = 2p+1$ [16, 18]. The positions of the integration points can be chosen arbitrarily but must be distinct; a natural choice is to place them at the $(2p+1)$-point Gauss-Legendre quadrature point positions. This choice has been shown to give good results [14].

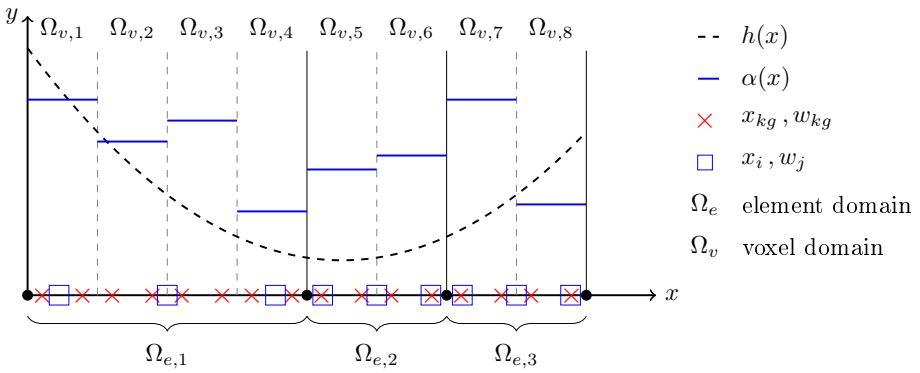


Fig. 1: Exemplary distribution of voxel integration points x_{kg} (red $\times$) and element integration points x_i (blue $\square$) for exact integration of a quadratic discontinuous function $g(x) = \alpha(x)\, h(x)$ on a domain discretized with three finite elements. The Lagrange function $l_j(x)$ is integrated exactly on each voxel k.

These then define the integration points for which the weights are to be determined. The red crosses show the Gauss-Legendre integration points used in the composed-quadrature scheme to evaluate the right-hand side of the moment fitting equation on each voxel Ω_k. The number of integration points per voxel m_v depends on the polynomial degree of the Lagrange basis p_q as $p_q = m_v - 1 \Leftrightarrow m_v = p_q + 1$, and is selected such that the basis functions can be integrated exactly on each voxel.

The overall procedure for computing the weights w_j in a single element is summarized in Algorithm 1. Firstly, all voxels within the element domain Ω_e are identified. For each voxel, a local Gauss-Legendre quadrature is performed to integrate the right-hand side (Eq. (13)), which is the product of the material function $\alpha(x)$ and the Lagrange basis function $l_j(x)$. The sum of these voxel-wise contributions yields the final weights w_j for each element-level integration point. The extension

Algorithm 1 Voxel-based moment fitting for a single element.

Input: Element domain Ω_e, material distribution $\alpha(x)$, Lagrange basis functions $\{l_j(x)\}_{j=1}^{m}$
Output: Moment-fitted weights $\{w_j\}_{j=1}^{m}$
for $j = 1$ **to** m **do**
    $w_j \leftarrow 0$
    for each voxel k with Ω_k **do** (see Eq. (14))
        Distribute Gauss-Legendre points $\{x_{kg}\}$ and weights $\{w_{kg}\}$ within Ω_k
        for each Gauss-Legendre point x_{kg} **do**
            $w_j \leftarrow w_j + \alpha(x_{kg}) \cdot l_j(x_{kg}) \cdot w_{kg}$
        end for
    end for
end for

to a d-dimensional setting is obtained by taking the respective tensor product of the 1D case. The basis functions $f_j(\boldsymbol{x})$ and the positions of the integration points $\boldsymbol{x_i}$ are determined through the tensor product of the $p_q = 2p + 1$ 1D Lagrange basis functions and the positions of the $p_q = 2p + 1$ 1D Gauss-Legendre points, respectively.

When the material is constant within each voxel, taking the value α_k on each of the n_v intervals $[x_k, x_{k+1}]$, $k = 1, \ldots, n_v$, the moments can be pre-integrated once and reused for every element with the same voxel layout. In one dimension, Eq. (13) then reduces to

$$w_j = \int_{\Omega_e} \alpha(x)\, l_j(x)\, dx = \sum_{k=1}^{n_v} \alpha_k \int_{x_k}^{x_{k+1}} l_j(x)\, dx, \qquad j = 1, \ldots, m. \tag{15}$$

These remaining integrals depend only on the Lagrange polynomials and the voxel layout, not on the material, and are therefore pre-computed once into

$$S \in \mathbb{R}^{n_v \times m}, \qquad S_{kj} = \int_{x_k}^{x_{k+1}} l_j(x)\, dx = \sum_{g=1}^{m_v} l_j\left(\frac{r_g+1}{2}\,(x_{k+1}-x_k)+x_k\right) w_g \frac{x_{k+1}-x_k}{2}, \tag{16}$$

which has one row for each voxel and one column for each of the $m = 2p+1$ one-dimensional Lagrange functions ($j = 1, \ldots, m$); here r_g and w_g denote the $m_v = p+1$ Gauss-Legendre points and weights on $[-1, 1]$, which integrate the Lagrange functions exactly. The one-dimensional moments are then simply $w_j = \sum_{k=1}^{n_v} \alpha_k\, S_{kj}$. Because both the Lagrange basis and the integration domain are tensor products of their one-dimensional counterparts, the multidimensional moments factorize into products of matrices S that are pre-computed per direction. In two dimensions, with the same voxelization in both directions, we obtain

$$w_{ij} = \sum_{k=1}^{n_v} \sum_{l=1}^{n_v} \alpha_{kl}\, S_{ki}\, S_{lj}, \qquad i, j = 1, \ldots, m, \tag{17}$$

where the first index of each S runs over the voxels and the second over the Lagrange functions. The general d-dimensional case follows analogously, with one summation and one factor S per spatial direction, $w_{\mathbf{j}} = \sum_{\mathbf{k}} \alpha_{\mathbf{k}} \prod_{c=1}^{d} S_{k_c j_c}$, where $\mathbf{j} = (j_1, \ldots, j_d)$ and $\mathbf{k} = (k_1, \ldots, k_d)$ are the Lagrange and voxel multi-indices. Equation (17) makes the origin of the speed-up explicit: the costly voxel-wise integration of the Lagrange functions is contained entirely in S and is carried out only once, whereas each element merely contracts the precomputed S with its voxel material values $\alpha_{\mathbf{k}}$. Algorithm 2 summarizes the pre-integrated scheme. In contrast to Algorithm 1, where a Gauss quadrature is carried out for each voxel within an element, this voxel-wise quadrature is precomputed once into S, so that we only have to compute the product of d values per voxel (per weight).

Algorithm 2 Pre-integrated voxel-based moment fitting for a single element.

Input: Element domain Ω_e subdivided into voxels, voxel material values $\alpha_{\mathbf{k}} = \alpha(\mathbf{x}_{\mathbf{k}})$, one-dimensional Lagrange basis functions $\{l_j(x)\}_{j=1}^{m}$
Output: Moment-fitted weights $\{w_{\mathbf{j}}\}$
for each one-dimensional voxel $k = 1, \ldots, n_v$ **do** (pre-integration of 1D moments)
  for each one-dimensional Lagrange function $j = 1, \ldots, m$ **do**
    $S_{kj} \leftarrow \int_{\Omega_k} l_j(x)\, dx$ (see Eq. 16)
  end for
end for
$w_{\mathbf{j}} \leftarrow 0$ for all multi-indices $\mathbf{j} = (j_1, \ldots, j_d)$
for each voxel $\mathbf{k} = (k_1, \ldots, k_d)$ in Ω_e **do** (assembly of full voxel moment fitting moments)
  $\alpha_{\mathbf{k}} \leftarrow \alpha(\mathbf{x}_{\mathbf{k}})$
  for each multi-index $\mathbf{j} = (j_1, \ldots, j_d)$ **do**
    $w_{\mathbf{j}} \leftarrow w_{\mathbf{j}} + \alpha_{\mathbf{k}} \prod_{c=1}^{d} S_{k_c j_c}$ (see Eq. (17))
  end for
end for

2.2 Computational complexity estimates of integration

In the following, the computational complexity to integrate the stiffness matrix of one finite element is estimated. To be able to put the complexity of voxel-based moment fitting into perspective, we start by rehearsing the well known Gauss-Legendre integration, continue with composed integration and discuss pre-integration which has often been applied in the voxel-based finite cell method.

2.2.1 Gauss-Legendre integration

The strain displacement operator $\mathbf{B}$ introduced in Equation (1) emerges from the well known transformation of the d-dimensional strong form given in Equation (2) using a Bubnov-Galerkin procedure

and its subsequent discretization with a polynomial degree p. The procedure is well known and not repeated here, since it is found in almost every standard textbook of computational mechanics, see e.g. [21] for an introduction specific to high-order discretizations and the Finite Cell Method. The dimension of the $\mathbf{B}$ operator is:

$$\mathbf{B} \in \mathbb{R}^{n_{\text{voigt}} \times n_{\text{dof}}} \tag{18}$$

where

$$n_{\text{voigt}} = d\,\frac{d+1}{2} \tag{19}$$

$$n_{\text{dof}} \;= n_{\text{u}}(p+1)^d \tag{20}$$

where n_{u} denotes the number of components of the displacement field. For a linear approximation, i.e. $p = 1$ in 3D, we therefore have $\mathbf{B} \in \mathbb{R}^{6\times 24}$. The element stiffness matrix computation on one quadrature point costs

$$N_{\text{ip}} = n_{\text{voigt}} \cdot n_{\text{dof}}^2, \tag{21}$$

whereby we only consider the multiplication of $\mathbf{B}^{\text{T}}\mathbf{B}$. The multiplication with $\mathbf{C}$ is disregarded, because it only adds a lower order term to the cost of matrix-matrix multiplication. Since the product $\mathbf{B}^{\text{T}}\mathbf{B}$ must be carried out and summed up at all quadrature points, the total cost of integrating a stiffness matrix using standard Gauss-Legendre integration with $n_{\text{ip}} = (p+1)^d$ integration points for a single finite element scales as

$$\begin{aligned} N_g &= n_{\text{ip}} && \cdot N_{\text{ip}} \\ &= n_{\text{ip}} && \cdot n_{\text{voigt}} && \cdot n_{\text{dof}}^2 \\ &= (p+1)^d && \cdot d\,\frac{d+1}{2} && \cdot n_{\text{u}}^2 (p+1)^{2d}. \end{aligned} \tag{22}$$

This confirms the p^9-scaling of the cost of integrating the stiffness matrix in p-FEM in 3D.

2.2.2 Composed Gauss-Legendre integration

Let us now consider the cost of composed Gauss-Legendre integration that will simply be termed composed integration in the sequel. Per finite element (i.e. per finite cell) we consider n_v^d voxels and distribute (p+1) Gauss-Legendre integration points per spatial dimension per voxel. Thus, composed integration is n_v^d times more expensive than the standard integration:

$$N_{\text{c}} = n_v^d N_{\text{g}} \tag{23}$$

2.2.3 Pre-integration

In the sequel, we consider pre-integration to be carried out by composed Gauss-Legendre integration with the cost N_{c}. It is not important which method is used for pre-integration as long as it exactly integrates Equation (1) i.e. it must respect the discontinuities introduced by the piecewise constant material distribution defined by the voxels n_v^d in a parametrized fashion. The cost of subsequent assembly is then simply the cost of the product of the material parameters stored in the voxels times the dimensionality of the stiffness matrix $n_{\text{dof}} \times n_{\text{dof}}$, i.e.,:

$$\begin{aligned} N_{\text{p}} = \;& N_{\text{v}} + && n_v^d \cdot n_{\text{dof}}^2 \\ = \;& N_{\text{v}} + && n_v^d \cdot n_{\text{u}}^2 (p+1)^{2d} \end{aligned} \tag{24}$$

Note that the term N_{v} only needs to be evaluated once for a single finite element and can then be re-used in all other finite elements that have the same number of sub-voxels in each dimension.

2.2.4 Voxel-based moment fitting integration

For each finite element $(2p+1)^d$ [16, 18] integration points are distributed within the element at positions x_i whose weights w_j are computed as given in Algorithm 1. To this end, similarly to

composed integration, $(p+1)^d$ integration points are distributed within each voxel. The complexity of the computation of these weights is

$$N_{\mathrm{w}} = \left((p+1) \cdot n_v \cdot (2p+1)\right)^d. \tag{25}$$

The total cost N_{v} is the sum of the computation of the weights N_{w} plus the cost of the integration at one integration point N_{ip} times the number of integration points in the element $(2p+1)^d$:

$$\begin{aligned} N_{\mathrm{v}} &= N_{\mathrm{w}} && +N_{\mathrm{ip}} && \cdot(2p+1)^d \\ &= \left((p+1) \cdot n_v \cdot (2p+1)\right)^d && +d\,\frac{d+1}{2} \cdot n_{\mathrm{u}}^2 (p+1)^{2d} && \cdot(2p+1)^d. \end{aligned} \tag{26}$$

The precomputation of voxel moments eliminates the voxel-wise Gauss quadrature, further reducing the cost of computing the weights to

$$N_{\mathrm{w,pre}} = \left(n_v \cdot (2p+1)\right)^d. \tag{27}$$

Note that Equation (27) omits the $p+1$ factor present in Equation (25). Consequently, the total cost of the voxel-based moment fitting becomes

$$\begin{aligned} N_{\mathrm{v}} &= N_{\mathrm{w,pre}} && +N_{\mathrm{ip}} && \cdot(2p+1)^d \\ &= \left(n_v \cdot (2p+1)\right)^d && +d\,\frac{d+1}{2} \cdot n_{\mathrm{u}}^2 (p+1)^{2d} && \cdot(2p+1)^d. \end{aligned} \tag{28}$$

This optimized version is used throughout the remainder of the paper.

3 Numerical Examples

In this section, the presented integration scheme is applied to various model problems. We show that the proposed approach integrates piecewise discontinuous functions accurately. We also analyze for which configurations voxel-based moment fitting integration outperforms pre-integration and compare both schemes to composed Gauss-Legendre quadrature. We also demonstrate that voxel-based composed integration is applicable to problems with adaptive mesh refinement, a key feature that has not been accomplished in literature for pre-integration and is computationally not sensible for composed Gauss-Legendre quadrature. All methods were implemented and optimized in the open-source framework mlhp [22].

3.1 Inhomogeneous unit cube with hole

3.1.1 Problem setup

Following the example in [13, 23], consider a cube $[-1,1]^3$ with a heterogeneous linear elastic material distribution described by the function $f(x,y,z) = (x+10)^2(y+10)(z+2)$. The domain is voxelized using $768 \times 768 \times 768 \approx 452$ million voxels, and the material distribution function is evaluated at the centroid of each voxel and assigned as a piecewise-constant material parameter within the respective voxel. This mimics a material distribution as expected from a CT-scan. A uniform traction of $\bar{t} = 1.0\,\mathrm{N\,mm^{-2}}$ is applied at $y = 1$. The normal displacements at the boundaries $x = -1$, $y = -1$ and $z = -1$ are set to zero. The Poisson's ratio is assumed to be $\nu = 0.3$ everywhere within the cube. The boundary conditions and the voxelized material distribution are shown in Fig. 2. For the analysis, the domain is discretized with $2 \times 2 \times 2$ up to $384 \times 384 \times 384$ finite elements and shape functions with polynomial degree $p = 1$.

3.1.2 Analysis

Figure 3 shows the strain energy for the voxelized cube under two integration strategies: (a) composed quadrature, i.e., exact Gauss-Legendre quadrature on each voxel, and (b) the presented voxel moment fitting. For all discretization levels, both quadrature methods yield identical strain energies. This result confirms the mathematical equivalence of both integration schemes shown in Eq. (6).

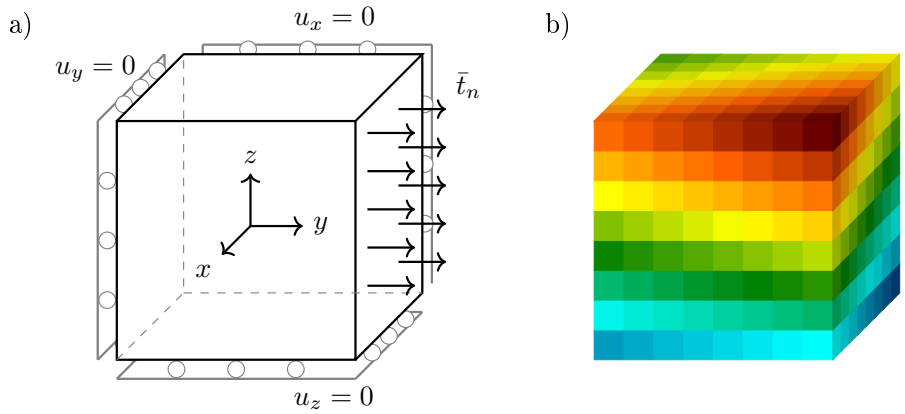


Fig. 2: (a) Simulation model with boundary conditions and (b) example voxelized representation of the material distribution with $n_v = 8$ voxels per side length.

The obtained integration weights w_j, if evaluated exactly on each sub-cell, reproduce exact integration. Note that this expression holds, even if the scaling function $\alpha(x)$ is discontinuous throughout the element domain, and is capable of representing discontinuous material fields without recourse to homogenization or average-stiffness approaches. Figure 4 compares the analytically estimated

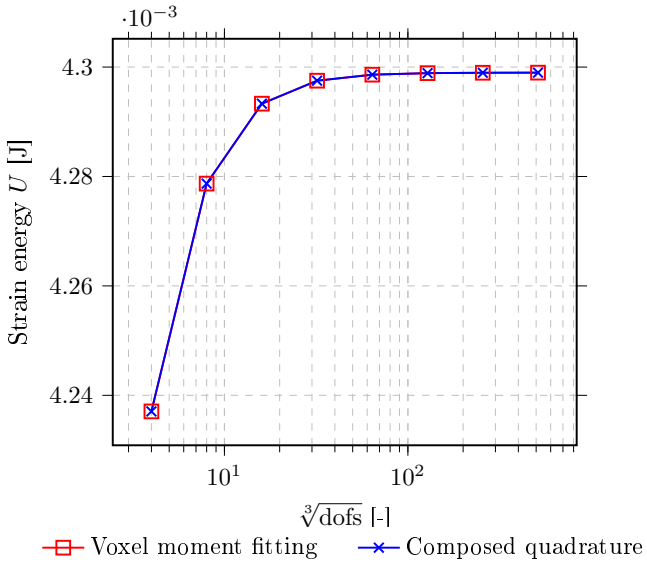


Fig. 3: Convergence graph for the voxel moment fitting integration scheme and the composed quadrature scheme.

operation counts (left) and the experimentally measured runtimes (right) for the three integration strategies. The dominant cost in the evaluation of the element stiffness matrix $\mathbf{K}_e$ scales with the number of element-level integration points at which Eq. (1) must be evaluated. The key advantage of the voxel moment fitting is that voxel contributions are absorbed into the element-integration-point weights w_j, avoiding a costly loop over all voxels during stiffness assembly; see Eqs. (25)–(26). A clear limitation emerges in the limit case of voxelFEM, where each voxel corresponds to a single finite element: the overhead of evaluating the moment-fitting weights then outweighs the savings in element integration, making the approach less efficient than composed quadrature. From the experimental results, the cross-over point between pre-integration and voxel moment fitting lies at approximately 6 voxels per element side length: above this threshold, e.g., more voxels within each finite element, pre-integration is significantly more efficient; below it, voxel moment fitting outperforms pre-integration. For larger polynomial degrees p, the element integration cost $N_{\text{ip}} \cdot (2p+1)^d$ (see Equation (26)) grows

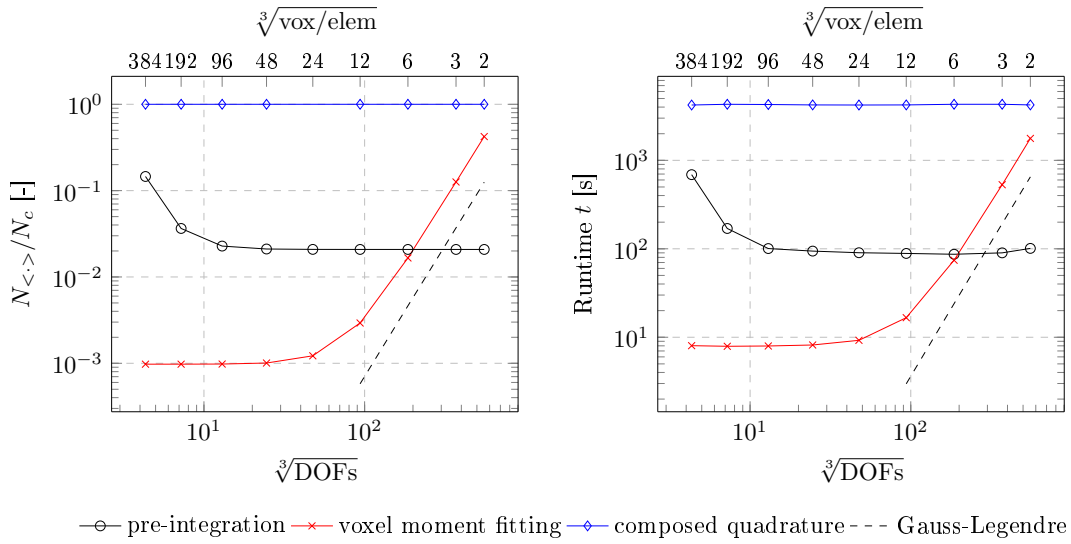


Fig. 4: (left) Estimated number of operations of the quadrature scheme given by the line plot $N_{<\cdot>}$, normalized with respect to the estimate obtained for composed quadrature N_c versus the experimentally measured runtimes t (right) for different integration schemes. The secondary x-axis shows the number of voxels per element side-length.

rapidly, which shifts the break-even point towards more voxels per element side length and causes voxel moment fitting to become less competitive at high p.

Figure 4 (right) shows the experimentally measured runtimes for the cube with the voxelized material distribution and the different integration techniques. The experimental results closely follow the expected analytical estimates. For a lower number of voxels per finite element, voxel moment fitting performs better than both composed quadrature and pre-integration, with a runtime reduction of 99.6% and 89.7% for 384 voxels per domain side length, respectively. Moreover, the additional cost of evaluating the right-hand-side integrals in the custom moment fitting scheme is visible as an offset in runtime compared to standard integration (i.e., distributing Gauss-Legendre points within each element without taking the material distribution into account). In the special case of voxelFEM, composed quadrature and standard quadrature have the same runtime. The runtime for the voxel moment fitting increases drastically because the evaluation of the moments w_j is performed on each voxel, effectively adding significant overhead to the integration cost.

The analytical and experimental runtime comparison shows that the dominant factor in the evaluation of the element stiffness matrix $\mathbf{K}_e$ scales with the number of element integration points at which Eq. (1) has to be evaluated. The main gain in efficiency of the voxel-based moment fitting comes from the term $(2p+1)^d \cdot N_{\text{ip}}$ effectively eliminating the need for iterations over each voxel during the evaluation of $\mathbf{K}_e$. Instead, the contributions of each voxel are absorbed into the element-integration-point weights, avoiding this costly loop. In the experiment, for more than 12 voxels per direction within one finite element, the voxel moment fitting significantly outperforms both conventional composed quadrature and pre-integration. This aligns with the approximate analytical cost. Moreover, the additional computational cost introduced by the evaluation of the moments w_j (see Eq. (13)) is small compared with the element integration cost.

For meshes with local refinement the situation with more then 6^d voxels per element is quite common, please refer to Section 3.3 which shows how fast the number of voxels within a finite element can grow. When each element contains exactly one voxel, the additional quadrature points introduced by moment fitting push the cost past that of standard Gauss quadrature. This case is trivial to catch during runtime and does not represent a limitation of the method. When moving towards multiple voxels within one finite element, the efficiency of the voxel moment fitting increases. Comparing with pre-integration, this yields a break-even point after which it is more feasible to use the voxel moment fitting. In the experimental investigation ($p = 1$ and $n_{v,\text{total}} = 768$ voxels per dimension), this break-even point lies at $n_v = 6$ voxels per element side length. Before that, pre-integration is more efficient than voxel moment fitting. For higher polynomial degrees, this break-even point shifts towards more voxels per element. Analytically, as the polynomial degree approaches infinity ($p \to \infty$),

this break-even point converges to a lower asymptotic bound of $n_v = \sqrt{2\, n_{v,\text{total}}}$, meaning the voxel moment fitting is guaranteed to perform better than the pre-integration, when more than $\sqrt{2\, n_{v,\text{total}}}$ voxels per element side length are embedded within one finite element. For the given voxelization of $n_{v,\text{total}} = 768$ and $p = 1$, this theoretical limit corresponds to approximately 39.2 voxels per element side length.

3.2 L-shaped Domain (2D)

Adaptive schemes improve the efficiency of finite element discretizations by automatically resolving local solution features. We use the multi-level *hp* method, which provides a simple rule set for constructing *hp*-bases [22] and reduces the *hp*-construction to a set of decoupled *p*-constructions on each refinement level. Rather than matching the coarse basis across a hanging node by a combination of finer shape functions, the multi-level *hp* method keeps the coarse functions on the parent element and removes the finer functions at the hanging node. Each leaf then acts as a finite element but carries shape functions from its coarser parents, so the basis differs across the refinement transition zones. Therefore, pre-integration cannot be combined with the multi-level *hp* method. However, the presented voxel-based moment fitting, that is inherently independent of the definition of a basis, can be used directly without modification.

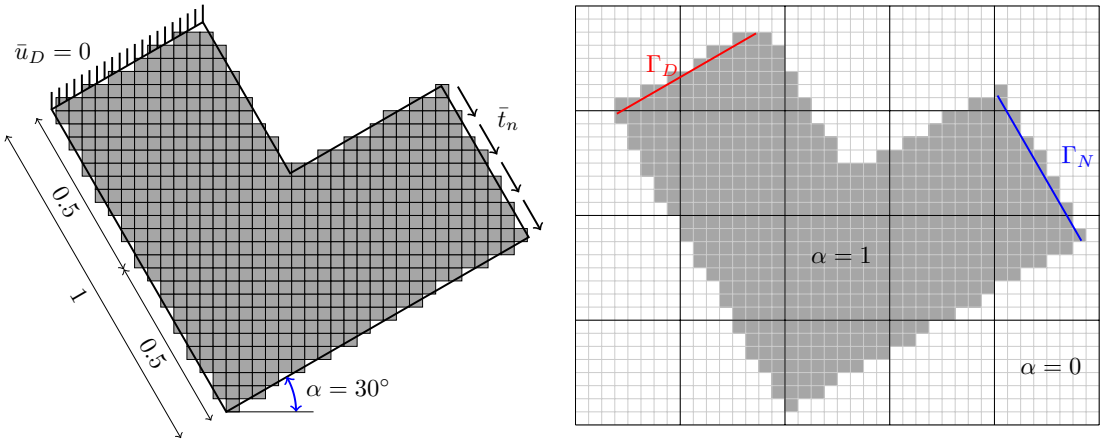


Fig. 5: Rotated L-shaped physical domain and corresponding voxelization with 8 voxels per element side length and boundary conditions (left). The voxel domain is extended to fill a Cartesian finite element grid of 5×4 elements (right).

To demonstrate this, we consider a plane-stress linear elasticity problem on an L-shaped domain, rotated by 30° counterclockwise and voxelized at several resolutions (Figure 5). An inside-outside test at each voxel center assigns the indicator $\alpha = 1$ inside the physical domain and $\alpha = 10^{-10}$ outside. This voxel-wise constant indicator function is the same α introduced in Section 2, so the moment-fitting scheme applies without modification. The material has a Young's modulus of $E = 210\,\text{GPa}$ and a Poisson's ratio of $\nu = 0.29$.

A Dirichlet boundary condition on the top-left edge and a traction on the bottom-right edge produce a corner singularity that local refinement must resolve. To keep both conditions clear of the fictitious domain, they are applied on surfaces offset slightly inward from the exact boundary so that, at every voxel resolution, they fall within the physical voxels. Because the Dirichlet condition is prescribed on this immersed boundary rather than a boundary-fitted one, we impose it weakly by adding a penalty term with parameter β to the weak form [24].

The finite cell mesh is aligned with the voxel grid, associating each voxel with a single finite element until refinement reaches sub-voxel resolution (Fig. 5(b)). We drive local refinement using a residual-based error estimator [25–27], which has been extended to an embedded domain context in [28]. Within the adaptive loop, we estimate the error η of the current finite element approximation in the energy norm as

$$\|\mathbf{u} - \mathbf{u}_h\|_a^2 \leq C \sum_{T \in \mathcal{T}_h} \eta_T^2 \tag{29}$$

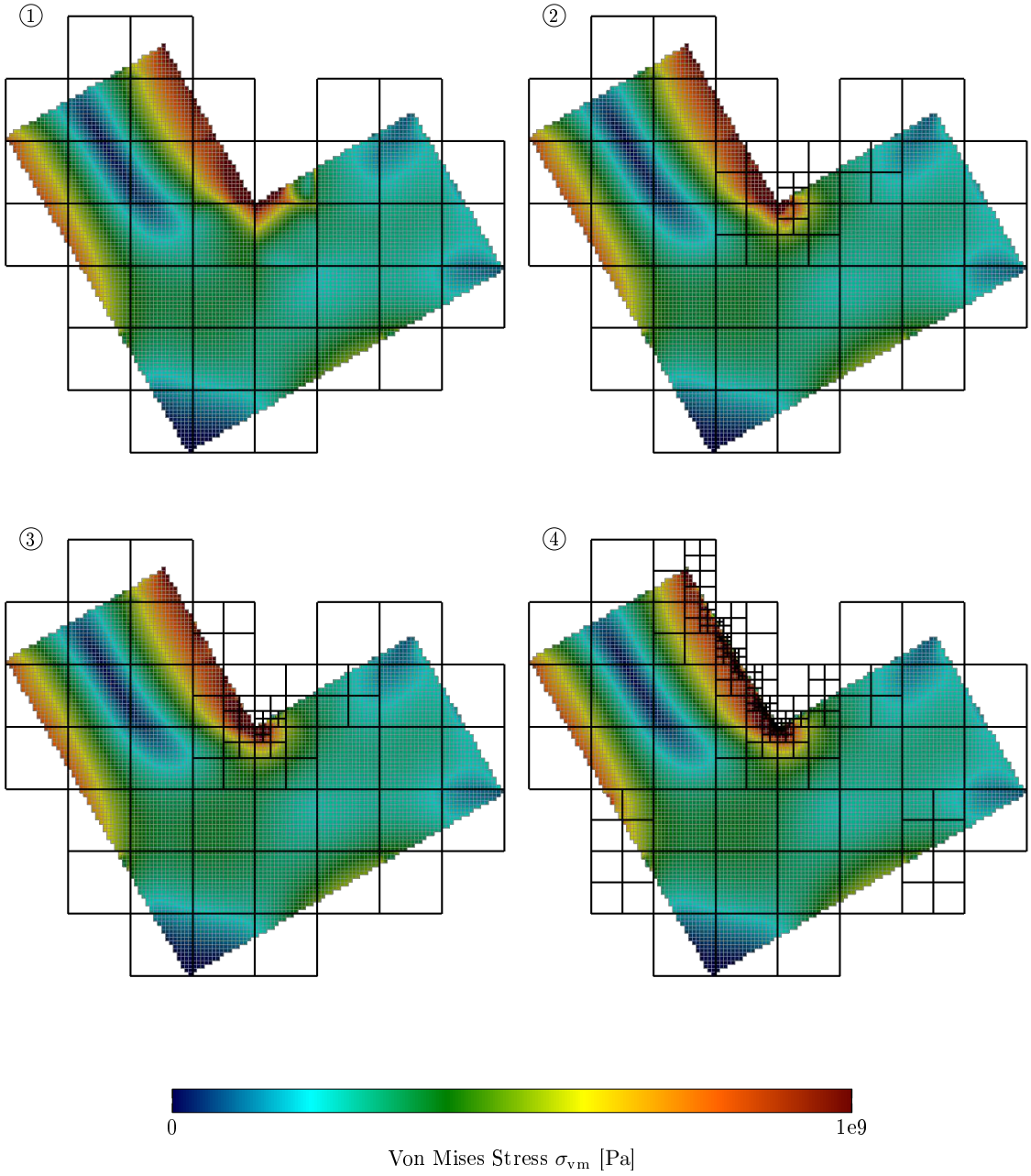


Fig. 6: Discretizations at different iterations for the L-shaped domain for $p = 3$ and 16 voxels per element side length. The low voxel resolution was chosen for visualization purposes. Markers ①–④ correspond to refinement iterations 0, 2, 3, and 8.

where

$$\eta_T^2 = \frac{h_T}{p_T} \|\mathcal{R}_{\partial T}(\mathbf{u}_h)\|^2_{L_2(\partial T \setminus \Gamma)}. \tag{30}$$

This estimator integrates the traction jumps

$$\mathcal{R}_{\partial T}(\mathbf{u}_h) = \tfrac{1}{2}[\![\boldsymbol{\sigma}(\mathbf{u}_h) \cdot \mathbf{n}]\!] \tag{31}$$

across the element interfaces, but disregards the volume and boundary terms commonly included in explicit residual-based error estimators. On element interfaces cut by the domain boundary, the integrand of Equation (30) is discontinuous and would ideally call for adaptive quadrature; we instead overintegrate each interface with $(p+5)^{d-1}$ Gauss points. After assembling the element-wise error

indicator, we mark all elements that satisfy $\eta_T > 0.5\,\eta_{max}$ for refinement, where η_{max} is the largest element error.

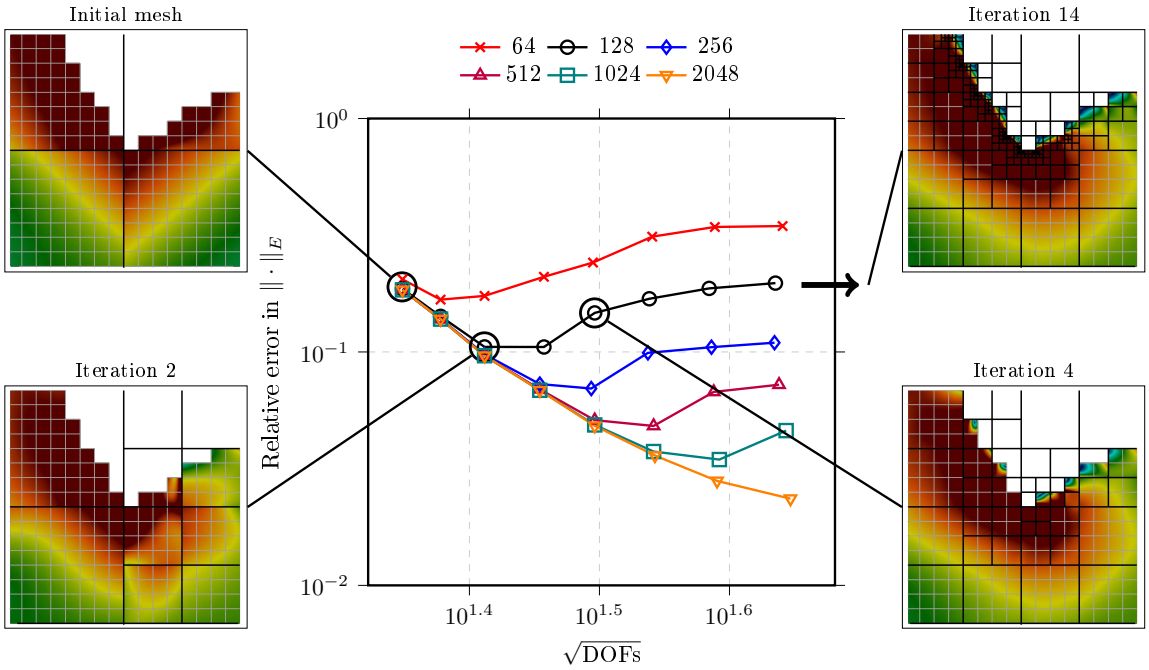


Fig. 7: Convergence of the relative error $\|\mathbf{u} - \mathbf{u}_h\|_E^2$ across different voxel resolutions for $p = 3$. Zoom-ins of the re-entrant corner of the L-shaped domain with the von Mises stress σ_{vm} are shown for iterations 0, 2, 4, and 14 and a voxelization of 128 voxels per element side length. *Note: the error value for iteration* 14 *is omitted from the plot.*

Figure 6 illustrates the evolution of the mesh and the corresponding von Mises stress distribution at refinement iterations 0, 2, 3, and 8 (①–④). Starting from the initial coarse mesh (①), the adaptive procedure refines the grid where the stress jumps between neighboring elements are large, and thus where the estimated error is high. In the early iterations (②, ③), refinement concentrates on the re-entrant corner, where the stress is singular. Once the error at the corner is localized, the estimator begins to pick up the many small re-entrant corners that the voxelized boundary introduces along the inner leg toward the Dirichlet boundary; at iteration 8 (④) this leg is heavily refined.

We now measure the energy error of the finite element solution on the voxelized domain. Two sources contribute: the finite element discretization error and the geometric error from approximating the exact boundary by voxels, and the larger of the two determines the overall error. Since the discrete solution lives on the voxelized domain while the reference is defined on the exact L-shape, we compare them in a modified energy norm

$$\|\mathbf{u} - \mathbf{u}_h\|_E^2 = \int_\Omega (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_h) : (\mathbf{C} : \boldsymbol{\varepsilon} - \mathbf{C}_h : \boldsymbol{\varepsilon}_h)\, d\Omega = \int_\Omega (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_h) : (\boldsymbol{\sigma} - \boldsymbol{\sigma}_h)\, d\Omega, \tag{32}$$

where $\mathbf{C}_h$ equals the material matrix $\mathbf{C}$ inside the physical domain and vanishes in the fictitious domain, so that the discrete stress $\boldsymbol{\sigma}_h := \mathbf{C}_h : \boldsymbol{\varepsilon}_h$ is nonzero only over the physical material; $\boldsymbol{\varepsilon}_h$ is the discrete strain. The reference solution is obtained with a boundary-conforming discretization, $p = 5$, initial element size $h = 0.25$, and 20 iterations of adaptive refinement.

Figure 7 shows the relative error against $\sqrt{\text{DOFs}}$ for $p = 3$ at several voxel resolutions, together with zoom-ins of the re-entrant corner at successive refinement iterations. All resolutions initially follow the same curve, since the discretization error dominates and is independent of the voxelization, in agreement with the a priori estimates [27]. Each curve then flattens, and eventually rises, once the refinement reaches the voxel size and the geometric error of the voxelized boundary takes over; the finer the voxelization, the longer the discretization-dominated rate persists before this plateau.

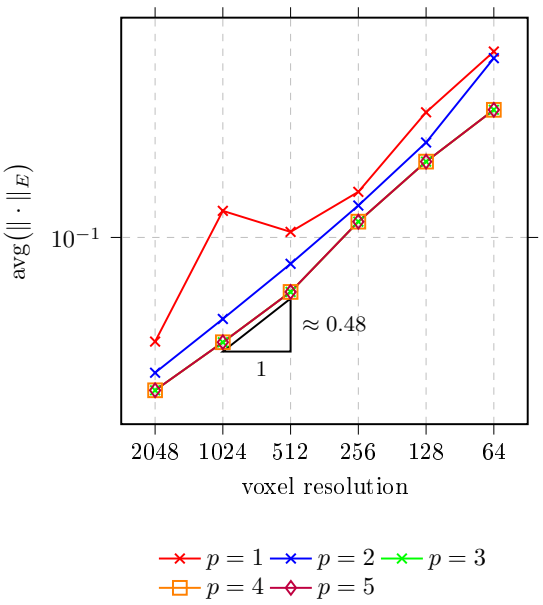


Fig. 8: Convergence of the average relative error $\|\mathbf{u} - \mathbf{u}_h\|_E^2$, taken over the refinement steps after the convergence curve has hit the geometric-error plateau, for different polynomial degrees.

The zoom-ins make the transition visible: by iteration 14, the refinement has begun to subdivide individual voxels.

To examine the geometric error, Figure 8 plots the relative energy error, averaged over the refinement steps beyond the onset of the plateau, against the voxel resolution for $p = 1, \ldots, 5$. In contrast to the mesh-refinement convergence of Figure 7, this plateau decreases at a rate of approximately $\frac{1}{2}$ as the voxelization is refined (Fig. 8, right), as expected for a voxelized geometry approximation [9]. High-order elements are therefore most useful at fine voxel resolutions: the geometric limit then lies well into the asymptotic regime, where high-order discretizations are most efficient, whereas a coarse voxelization may reach the plateau before high order can pay off. Refining beyond the voxel size no longer reduces the error of the intended problem; instead, it resolves the voxelized geometry, in which every voxel corner acts as a singularity. In practice this is easy to detect, since one can simply check whether elements would refine below the voxel size.

3.3 Phase-field fracture of a notched plate with a hole (3D)

The final example demonstrates how the proposed integration scheme efficiently handles the computational challenges of phase-field fracture in materials with spatially varying properties. The main idea of the phase-field approach to quasi-static brittle fracture is to represent a discrete crack through a smooth scalar variable $s(\boldsymbol{x}, t) \in [0, 1]$, as shown in Fig. 9, where zero denotes fully damaged material and one represents intact material. The propagation of the crack front is described by the

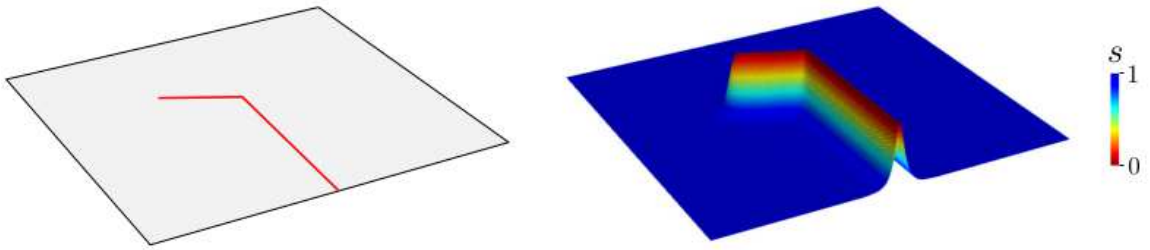


Fig. 9: Explicit crack (left) and phase-field regularized crack field (right). The width of the crack is controlled by the length-scale parameter l_0 [29].

minimization of the following functional [30]:

$$E_{l_0}(\boldsymbol{u}, s) = \int_\Omega g(s)\,\Psi(\boldsymbol{\varepsilon})\,d\boldsymbol{x} + \frac{G_c}{c_w}\int_\Omega \left(\frac{1}{2l_0}w(s) + 2l_0|\nabla s|^2\right) d\boldsymbol{x}, \tag{33}$$

where $g(s)$ is the degradation function responsible for modeling the loss of stiffness in the material, G_c is the critical energy release rate, c_w is a scaling parameter, and $w(s)$ is the energy dissipation function. The width of the crack is controlled by the regularization length l_0. Including the work done by external forces, adopting an isotropic AT-2 formulation, and following the variational principle (for details, see, e.g., [31, 32]), Eq. (33) can be reformulated as the following system of coupled equations:

$$\operatorname{div}(\hat{\boldsymbol{\sigma}}) + \rho\boldsymbol{b} = 0, \tag{34a}$$

$$-4l_0^2\Delta s + s + 2l_0\frac{\mathcal{H}}{G_c}g'(s) = 1, \tag{34b}$$

where eq. (34a) governs mechanical equilibrium and eq. (34b) describes crack evolution. Here, $\mathcal{H}$ is the history variable that ensures irreversibility (i.e., no crack healing) of the phase field, and is defined as $\mathcal{H} := \max_{t\in[0,T]}\Psi^+(\boldsymbol{\varepsilon}(\boldsymbol{x}, t))$ [33]. The problem is subject to the boundary conditions:

$$\boldsymbol{u} = \bar{\boldsymbol{u}}_n \qquad \forall\,\boldsymbol{x} \in \Gamma_D, \tag{35a}$$

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \bar{\boldsymbol{t}}_n \qquad \forall\,\boldsymbol{x} \in \Gamma_N, \tag{35b}$$

$$\nabla s\cdot\boldsymbol{n} = 0 \qquad \forall\,\boldsymbol{x} \in \Gamma_D \cup \Gamma_N. \tag{35c}$$

The degraded stress tensor for the isotropic, hybrid formulation is $\hat{\boldsymbol{\sigma}} = g(s)\,\partial\Psi(\boldsymbol{\varepsilon})/\partial\boldsymbol{\varepsilon}$ and presented in [34, 35]. The isotropic formulation enables the solution through a staggered scheme, as in [10, 31, 32]. In addition to rendering the elastic equation linear, the isotropic hybrid formulation lets the degradation enter as a scalar factor $g(s)$ at the element integration points. With the voxel moment fitting providing the weights w_i that absorb the voxel material as before, the element stiffness is assembled as

$$\mathbf{K}_e = \sum_i g(s(\mathbf{x}_i))\,\mathbf{B}^T(\mathbf{x}_i)\,\mathbf{C}\,\mathbf{B}(\mathbf{x}_i)\,w_i, \tag{36}$$

where $\mathbf{x}_i$ are the $(2p+1)^d$ element integration points. Because $g(s)$ raises the polynomial degree of the integrand above $2p$, this rule no longer integrates the degraded stiffness exactly; in our simulations the limit loads and the crack path are nonetheless largely unaffected, so the underintegration is acceptable. For the integration of the phase-field equation, standard Gauss-Legendre quadrature is used, because no discontinuous material parameters influence the phase field except the critical energy release rate G_c, which is assumed to be constant throughout the specimen. As a test case,

Table 1: Simulation and material parameters for the phase-field problem.

E [kN mm^{-2}]	ν [-]	G_c [N mm^{-1}]	v_x, v_y, v_z [-]	p [-]	k [-]	l_0 [mm]	Load steps [-]
1.8–6.0	0.22	0.0027	832, 1536, 320	1	4	1.25	50

the notched plate with a circular hole from [10, 32, 34] is considered. The geometry and boundary conditions are shown in Fig. 10. The domain is voxelized with $832 \times 1536 \times 320$ voxels in a similar fashion as the L-shaped domain from Section 3.2, and, in addition, a heterogeneous Young's modulus distribution mimicking a layer-wise material is assigned on a per-voxel basis. To avoid issues with crack nucleation, an initial crack is introduced by locally modifying the history variable $\mathcal{H}$ and solving the initial phase-field problem. An initial discretization with $13 \times 24 \times 10$ elements is used; on the unrefined mesh, each element (the *root element* in octree terminology) contains $64 \times 64 \times 32$ voxels. With $k = 4$ refinement levels, this corresponds to $4\times4\times2$ voxels on the finest level. The base material and simulation parameters are summarized in Tab. 1.

Figure 11 shows the load–displacement curve for the problem. It is evident that the characteristics of the heterogeneous fracture process of the 3D problem are captured. Fig. 12 shows the developed

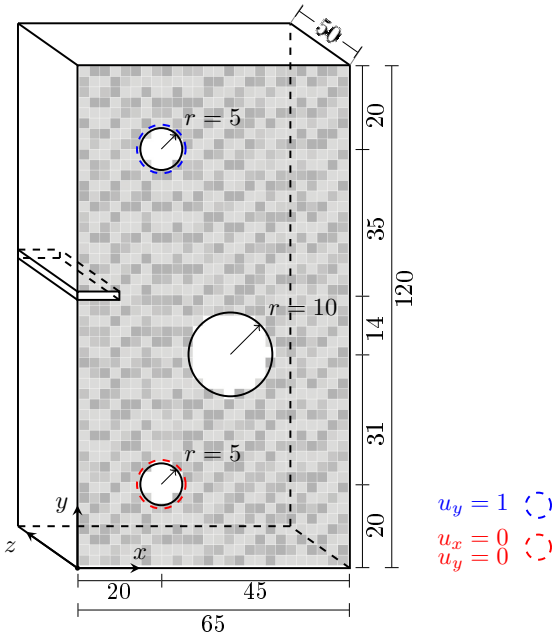


Fig. 10: Simulation setup, boundary conditions, and material distribution. A darker voxel color represents a higher Young's modulus E. To prevent boundary effects, the radii of both circular boundaries (red and blue) are increased slightly so as to lie completely within the voxel domain. All dimensions are given in mm. Note that the coarse voxelization shown is for illustration purposes only; the voxelization used for the computation, with $832 \times 1536 \times 320$ voxels, is significantly finer.

crack at different time steps of the simulation for $p = 1$ and the voxel moment fitting. In addition to the phase-field variable s, the mesh is depicted to show the refinement around the crack path. At point ① on the load–displacement curve, a crack starts to propagate towards a pore, deviating from the linear elastic response and decreasing the reaction force. In the second and third stages (②, ③), the crack propagates through pores, visible in the load–displacement curve as abrupt decreases in the force with subsequent small recoveries until the next rupture. Finally, at ④, the crack has propagated through the domain and the reaction force is zero. The complete simulation of the 50 load steps with roughly 2 500 staggered iterations in total took 30 hours and 15 minutes for the voxel moment fitting on a 64-core CPU at 5.4 GHz. For reference, a single iteration for the composed quadrature took 68 minutes. The corresponding full scale simulation would have taken 118 days and was therefore not carried out. Immediately it is clear, that without the presented integration scheme, such examples are not computable.

4 Conclusion

This work presents a voxel moment-fitting scheme for the accurate and efficient evaluation of finite element integrals on immersed voxel data. The scheme absorbs the per-voxel material or indicator values directly into custom quadrature weights, reproducing the exact value of composed quadrature while using far fewer integration points. The integration points stay at Gauss-Legendre positions, although $2p+1$ per direction are needed for integrands of degree $2p$. They form a tensor-product grid, which opens the scheme to combination with methods that exploit this structure, such as matrix-free operators with fast Kronecker-product (sum-factorization) algorithms. Apart from the fitted weights, no change to a standard finite element code is required.

A scaling analysis and runtime measurements place the break-even against pre-integration at about six voxels per element side length for $p = 1$: below it pre-integration is cheaper, above it voxel moment fitting is faster, by up to a factor of eleven in our tests, while remaining far cheaper than composed quadrature throughout.

Because the scheme does not depend on the finite element basis, it can be combined with locally enriched discretizations such as multi-level *hp* or the extended finite element method, for which pre-integrated quantities cannot be reused, and is therefore well suited to adaptive refinement. We also quantify how the discretization and geometry errors interact: at a fixed voxel resolution, refining the

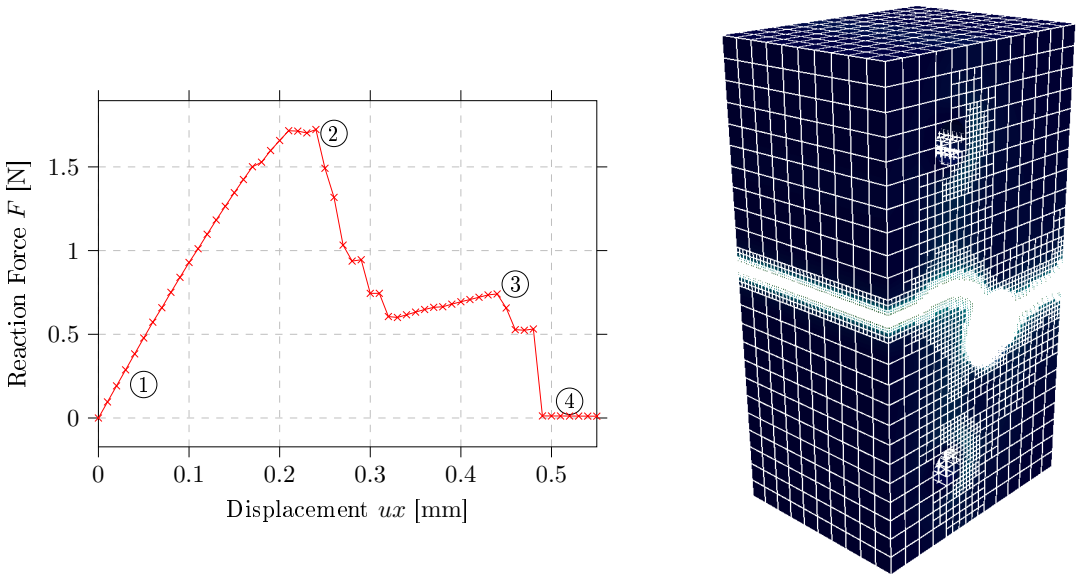


Fig. 11: Load–displacement curve for the Phase-Field simulation (left) along with the mesh for the final load step (right).

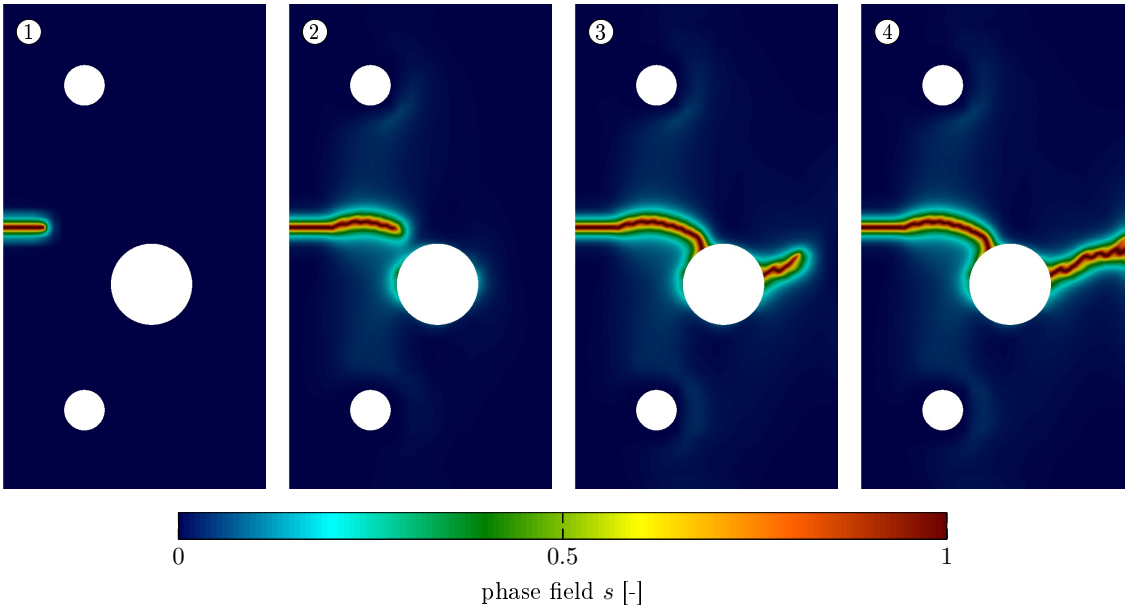


Fig. 12: Crack patterns at different displacement steps for the tensile loading, computed with the voxel moment fitting and the simulation parameters shown in Tab. 1. The steps 1–4 are also marked on the load–displacement curve in Fig. 11.

mesh reduces the error at the expected convergence rate until it reaches a plateau set by the voxelized geometry, and this plateau decreases at a rate of about one half as the voxelization is refined. This yields the concrete recommendation to stop adaptive refinement once individual voxels start being resolved, since beyond that point refinement only resolves the voxelized geometry rather than the intended one.

Finally, a phase-field fracture example demonstrates the scheme on a large three-dimensional problem that composed quadrature would render infeasible, reducing an estimated 118 days of computation to about 30 hours. There the degradation enters as a scalar factor at the element integration

points, raising the integrand degree above the order the rule integrates exactly, so that the element stiffness is underintegrated. Nonetheless, the limit loads and crack path are largely unaffected, indicating that the method tolerates an imperfect integrand rather than breaking down.

Declarations

Funding. The research presented in this paper is being conducted within the project "Bridging Scales – From Geometric Part Details to Construction Elements (C01)". The project is part of the collaborative research centre "Additive Manufacturing in Construction - The Challenge of Large Scale", funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 414265976 - TRR 277.

Competing interests. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author contributions. **Matti Christmann:** conceptualization, formal analysis, investigation, methodology, software, validation, visualization, writing - original draft, writing - review and editing. **Philipp Kopp:** conceptualization, methodology, software, validation, visualization, supervision, writing - review and editing. **Stefan Kollmannsberger:** conceptualization, funding acquisition, project administration, supervision, resources, writing - review and editing.

Code and data availability. The current state of the source code for the voxel moment-fitting implementation and the drivers used to generate the numerical examples have been archived on Zenodo at 10.5281/zenodo.20746104. Further bugfixes and developments will be made available at https://gitlab.com/hpfem/publications/2026_voxelbasedmomentfitting.

Generative AI declaration. During the preparation of this paper, the authors used Anthropic's Claude (model Opus 4.8, accessed June 2026) to reorganize notes and improve the readability of the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

[1] Willot, F.: Fourier-based schemes for computing the mechanical response of composites with accurate local fields. Comptes Rendus Mécanique **343**(3), 232–245 (2015) https://doi.org/10.1016/j.crme.2014.12.005

[2] Lucarini, S., Upadhyay, M.V., Segurado, J.: Fft based approaches in micromechanics: fundamentals, methods and applications. Modelling and Simulation in Materials Science and Engineering **30**(2), 023002 (2021) https://doi.org/10.1088/1361-651X/ac34e1

[3] Yosibash, Z., Myers, K., Trabelsi, N., Sternheim, A.: Autonomous FEs (AFE) - A stride toward personalized medicine. Computers & Mathematics with Applications **80**(11), 2417–2432 (2020) https://doi.org/10.1016/j.camwa.2020.03.012

[4] Hug, L., Dahan, G., Kollmannsberger, S., Rank, E., Yosibash, Z.: Predicting fracture in the proximal humerus using phase field models. Journal of the Mechanical Behavior of Biomedical Materials **134**, 105415 (2022) https://doi.org/10.1016/j.jmbbm.2022.105415

[5] Ferguson, O.V., Mikkelsen, L.P.: Three-dimensional finite element modeling of anisotropic materials using X-ray computed micro-tomography data. Software Impacts **17**, 100523 (2023) https://doi.org/10.1016/j.simpa.2023.100523

[6] Saillard, E., Gardegaront, M., Levillain, A., Bermond, F., Mitton, D., Pialat, J.-B., Confavreux, C., Grenier, T., Follet, H.: Finite element models with automatic computed tomography bone segmentation for failure load computation. Scientific Reports **14**(1), 16576 (2024) https://doi.org/10.1038/s41598-024-66934-w . Accessed 2026-04-09

[7] Zhang, Y.J.: Geometric Modeling and Mesh Generation from Scanned Images. Chapman and Hall/CRC, ??? (2018). https://doi.org/10.1201/9781315372259

[8] Martinez-Weissberg, S., Pazner, W., Yosibash, Z.: Finite element analysis of very large bone models based on micro-CT scans (2025). https://arxiv.org/abs/2512.24401

[9] Düster, A., Parvizian, J., Yang, Z., Rank, E.: The finite cell method for three-dimensional problems of solid mechanics. Computer Methods in Applied Mechanics and Engineering **197**(45–48), 3768–3782 (2008) https://doi.org/10.1016/j.cma.2008.02.036 . Accessed 2014-05-26

[10] Hosseini, S.F., Gorji, M., Sartorti, R., Radtke, L., Düster, A.: Application of the non-negative moment fitting approach in the finite cell analysis of brittle phase-field fracture. International Journal of Computational Methods **22**(10), 2550022 (2025) https://doi.org/10.1142/S0219876225500227

[11] Korshunova, N., Papaioannou, I., Kollmannsberger, S., Straub, D., Rank, E.: Uncertainty quantification of microstructure variability and mechanical behavior of additively manufactured lattice structures. Computer Methods in Applied Mechanics and Engineering **385**, 114049 (2021) https://doi.org/10.1016/j.cma.2021.114049

[12] De Biasi, R., Oztoprak, O., Zanini, F., Carmignato, S., Kollmannsberger, S., Benedetti, M.: Predicting fatigue life of additively manufactured lattice structures using the image-based finite cell method and average strain energy density. Materials & Design **246**, 113321 (2024) https://doi.org/10.1016/j.matdes.2024.113321

[13] Yang, Z., Ruess, M., Kollmannsberger, S., Düster, A., Rank, E.: An efficient integration technique for the voxel-based finite cell method. International Journal for Numerical Methods in Engineering **91**(5), 457–471 (2012) https://doi.org/10.1002/nme.4269

[14] Müller, B., Kummer, F., Oberlack, M.: Highly accurate surface and volume integration on implicit domains by means of moment-fitting. International Journal for Numerical Methods in Engineering **96**(8), 512–528 (2013) https://doi.org/10.1002/nme.4569 https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.4569

[15] Garhuom, W., Düster, A.: Non-negative moment fitting quadrature for cut finite elements and cells undergoing large deformations. Computational Mechanics **70**(5), 1059–1081 (2022) https://doi.org/10.1007/s00466-022-02203-9

[16] Hubrich, S., Düster, A.: Numerical integration for nonlinear problems of the finite cell method using an adaptive scheme based on moment fitting. Computers & Mathematics with Applications **77**(7), 1983–1997 (2019) https://doi.org/10.1016/j.camwa.2018.11.030

[17] Hubrich, S., Di Stolfo, P., Kudela, L., Rank, E.: Numerical integration of discontinuous functions: moment fitting and smart octree. Computational Mechanics **60**(6), 863–881 (2017) https://doi.org/10.1007/s00466-017-1441-0

[18] Joulaian, M., Hubrich, S., Düster, A.: Numerical integration of discontinuities on arbitrary domains based on moment fitting. Computational Mechanics **57**, 979–999 (2016) https://doi.org/10.1007/s00466-016-1273-3

[19] Düster, A., Allix, O.: Selective enrichment of moment fitting and application to cut finite elements and cells. Computational Mechanics **65**(2), 429–450 (2020) https://doi.org/10.1007/s00466-019-01776-2

[20] Abedian, A., Düster, A.: Equivalent legendre polynomials: Numerical integration of discontinuous functions in the finite element methods. Computer Methods in Applied Mechanics and Engineering **343**, 690–720 (2019) https://doi.org/10.1016/j.cma.2018.08.002

[21] Düster, A., Rank, E., Szabó, B.: The p -Version of the Finite Element and Finite Cell Methods. In: Stein, E., Borst, R., Hughes, T.J.R. (eds.) Encyclopedia of Computational Mechanics Second Edition, pp. 1–35. John Wiley & Sons, Ltd, ??? (2017). https://doi.org/10.1002/9781119176817.ecm2003g . https://onlinelibrary.wiley.com/doi/10.1002/9781119176817.ecm2003g Accessed 2022-11-23

[22] Kopp, P., Rank, E., Calo, V.M., Kollmannsberger, S.: Efficient multi-level hp-finite elements in arbitrary dimensions. Computer Methods in Applied Mechanics and Engineering **401**, 115575 (2022) https://doi.org/10.1016/j.cma.2022.115575

[23] Ruess, M., Tal, D., Trabelsi, N., Yosibash, Z., Rank, E.: The finite cell method for bone simulations: verification and validation. Biomechanics and Modeling in Mechanobiology **11**(3), 425–437 (2012) https://doi.org/10.1007/s10237-011-0322-2

[24] Zander, N., Kollmannsberger, S., Ruess, M., Yosibash, Z., Rank, E.: The finite cell method for linear thermoelasticity. Computers & Mathematics with Applications **64**(11), 3527–3541 (2012) https://doi.org/10.1016/j.camwa.2012.09.002

[25] Melenk, J.M., Wohlmuth, B.I.: On residual-based a posteriori error estimation in hp-fem. Advances in Computational Mathematics **15**(1-4), 311–331 (2001) https://doi.org/10.1023/A:1014268310921

[26] Melenk, J.M.: hp-interpolation of nonsmooth functions and an application to hp-a posteriori error estimation. SIAM Journal on Numerical Analysis **43**(1), 127–155 (2005) https://doi.org/10.1137/S0036142903432930

[27] D'Angella, D., Zander, N., Kollmannsberger, S., Rank, E.: Multi-level hp-adaptivity and explicit error estimation. Advances in Modeling and Simulation in Engineering Sciences **3**(1), 33 (2016) https://doi.org/10.1186/s40323-016-0085-5

[28] Di Stolfo, P., Schröder, A.: Reliable residual-based error estimation for the finite cell method. Journal of Scientific Computing **87**(1), 12 (2021) https://doi.org/10.1007/s10915-021-01417-y

[29] Gerasimov, T., De Lorenzis, L.: On penalization in variational phase-field models of brittle fracture. Computer Methods in Applied Mechanics and Engineering **354**, 990–1026 (2019) https://doi.org/10.1016/j.cma.2019.05.038

[30] Bourdin, B., Francfort, G.A., Marigo, J.-J.: The variational approach to fracture. Journal of Elasticity **91**(1), 5–148 (2008) https://doi.org/10.1007/s10659-007-9107-3

[31] Hug, L., Kollmannsberger, S., Yosibash, Z., Rank, E.: A 3d benchmark problem for crack propagation in brittle fracture. Computer Methods in Applied Mechanics and Engineering **364**, 112905 (2020) https://doi.org/10.1016/j.cma.2020.112905

[32] 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. Computational Mechanics **63**(6), 1283–1300 (2019) https://doi.org/10.1007/s00466-018-1649-7

[33] Miehe, C., Welschinger, F., Hofacker, M.: Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field fe implementations. International Journal for Numerical Methods in Engineering **83**(10), 1273–1311 (2010) https://doi.org/10.1002/nme.2861

[34] Ambati, M., Gerasimov, T., De Lorenzis, L.: A review on phase-field models of brittle fracture and a new fast hybrid formulation. Computational Mechanics **55** (2014) https://doi.org/10.1007/s00466-014-1109-y

[35] Amor, H., Marigo, J.-J., Maurini, C.: Regularized formulation of the variational brittle fracture with unilateral contact: Numerical experiments. Journal of the Mechanics and Physics of Solids **57**(8), 1209–1229 (2009) https://doi.org/10.1016/j.jmps.2009.04.011