A Jacobian-free Newton-Krylov method for high-order cell-centred finite volume solid mechanics

Ivan Batistić[1,2], Pablo Castrillo[1,3], Philip Cardiff[1*]

[1*]School of Mechanical and Materials Engineering, University College Dublin, Ireland.

[2]Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Croatia.

[3]Instituto de Estructuras y Transporte, Facultad de Ingeniería, Universidad de la República, Uruguay.

*Corresponding author. E-mail: philip.cardiff@ucd.ie

Abstract

This work extends the application of Jacobian-free Newton–Krylov (JFNK) methods to higher-order cell-centred finite-volume formulations for solid mechanics. While conventional cell-centred schemes are typically restricted to second-order accuracy, we present third- and fourth-order formulations that employ local least-squares reconstructions for gradient evaluation combined with Gaussian quadrature at cell faces for flux integration. These schemes enable accurate resolution of complex stress and deformation fields in both linear and nonlinear solid mechanics, while retaining the flexibility and geometric generality of finite-volume methods. A novel contribution of this study is the development and assessment of a JFNK solution strategy for these higher-order formulations, which eliminates the need to assemble and store large Jacobian matrices explicitly. Instead, we demonstrate that a compact-stencil approximate Jacobian can be effectively employed as a preconditioner, mirroring the efficiency gains observed in second-order frameworks. The proposed methodology is benchmarked across a suite of two- and three-dimensional test problems involving elastic and nonlinear materials, where key performance metrics, including accuracy, computational cost, memory usage, and robustness, are systematically evaluated. Results confirm that the higher-order formulations deliver substantial accuracy improvements over second-order schemes, while the JFNK approach achieves strong performance and scalability with only minimal modifications to existing segregated frameworks. These findings underscore the potential of combining higher-order finite-volume methods with JFNK solvers to advance the state of the art in computational solid mechanics. The implementations are openly released in the solids4foam toolbox for OpenFOAM, supporting further exploration and adoption by the community.

Keywords: Jacobian-free Newton-Krylov, higher-order, cell-centred finite volume method, solid mechanics, solids4foam, OpenFOAM

1 Introduction

High-order finite-volume discretisations are generally defined as schemes that achieve at least third-order accuracy [1]. Over the past two decades, such methods have attracted growing interest within the computational fluid dynamics (CFD) community, leading to continuous development and refinement [2–5]. In contrast, the application of high-order FV discretisations to solid mechanics remains relatively underexplored [6–8], despite several recent advances. The untapped potential of these formulations is significant: for a given level of accuracy, high-order methods can be computationally more efficient and reduce data movement, an increasingly important factor for large-scale simulations on high-performance computing systems, where memory bandwidth is often the main bottleneck. As noted in [9], these two criteria, computational efficiency and reduced data movement, are guiding principles in the development of next-generation discretisations. Within the solid mechanics community, the development of such methods lags decades behind the finite-element counterparts [10]. Their greatest promise lies in multiphysics simulations, where solving the coupled problem within a unified finite volume framework is particularly advantageous.

The first high-order, cell-centred finite volume method for solid mechanics was introduced by Demirdžić [6], who also pioneered the general finite volume approach for solids [11, 12]. In his work, a fourth-order discretisation for two-dimensional structured hexahedral meshes was proposed, and it was shown that shear locking is not inherent to high-order finite volume formulations but is a limitation of conventional second-order schemes. However, Demirdžić's formulation was not easily generalisable, and he suggested adopting a Moving Least-Squares (MLS) reconstruction, similar to those already used in CFD [13, 14]. This step was realised later by Castrillo et. al. [7, 15], where it was demonstrated that higher-order accuracy can be achieved through gradient reconstruction using least-squares interpolation, specifically the Local Regression Estimator (LRE) technique, and face-flux integration via Gaussian quadrature.

A major distinction between these studies lies in their solution strategies. Demirdžić [6] employed a semi-implicit, segregated framework with Picard iterations, whereas Castrillo et. al. [7, 8] adopted a fully implicit, block-coupled Newton–Raphson approach. While segregated solvers are simpler to implement, they typically exhibit only linear convergence and limited robustness. The Newton–Raphson method, in contrast, achieves quadratic convergence but requires the explicit formation and storage of the Jacobian matrix, which is both computationally expensive and memory-intensive. Moreover, constructing an exact Jacobian for nonlinear material models is often non-trivial, further complicating implementation.

To overcome these limitations, Jacobian-free Newton–Krylov (JFNK) methods have emerged as a powerful alternative. JFNK algorithms retain the quadratic convergence properties of Newton's method while avoiding explicit Jacobian construction by estimating matrix-vector products through finite differencing of the residual. These methods were first introduced in the 1980s and early 1990s for solving ordinary and partial differential equations [16–19], and subsequently applied to the Navier–Stokes equations with various preconditioners and linear solvers [20–23]. Despite their potential, JFNK methods are still not widely adopted in CFD, although some codes provide them as built-in options [24]. Of particular relevance is that recent developments in high-order finite volume schemes increasingly rely on JFNK formulations, due to their natural compatibility with compact-stencil preconditioners and flexible treatment of nonlinearities [2]. Notably, the

preconditioning matrix can be constructed from a lower-order spatial discretisation, which provides a good approximation to the true Jacobian while maintaining computational efficiency and robustness [20–23, 25–29].

The application of JFNK methods to cell-centred FV solid mechanics was first demonstrated by the present authors in previous work [30]. There, it was shown that a compact-stencil approximate Jacobian is highly effective as a preconditioner and that the resulting JFNK solution procedure, when combined with a second-order finite volume discretisation, offers a feasible, efficient, and robust framework applicable to static, dynamic, linear, and nonlinear solid mechanics problems. However, the combination of JFNK with high-order finite volume formulations for solids has remained unexplored prior to the current study.

The present work addresses this gap by extending the JFNK framework to third- and fourth-order cell-centred finite volume formulations for both linear and nonlinear solid mechanics. High-order accuracy is achieved through least-squares-based gradient reconstruction and high-order flux integration, following the approaches validated in prior studies [7, 8]. We show that even in this higher-order setting, a compact-stencil Jacobian derived from a lower-order scheme remains an effective preconditioner for the Krylov solver, thereby preserving the efficiency and fast convergence of the JFNK approach. To our knowledge, this work provides the first demonstration of a JFNK solver successfully applied to high-order (beyond second-order) finite-volume solid mechanics formulations.

Another key contribution of this study is its implementation within a conventional open-source framework, which distinguishes it from earlier custom-code prototypes. Specifically, we build our solver using the OpenFOAM-based solids4foam toolbox [31, 32], maintaining its support for arbitrary polyhedral cells by decomposing cells and faces into tetrahedral and triangles. This *conventional* implementation means that the high-order JFNK solver can handle general unstructured meshes and can be directly compared against standard second-order solvers. The implementation is streamlined to be more consistent with the OpenFOAM paradigm, removing the additional boundary unknowns required in [7, 8] and implementing a different treatment of boundary conditions. Finally, we introduction of the α-stabilisation scheme [33–35], which proved to be necessary for stabilising solutions on unstructured (irregular) grids and suppressing high-frequency oscillations or zero-energy modes. The concept of α-stabilisation originates from CFD but is shown here to be naturally compatible with high-order FV discretisations. Importantly, it scales consistently with the order of accuracy, unlike conventional second-order stabilisations (second order Rhie–Chow-type stabilisation [32]) used in segregated solvers. The aim of this paper is therefore twofold: to contribute to the relatively sparse literature on high-order cell-centred finite-volume methods for solids, and to explore the powerful synergy between such methods and Jacobian-free Newton-Krylov solution strategies.

The remainder of this paper is structured as follows: Section 2 presents the governing equations, followed by the formulation of the high-order FV discretisation in Section 3. Section 4 summarises the JFNK solution algorithm. Section 5 reports the performance of the proposed solver across a series of benchmark problems, analysing accuracy, computational cost, robustness and memory requirements. Finally, Section 6 provides concluding remarks and outlines future research directions.

2 Mathematical Model

For arbitrary body of volume Ω bounded by surface Γ with outward facing unit normal $\boldsymbol{n}$ the strong integral form of linear momentum in *total* Lagrangian form is:

$$\int_{\Omega_o} \rho_o \frac{\partial^2 \boldsymbol{u}}{\partial t^2} \mathrm{d}\Omega_o = \oint_{\Gamma_o} \left(\boldsymbol{n}_o \cdot \boldsymbol{P}^T\right) \mathrm{d}\Gamma_o + \int_{\Omega_o} \boldsymbol{f}_b \, \mathrm{d}\Omega_o, \tag{1}$$

where ρ is density, $\boldsymbol{u}$ is the displacement vector, $\boldsymbol{P}$ is the first Piola–Kirchhoff stress tensor, and $\boldsymbol{f}_b$ is a body force per unit volume, e.g., $\rho\boldsymbol{g}$, where $\boldsymbol{g}$ is gravity. Subscript o is used to indicate quantities in the initial reference configuration. Through Nanson's formula it is possible to relate the first Piola–Kirchhoff stress tensor with the Cauchy $\boldsymbol{\sigma}$ stress tensor in the current configuration:

$$\boldsymbol{P} = J\boldsymbol{\sigma}\boldsymbol{F}^{-\mathrm{T}}, \tag{2}$$

where $\boldsymbol{F}$ is deformation gradient defined as $\mathbf{I} + (\boldsymbol{\nabla}_o \boldsymbol{u})^{\mathrm{T}}$ and J is its determinant $J = \det(\boldsymbol{F})$. This work also considers linear geometry formulation, i.e. small strain assumption:

$$\int_{\Omega} \rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} \, \mathrm{d}\Omega = \oint_{\Gamma} \boldsymbol{n} \cdot \boldsymbol{\sigma}_s \, \mathrm{d}\Gamma + \int_{\Omega} \boldsymbol{f}_b \, \mathrm{d}\Omega, \tag{3}$$

where $\boldsymbol{\sigma}_s$ is the engineering (small strain) stress tensor which coincide with Piola stress tensor.

Constitutive relation for Cauchy $\boldsymbol{\sigma}$ stress in Equations 2 and 3 is given by a chosen mechanical law. Mechanical laws considered in this work (Hooke's law, St. Venant-Kirchhoff, Mooney–Rivlin, neo-Hookean, and Guccione) are briefly outlined in Appendix A. The governing equations are complemented by boundary conditions, with three types considered here: prescribed displacement, prescribed traction, and symmetry.

3 High-order Finite Volume Discretisation

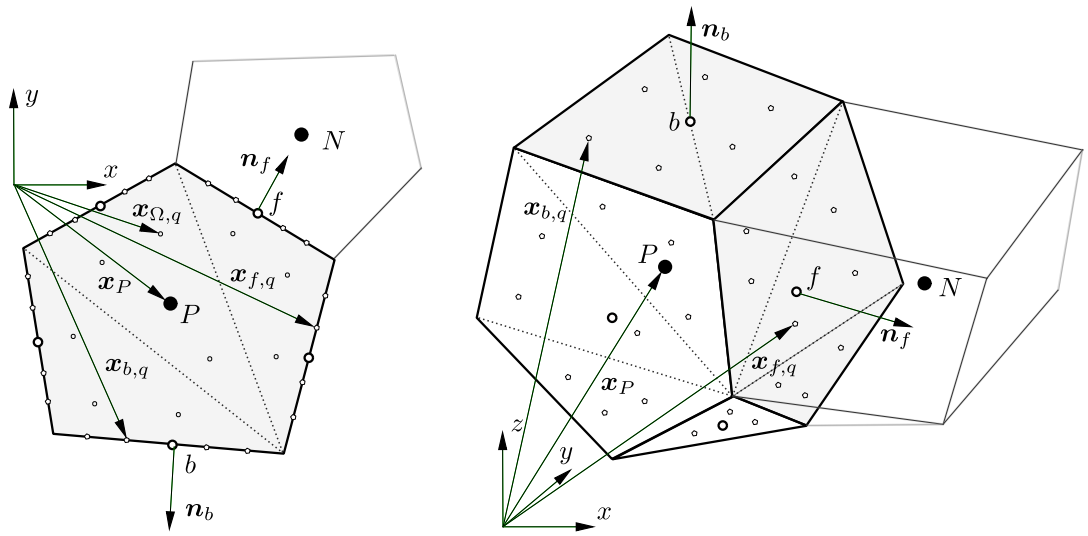
In this work, a finite volume discretization is employed to approximate the strong form of the governing equations. The computational points are located at the cell centroids, where the solution is treated as point-valued rather than cell-averaged. This type of discretization is commonly referred to as the deconvolution finite volume method [2, 36].

3.1 Solution Domain Discretisation

The spatial domain is partitioned into a finite set of contiguous convex polyhedral control volumes, each denoted by P. Representative control volumes in 2D and 3D settings is is shown in Figure 1. The computational node associated with each cell P is positioned at the cell centroid $\boldsymbol{x}_P$, the cell volume is denoted by Ω_P, and the centroid of a neighboring control volume is denoted by N. Each control volume is bounded by a set of polygonal faces, which are categorized as follows:

- Internal faces — shared between adjacent control volumes. The centroid of an internal face is denoted by f, its outward unit normal vector by $\boldsymbol{n}_f$, and its face area by Γ_f.
- Boundary faces — located on the boundary of the spatial domain. The centroid of a boundary face is denoted by b, its outward unit normal vector by $\boldsymbol{n}_b$, and its surface area by Γ_b.

Accurate and robust flux integration is required over these arbitrary polyhedral volumes. To facilitate this, each polygonal face f is subdivided into triangular subfaces, on which quadrature points



(a) Control volume in 2D (b) Control volume in 3D

Fig. 1 Representative convex polyhedral cell P and neighbouring cell N. The positions of the face centre f and cell centre P are evaluated as the arithmetic means of their respective nodal coordinates.

$\boldsymbol{x}_{fg}$ are defined for numerical flux evaluation. The fan triangulation method is adopted for face decomposition to minimise number of integration points. Although the computational mesh is always three-dimensional, two-dimensional problems are treated as planar, where the surface integration reduces to line integration along edges, therefore, no subdivision is required. Each cell volume P is decomposed into tetrahedral (in 3D) or triangular (in 2D) subelements, with volume quadrature points $\mathbf{x}_{\Omega,q}$ defined within each subelement. For all geometric entities, Gaussian quadrature rules are employed. Specifically, Gauss–Legendre quadrature is used for line integrals, Dunavant's symmetric Gaussian quadrature rules [37] are applied for triangular subelements, and Shunn and Ham's symmetric quadrature rules [38] are adopted for volume integration over tetrahedral subelements. A key advantage of this geometric framework is its unified treatment of arbitrary polyhedral topologies, all control volumes are handled consistently using the same reconstruction and integration procedures.

Before proceeding with the discretization of the volume and surface integrals, we introduce the following sets of cell faces. The set of internal faces is denoted by $\mathcal{F}_P^{\text{int}}$, and the set of boundary faces by $\mathcal{F}_P^{\text{bnd}}$. The boundary-face set $\mathcal{F}_P^{\text{bnd}}$ is further partitioned into three mutually disjoint subsets, $\mathcal{F}_P^{bnd} := \mathcal{F}_P^{\text{dip}} \cup \mathcal{F}_P^{\text{sym}} \cup \mathcal{F}_P^{\text{trac}}$, representing boundary faces where displacement ($\mathcal{F}_P^{\text{dip}}$), traction ($\mathcal{F}_P^{\text{trac}}$), or symmetry ($\mathcal{F}_P^{\text{sym}}$) conditions are prescribed. For convenience, we additionally define the set of non-traction boundary faces as $\mathcal{F}_P^{\text{non-trac}} := \mathcal{F}_P^{\text{dip}} \cup \mathcal{F}_P^{\text{sym}}$.

3.2 Surface Integrals

The surface integral term is discretised by integrating over quadrature points of each cell face:

$$\begin{aligned}\oint_{\Gamma_P} \boldsymbol{n}\cdot\boldsymbol{\sigma}_s \,\mathrm{d}\Gamma_P = \sum_{f\,\in\,\mathcal{F}_P}\int_{\Gamma_f} \boldsymbol{n}\cdot\boldsymbol{\sigma}_s \,\mathrm{d}\Gamma_f \approx \sum_{f\,\in\,\mathcal{F}_P^{\text{int}}} \boldsymbol{n}_f\cdot\left[\sum_{q=1}^{q=N_{f,q}} \alpha_q\,(\boldsymbol{\sigma}_s)_{f,q}\right]\Gamma_f \\ + \sum_{b\,\in\,\mathcal{F}_P^{\text{non-trac}}} \boldsymbol{n}_b\cdot\left[\sum_{q=1}^{q=N_{f,q}} \alpha_q\,(\boldsymbol{\sigma}_s)_{b,q}\right]\Gamma_b + \sum_{b\,\in\,\mathcal{F}_P^{\text{trac}}} \boldsymbol{n}_b\cdot\sum_{q=1}^{q=N_{f,q}} \boldsymbol{T}_{b,q}\Gamma_b,\end{aligned} \tag{4}$$

where Γ_P indicates the surface of cell P, vector $\boldsymbol{T}_{b,g}$ represents the prescribed traction at the traction boundary quadrature point, $N_{f,q}$ is the number of face quadrature points and α_q is quadrature weight. Subscript $_{f,q}$ indicates that quantity is calculated at quadrature point location $\boldsymbol{x}_{f,q}$. For non-triangular faces, face quadrature weights are scaled by the ratio of each triangle's area to the total face area, computed as the sum of all sub-triangle areas. Using a consistent definition of face area is essential here, as inconsistency can degrade overall accuracy.

The stress tensor at a quadrature point $\boldsymbol{x}_{f,q}$, denoted $(\boldsymbol{\sigma}_s)_{f,q}$, is computed as a function of the displacement gradient according to the adopted mechanical constitutive law. For the case of engineering (Cauchy) stress, it is given by:

$$\boldsymbol{\sigma}_s(\boldsymbol{x}_{f,q}) = \mu(\nabla \boldsymbol{u})_{f,q} + \mu(\nabla \boldsymbol{u})^T_{f,q} + \lambda \mathrm{tr}((\nabla \boldsymbol{u})_{f,q})\mathbf{I} \tag{5}$$

where μ and λ are the Lamé parameters. Expressions for other constitutive laws are provided in Appendix A. The computation of the displacement gradient $(\nabla \boldsymbol{u})_{f,q}$ is shown in Section 3.5.4.

3.3 Volume Integrals

The volume integral is discretised by integrating over the cell volume quadrature points:

$$\int_{\Omega_P} \boldsymbol{f}_b \mathrm{d}\Omega_P \approx \sum_{q=1}^{q=N_{\boldsymbol{f},q}} \beta_q \ (\boldsymbol{f}_b)_{\Omega,q} \, \Omega_P, \tag{6}$$

where $N_{\boldsymbol{f},q}$ is the number of cell quadrature points and β_q are the corresponding quadrature weights. Similarly, the inertia term (e.g. left-hand side term of Equation (3)):

$$\int_{\Omega_P} \rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} \mathrm{d}\Omega_P \approx \sum_{q=1}^{q=N_{\boldsymbol{u},q}} \gamma_q \ \boldsymbol{\rho}_{\Omega,q} \left(\frac{\partial^2 \boldsymbol{u}}{\partial t^2} \right)_{\Omega,q} \Omega_P, \tag{7}$$

where $N_{\boldsymbol{u},q}$ and γ_q are quadrature points and corresponding weights used for inertia term integration. Subscript $_{\Omega,q}$ is used to denote that quantity is calculated at quadrature point location $\boldsymbol{x}_{\Omega,q}$.

To evaluate these integrals over arbitrary polyhedral cells, the control volume is partitioned into a set of non-overlapping tetrahedra using the face subdivisions already defined for surface integration. Each tetrahedron is formed by connecting the cell centroid P with one of the triangular faces, as illustrated in Figure 2. The cell partition is resolved at the quadrature weight calculation

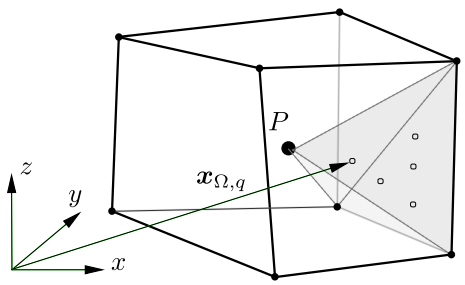


Fig. 2 Volume integration, cell decomposition into tetrahedral elements

level, where the integration weights are scaled by the ratio of its host tetrahedron's volume to the

total cell volume. As in the computation of face quadrature weights, it is crucial to use a consistent cell volume [2].

3.4 Stabilisation Scheme

The evaluation of the surface term (diffusive flux) requires estimates of the solution gradient at the quadrature points $\boldsymbol{x}_{f,q}$ of a face. The most common approach is to average the gradients reconstructed from each side of a quadrature point, either using arithmetic averaging [34, 35] or interpolation with geometrically weighted coefficients [30, 39, 40]. In the present work, the solution gradient is evaluated directly at the quadrature points, eliminating the need for such interpolation. However, as in the aforementioned averaging-based approaches, the discretisation of the diffusive flux may lead to numerical instabilities, manifested as zero-energy (checkerboard-type) modes. To suppress these high-frequency oscillations, an appropriate stabilisation (damping) must be introduced.

The most widely used stabilization technique in the finite-volume framework is the Rhie–Chow type stabilization [41], which is second-order accurate [30]. Since this limits the overall convergence rate to second order, it is not suitable for the present high-order discretisation. Instead, we employ an α-damping scheme [33], which preserves the designed high-order convergence of the method [34, 35]. This damping is introduced in the form of an additional traction term $\boldsymbol{t}_s$, defined as a function of the solution jump between the values extrapolated from the centroids of the two cells sharing a common face:

$$\boldsymbol{t}_s = \alpha \frac{\bar{K}}{|\boldsymbol{d}_{PN} \cdot \boldsymbol{n}_f|} (\boldsymbol{u}^*_{fN} - \boldsymbol{u}^*_{fP}) \tag{8}$$

where α is scale factor and $\bar{K}$ a stiffness-type parameter that gives the stabilisation an appropriate scale and dimension. It is calculated using Lamé parameters $\bar{K} = 2\mu + \lambda$, following previous works [39, 40, 42] where this value is used for Rhie-Chow stabilisation. The vector $\boldsymbol{d}_{PN}$ connects the centroid of cell P with that of its neighboring cell N, while the product $\boldsymbol{d}_{PN} \cdot \boldsymbol{n}_f$ serves as a skewness measure, increasing the damping on highly skewed cells [34]. The quantities $\boldsymbol{u}^*_{fN}$ and $\boldsymbol{u}^*_{fP}$ denote the displacements at the face center f, extrapolated from the cell centers using the derivatives evaluated at N and P, respectively:

$$\begin{aligned} \boldsymbol{u}^*_{fN} &= \boldsymbol{u}_N + \boldsymbol{d}_{Pf} \cdot (\nabla \boldsymbol{u})_N + \frac{1}{2} \boldsymbol{d}^2_{Nf} : (\nabla \nabla \boldsymbol{u})_N + \frac{1}{6} \boldsymbol{d}^2_{Nf} :: (\nabla \nabla \nabla \boldsymbol{u})_N, \\ \boldsymbol{u}^*_{fP} &= \boldsymbol{u}_P + \boldsymbol{d}_{Pf} \cdot (\nabla \boldsymbol{u})_P + \frac{1}{2} \boldsymbol{d}^2_{Pf} : (\nabla \nabla \boldsymbol{u})_P + \frac{1}{6} \boldsymbol{d}^2_{Pf} :: (\nabla \nabla \nabla \boldsymbol{u})_P, \end{aligned} \tag{9}$$

where $\boldsymbol{d}_{Pf} = \boldsymbol{x}_f - \boldsymbol{x}_P$ and $\boldsymbol{d}_{Nf} = \boldsymbol{x}_f - \boldsymbol{x}_N$ represent the vectors from the cell centers to the face center. The second and third derivatives in Equation (9) are employed only for third- and fourth-order discretizations, respectively, and their computation is described in Section 3.5.

The damping contribution is added to the surface integral in Equation (4), for internal faces and boundary faces on which displacement is prescribed:

$$\sum_{f \in \mathcal{F}_P^{\text{int}} \cup \mathcal{F}_P^{\text{disp}}} \boldsymbol{t}_s \Gamma_f = \sum_{f \in \mathcal{F}_P^{\text{int}}} \alpha \frac{\bar{K}}{|\boldsymbol{d}_{PN} \cdot \boldsymbol{n}_f|} (\boldsymbol{u}^*_{fN} - \boldsymbol{u}^*_{fP}) \Gamma_f + \sum_{b \in \mathcal{F}_P^{\text{disp}}} \alpha \frac{\bar{K}}{|\boldsymbol{d}_{Pb} \cdot \boldsymbol{n}_b|} (\boldsymbol{u}_b - \boldsymbol{u}^*_{fP}) \Gamma_b, \tag{10}$$

where $\boldsymbol{u}_b$ is the prescribed displacement and $\boldsymbol{d}_{Pb} = \boldsymbol{x}_b - \boldsymbol{x}_P$. Note that the damping term is integrated using a single quadrature point per face, as there is no requirement for high accuracy in

evaluating the stabilization contribution; it is only important that the added term converges with the desired order.

3.5 High order interpolation scheme

For an arbitrary point in space, denoted by $\tilde{\boldsymbol{x}}$ (for example, a face quadrature point), the displacement field can be reconstructed using a pointwise interpolation of the surrounding cell-centre values:

$$(\boldsymbol{u})_{\tilde{\boldsymbol{x}}} = \sum_{N \in \mathcal{S}} c_N \boldsymbol{u}_N, \tag{11}$$

where c_N are scalar interpolation coefficients associated with the neighbouring cell-centre (computational) points $\boldsymbol{u}_N$, and $\mathcal{S}$ denotes the interpolation stencil, i.e. the set of computational points used in the interpolation, with $|\mathcal{S}|$ being its size. The same interpolation can be used to evaluate the first and higher-order spatial derivatives of the displacement field:

$$\left(\frac{\partial \boldsymbol{u}}{\partial x}\right)_{\tilde{\boldsymbol{x}}} = \sum_{N \in \mathcal{S}} c_{x_N} \boldsymbol{u}_N, \qquad \left(\frac{\partial \boldsymbol{u}}{\partial x}\right)_{\tilde{\boldsymbol{x}}} = \sum_{N \in \mathcal{S}} c_{y_N} \boldsymbol{u}_N, \qquad \left(\frac{\partial \boldsymbol{u}}{\partial z}\right)_{\tilde{\boldsymbol{x}}} = \sum_{N \in \mathcal{S}} c_{z_N} \boldsymbol{u}_N, \tag{12}$$

where c_{x_N}, c_{y_N} and c_{z_N} are the interpolation coefficients corresponding to the spatial derivatives in the $x-$, $y-$, and $z-$directions, respectively. The procedure for computing these interpolation coefficients is described in the following section.

3.5.1 Local Regression Estimators

The Local Regression Estimators (LRE) method is used to determine the interpolation coefficients in Equations (11) and (12). A truncated Taylor expansion of the displacement field, denoted by $\bar{\boldsymbol{u}}$, in the vicinity of $\tilde{\boldsymbol{x}}$ is written as:

$$\begin{aligned} \bar{\boldsymbol{u}}(\boldsymbol{x}) = (\boldsymbol{u})_{\tilde{\boldsymbol{x}}} &+ \left(\frac{\partial \boldsymbol{u}}{\partial x}\right)_{\tilde{\boldsymbol{x}}} (x - \tilde{x}) + \left(\frac{\partial \boldsymbol{u}}{\partial y}\right)_{\tilde{\boldsymbol{x}}} (y - \tilde{y}) + \left(\frac{\partial \boldsymbol{u}}{\partial z}\right)_{\tilde{\boldsymbol{x}}} (z - \tilde{z}) + \frac{1}{2}\left(\frac{\partial^2 \boldsymbol{u}}{\partial x^2}\right)_{\tilde{\boldsymbol{x}}} (x - \tilde{x})^2 \\ &+ \frac{1}{2}\left(\frac{\partial^2 \boldsymbol{u}}{\partial x \partial y}\right)_{\tilde{\boldsymbol{x}}} (x - \tilde{x})(y - \tilde{y}) + \frac{1}{2}\left(\frac{\partial^2 \boldsymbol{u}}{\partial x \partial z}\right)_{\tilde{\boldsymbol{x}}} (x - \tilde{x})(z - \tilde{z}) + \dots \end{aligned} \tag{13}$$

The expansion is truncated at polynomial order p, which corresponds to N_t terms, where $N_t = \frac{(d+p)!}{(d!\,p!)}$ in d spatial dimensions. The truncated expansion can then be written compactly using a polynomial basis $\mathbf{q}$ and a vector of unknown coefficients $\bar{\mathbf{a}}$ representing the derivatives:

$$\bar{\boldsymbol{u}} = \mathbf{q}^T(\boldsymbol{x} - \tilde{\boldsymbol{x}})\bar{\mathbf{a}}(\tilde{\boldsymbol{x}}), \tag{14}$$

where

$$\begin{aligned} \mathbf{q}^T(\boldsymbol{x} - \tilde{\boldsymbol{x}}) &= \left[1, (x - \tilde{x}), (y - \tilde{y}), (z - \tilde{z}), \frac{1}{2}(x - \tilde{x})^2, \dots\right], \\ \bar{\mathbf{a}}(\tilde{\boldsymbol{x}}) &= \left[(\boldsymbol{u})_{\tilde{\boldsymbol{x}}}, \left(\frac{\partial \boldsymbol{u}}{\partial x}\right)_{\tilde{\boldsymbol{x}}}, \left(\frac{\partial \boldsymbol{u}}{\partial y}\right)_{\tilde{\boldsymbol{x}}}, \left(\frac{\partial \boldsymbol{u}}{\partial z}\right)_{\tilde{\boldsymbol{x}}}, \left(\frac{\partial^2 \boldsymbol{u}}{\partial x^2}\right)_{\tilde{\boldsymbol{x}}}, \dots\right]. \end{aligned} \tag{15}$$

The LRE method estimates the parameter vector $\bar{\mathbf{a}}$ by minimizing the weighted sum of squared differences between the Taylor approximation and the cell-centre values:

$$\mathcal{R} = \frac{1}{2} \sum_{N \in \mathcal{S}} w_N \left[\bar{\boldsymbol{u}}(\boldsymbol{x}_N) - \boldsymbol{u}_N \right]^2 = \sum_{N \in \mathcal{S}} w_N \left[\mathbf{q}^T(\boldsymbol{x} - \tilde{\boldsymbol{x}}) \bar{\mathbf{a}}(\tilde{\boldsymbol{x}}) - \boldsymbol{u}_N \right]^2 \tag{16}$$

Minimizing Equation (16) with respect to $\bar{\mathbf{a}}(\tilde{\boldsymbol{x}})$ leads to the following normal equations:

$$\bar{\mathbf{M}}(\tilde{\boldsymbol{x}}) \bar{\mathbf{a}}(\tilde{\boldsymbol{x}}) = \bar{\mathbf{Q}}(\tilde{\boldsymbol{x}}) \mathbf{W}(\tilde{\boldsymbol{x}}) \mathbf{u}, \tag{17}$$

where $\mathbf{M}(\tilde{\boldsymbol{x}}) = \mathbf{Q}(\tilde{\boldsymbol{x}}) \mathbf{W}(\tilde{\boldsymbol{x}}) \mathbf{Q}^T(\tilde{\boldsymbol{x}})$, $\mathbf{Q}(\tilde{\boldsymbol{x}})$ is the $N_t \times |\mathcal{S}|$ matrix whose $N-$th column is $\mathbf{q}^T(\boldsymbol{x} - \tilde{\boldsymbol{x}})$ and $\mathbf{W}(\tilde{\boldsymbol{x}})$ is a diagonal weighting matrix of size $|\mathcal{S}| \times |\mathcal{S}|$. Defining $\mathbf{A}(\tilde{\boldsymbol{x}}) = \mathbf{M}(\tilde{\boldsymbol{x}})^{-1} \mathbf{Q}(\tilde{\boldsymbol{x}}) \mathbf{W}(\tilde{\boldsymbol{x}})$, the system becomes:

$$\bar{\mathbf{a}}(\tilde{\boldsymbol{x}}) = \bar{\mathbf{A}}(\tilde{\boldsymbol{x}}) \mathbf{u}, \tag{18}$$

The rows of $\bar{\mathbf{A}}$ correspond to the interpolation coefficients used in Eqs. (11) and (12). The interpolation coefficients from matrix $\bar{\mathbf{A}}$ depend only on the mesh (geometry), so they can be computed once (or whenever the mesh moves) and stored. The inversion of $\bar{\mathbf{M}}(\tilde{\boldsymbol{x}})$ may be ill-conditioned, for poor spatial distribution of stencil points. This issue is mitigated by scaling the Taylor basis, ensuring sufficient stencil size, restricting the weights in $\mathbf{W}$ to positive values, and employing a QR factorization via Householder transformations [43], see [7, 15].

The weight function in Equation (16) is radially symmetric exponential function:

$$w_N = \frac{e^{-\tilde{d}^2 k^2} - e^{-k^2}}{1 - e^{-k^2}}, \tag{19}$$

where $d = |\tilde{\boldsymbol{x}} - \boldsymbol{x}_N|$, $d_s = 2r_s$ is the stencil diameter, $\tilde{d} = d/d_s$ and r_s is the stencil radius, defined as the maximum distance $|\tilde{\boldsymbol{x}} - \boldsymbol{x}_N|$ in stencil $\mathcal{S}$. The weighting function in Equation (19) has been previously used in the context of the Navier–Stokes equations [14, 44] and solid mechanics problems [7, 8, 15]. Following these studies, the shape parameter $k = 6$ is adopted herein for both 2D and 3D cases, as it was shown to provide optimal performance [7, 8, 15].

3.5.2 Interpolation Stencil

The minimum stencil size $|\mathcal{S}|_{\min}$ is determined by the dimension of the polynomial basis and the interpolation order, i.e. $|\mathcal{S}|_{\min} = N_t$:

$$\begin{aligned} &\text{for 2D:} \quad |\mathcal{S}|_{\min} = \frac{(p+1)(p+2)}{2}, \\ &\text{for 3D:} \quad |\mathcal{S}|_{\min} = \frac{(p+1)(p+2)(p+3)}{6}. \end{aligned} \tag{20}$$

The actual stencil size $|S|$ used for computations is set as:

$$|\mathcal{S}| = |\mathcal{S}|_{\min} + n^+ \tag{21}$$

where n^+ is a user-defined number of surplus points. The choice of n^+ represents a trade-off: an excessively large stencil introduces numerical dissipation and increases computational cost, whereas a small stencil may lead to unstable interpolation [44]. Based on those findings presented in [7, 8],

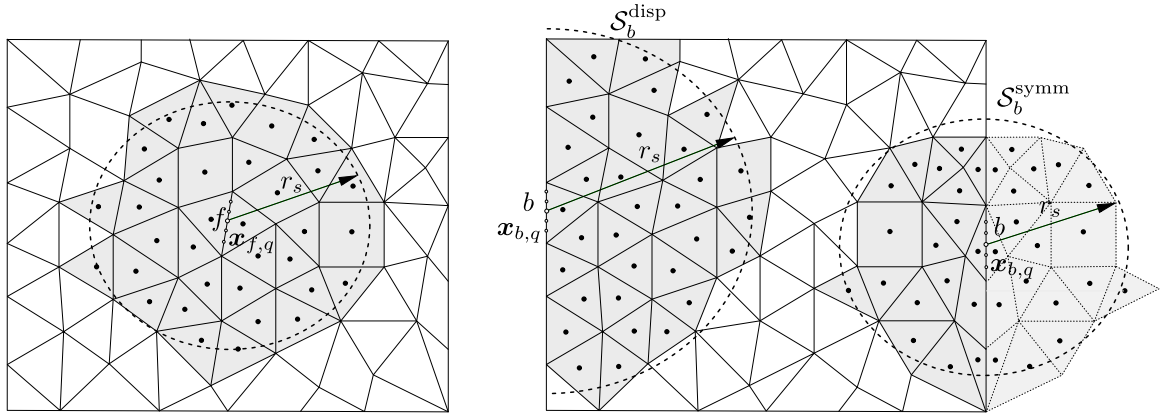
n^+ is set to 10 for all two-dimensional cases, and to 45, 55, and 65 for $p = 1$, $p = 2$, and $p = 3$ in three-dimensional cases, respectively. These values are consistent with those reported in the literature. In particular, [45] noted that within the $k-$exact least-squares framework, the stencil size is typically chosen in the range $1.5\,|\mathcal{S}|_{\min} \leq |\mathcal{S}| \leq 3\,|\mathcal{S}|_{\min}$.

The stencil is constructed using a nearest-neighbour approach, where the set $\mathcal{S}$ comprises the cells closest to the reference point (face or cell centre), enclosed within a sphere of radius r_s. To reduce computational cost, all quadrature points on a given face share the same stencil, and likewise, all quadrature points within a cell use a common stencil. The construction of the internal-face stencil $\mathcal{S}_f$ is illustrated in Figure 3(a).

Stencil construction is adapted for boundary faces according to the imposed boundary condition (see Figure 3(b)):

- Traction boundary – no stencil is defined, since interpolation is not applied.
- Displacement boundary – the stencil, denoted $\mathcal{S}_b^{\text{disp}}$, is constructed in the same manner as for internal faces.
- Symmetry boundary – the stencil, denoted $\mathcal{S}_b^{\text{symm}} := \mathcal{S}_b^{\text{symm,p}} \cup \mathcal{S}_b^{\text{symm,g}}$, is divided equally: half of the points correspond to the physical domain $\mathcal{S}_b^{\text{symm,p}}$, while the remaining half are mirrored ghost points $\mathcal{S}_b^{\text{symm,g}}$ that replicate both the position and state variables.

Alternative stencil construction strategies that account for the local cell distribution may also be employed [44, 45]; however, they poses greater implementation complexity and in some cases higher computational cost. Chosen approach represents a efficient and robust choice for solid mechanics applications, particularly for problems involving moving meshes.



(a) Stencil $\mathcal{S}_f$ for internal face f quadrature points $\boldsymbol{x}_{f,q}$

(b) Stencil for boundary face b quadrature points $\boldsymbol{x}_{b,q}$ in the case of displacement $\mathcal{S}_b^{\text{disp}}$ and symmetry $\mathcal{S}_b^{\text{symm}}$ boundary type.

Fig. 3 Interpolation stencils for internal f and boundary b face quadrature points, $\boldsymbol{x}_{f,q}$ and $\boldsymbol{x}_{b,q}$, respectively.

3.5.3 Boundary Conditions

Boundary conditions are imposed through modifications of the local polynomial reconstruction, except for traction boundaries, where the known tractions at the quadrature points are directly integrated and added to the surface term in Equation (4).

At displacement boundaries, the prescribed displacement is enforced in a weak sense by introducing a penalty contribution into the polynomial reconstruction. This is achieved by extending the stencil of each boundary quadrature point with an additional ghost point, collocated with the

quadrature point itself and assigned the prescribed displacement value $\mathbf{u}_{b,q}$. The inclusion of this ghost point effectively augments the reconstruction matrices $\mathbf{W}$ and $\mathbf{Q}$. Since the Taylor expansion in Equation (13) is centred at the same location, the associated polynomial basis vector simplifies to $\mathbf{q}^{\mathrm{T}} = [1,\, 0,\, 0,\, \ldots,\, 0]$, and its weight in $\mathbf{W}$ is typically set to unity. Increasing this weight enhances the local satisfaction of the prescribed displacement but does not affect the overall convergence rate.

At symmetry boundaries, the entire interior stencil is reflected across the symmetry plane. The polynomial basis vectors for these mirrored points are computed using their reflected coordinates. This approach guarantees enforcement of symmetry constraints; zero tangential component of boundary traction $(\mathbf{I}-2\boldsymbol{n}_b\otimes\boldsymbol{n}_b)\cdot\boldsymbol{T}_{b,q}=0$, and zero normal component of displacement $\boldsymbol{n}_b\cdot\boldsymbol{u}_b=0$.

Achieving high-order convergence near boundaries requires careful attention to geometric representation. When curved surfaces are approximated by planar faces, the resulting geometric discretisation is second-order accurate, which inherently limits the overall convergence rate of the method. High-order accuracy can be recovered either by introducing isoparametric boundary representations or by maintaining flat faces while introducing some sort of correction [**? ?**]. Curved boundary treatment has not yet been tackled in literature (for finite volume solid mechanics) and is therefore left for future work, as it presents a challenging problem deserving dedicated study.

3.5.4 Computing Gradients

For all quadrature points on internal faces, Equation (18) is evaluated, and the interpolation coefficients obtained from the matrix $\bar{\mathbf{A}}$ are stored for use in computing the displacement gradient:

$$(\nabla \boldsymbol{u})_{f,q} = \sum_{N \,\in\, \mathcal{S}_f} \boldsymbol{c}_{\boldsymbol{x}_N} \otimes \boldsymbol{u}_N, \tag{22}$$

where $\boldsymbol{c}_{\boldsymbol{x}_N}^T = [\,c_{x_N},\, c_{y_N},\, c_{z_N}]$ is interpolation vector (see Equation (12)). The same procedure is applied at displacement boundaries using the stencil $\mathcal{S}_b^{\mathrm{disp}}$ extended by ghost point:

$$(\nabla \boldsymbol{u})_{b,q} = \left(\sum_{N \,\in\, \mathcal{S}_b^{\mathrm{disp}}} \boldsymbol{c}_{\boldsymbol{x}_N} \otimes \boldsymbol{u}_N \right) + \boldsymbol{c}_{\boldsymbol{b}_N} \otimes \boldsymbol{u}_{b,q}, \tag{23}$$

where $\boldsymbol{u}_{b,q}$ is prescribed displacement at boundary quadrature point. Coefficient vector $\boldsymbol{c}_{\boldsymbol{b}_N}$ is extracted from corresponding location from the same matrix $\bar{\mathbf{A}}$ as for $\boldsymbol{c}_{\boldsymbol{x}_N}$ coefficients. For symmetry boundary, the contribution of the mirrored (ghost) points is also included, giving:

$$(\nabla \boldsymbol{u})_{b,q} = \sum_{N \,\in\, \mathcal{S}_b^{\mathrm{symm,p}}} \boldsymbol{c}_{\boldsymbol{x}_N} \otimes \boldsymbol{u}_N + \sum_{N \,\in\, \mathcal{S}_b^{\mathrm{symm,g}}} \boldsymbol{c}_{\boldsymbol{x}_N} \otimes (\boldsymbol{R}_b \cdot \boldsymbol{u}_N), \tag{24}$$

where $\boldsymbol{R}_b = (\mathbf{I} - 2\boldsymbol{n}_b \otimes \boldsymbol{n}_b)$ is reflection tensor [46]. Note that only first derivatives are stored at face quadrature points, as higher derivatives are not required by the constitutive equations. In the present solution procedure, the displacements $\boldsymbol{u}_N$ are known, making the gradient evaluation straightforward. However, if $\boldsymbol{u}_N$ are treated as unknowns, the interpolation coefficients must be incorporated into the left-hand-side system matrix, as shown in [7, 8] and Appendix B.

The stabilization scheme requires the evaluation of first- and higher-order derivatives at cell centres when computing Equation (9). For this purpose, each cell is associated with a stencil $\mathcal{S}_P$,

constructed in the same manner as the face stencil. At each cell centre, the gradient is computed according to Equation (22), while the Hessian is evaluated for $p = 2$, and the third-order derivative tensor for $p = 3$. The formulation and computation of the Hessian and the third-order derivative tensor are described in Appendix C.

4 Solution Algorithm

The nonlinear system arising from the finite volume discretization is solved using the Jacobian-free Newton–Krylov (JFNK) algorithm. The subsequent subsection describes the numerical formulation and implementation aspects of this approach.

4.1 Jacobian-free Newton-Krylov Algorithm

The linear momentum balance (Equation (1) and (3)) can be expressed as:

$$\boldsymbol{R}(\boldsymbol{u}) = 0, \tag{25}$$

where $\boldsymbol{u}$ is primary unknown field (displacement) and $\boldsymbol{R}$ represents the *residual* (imbalance) of the equation. For the implicit discretisation of momentum balance, residual vector is the function of the unknown displacement field at iteration $k + 1$:

$$\boldsymbol{R}(\boldsymbol{u}_{k+1}) = 0, \tag{26}$$

The residual can be approximated using first-order Taylor expansion about current point k:

$$\boldsymbol{R}(\boldsymbol{u}_{k+1}) \approx \boldsymbol{R}(\boldsymbol{u}_k) \; + \; \boldsymbol{R}'(\boldsymbol{u}_k)(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k), \tag{27}$$

where $\boldsymbol{R}'$ is the Jacobian matrix $\boldsymbol{J} \equiv \boldsymbol{R}' \equiv \partial \boldsymbol{R}/\partial \boldsymbol{u}$. Imposing $\boldsymbol{R}(\boldsymbol{u}_{k+1}) = \boldsymbol{0}$ yields the standard Newton update equation:

$$\boldsymbol{J}(\boldsymbol{u}_k)\delta\boldsymbol{u} = -\boldsymbol{R}(\boldsymbol{u}_k) \tag{28}$$

where $\delta\boldsymbol{u} = \boldsymbol{u}_{k+1} - \boldsymbol{u}_k$. In the Jacobian-free Newton–Krylov approach, the Jacobian matrix is not formed explicitly. Instead, a Krylov subspace method (e.g., GMRES) is used to solve the linear system iteratively, requiring only the action of the Jacobian on a arbitrary vector $\boldsymbol{v}$:

$$\boldsymbol{J}\boldsymbol{v} \approx \frac{\boldsymbol{R}(\boldsymbol{u} + \epsilon\boldsymbol{v}) - \boldsymbol{R}(\boldsymbol{u})}{\epsilon} \tag{29}$$

where ϵ is a small scalar perturbation. To mitigate the effect of ill-conditioned Jacobian matrices on convergence and robustness, a right preconditioning strategy is employed:

$$\boldsymbol{J}\boldsymbol{P}^{-1}\boldsymbol{v} \approx \frac{\boldsymbol{R}(\boldsymbol{u} + \epsilon\boldsymbol{P}^{-1}\boldsymbol{v}) - \boldsymbol{R}(\boldsymbol{u})}{\epsilon} \tag{30}$$

where $\boldsymbol{P}$ denotes the preconditioning matrix. As the preconditioner, we adopt a compact-stencil approximate Jacobian, the one that is used in the second-order cell-centre semi-implicit discretisation with segregated solution procedure [40, 47]. Following the classification of Knoll and Keyes [26], this preconditioning choice may be considered "physics-based".

The approximate Jacobian, for cell P, corresponds to the discretised diffusion term using a central-difference scheme:

$$\boldsymbol{P}_P = \oint_{\Gamma_P} \bar{K}\, \boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u}\, \mathrm{d}\Gamma_P \approx \sum_{f \in \mathcal{F}_P^{\text{int}}} \bar{K}\, |\boldsymbol{\Delta}_f| \, \frac{\boldsymbol{u}_N - \boldsymbol{u}_P}{|\boldsymbol{d}_{PN}|} \Gamma_f + \sum_{b \in \mathcal{F}_P^{\text{disp}}} \bar{K}\, |\boldsymbol{\Delta}_b| \, \frac{\bar{\boldsymbol{u}}_b - \boldsymbol{u}_P}{|\boldsymbol{d}_{Pb}|} \, |\boldsymbol{\Gamma}_b| + \sum_{b \in \mathcal{F}_P^{\text{symm}}} \bar{K}\, |\boldsymbol{\Delta}_b| \, \frac{\boldsymbol{R}_b \cdot \boldsymbol{u}_P - \boldsymbol{u}_P}{|\boldsymbol{d}_{Pb}|} \Gamma_b, \tag{31}$$

where $\boldsymbol{\Delta}_f = \boldsymbol{d_{PN}}/(\boldsymbol{d}_{PN} \cdot \boldsymbol{n}_f)$ is *over-relaxed orthogonal* vector [48]. Note that the non-orthogonal correction, which is normally required to preserve accuracy when discretising the diffusion term on skewed meshes, is omitted here. Including it would enlarge the stencil (molecule) and compromise the compactness achieved with the central-difference approximation. As will be shown later, this simplification does not degrade robustness or accuracy, since the overall accuracy of the discretisation is governed primarily by the residual evaluation itself.

For the JFNK solver, the PETSc library (version 3.22.2) [49] is used, where the interfaces for evaluating the approximate Jacobian (matrix $\boldsymbol{P}$) and the high-order residual ($\boldsymbol{R}(\boldsymbol{u})$) are implemented within the `solids4foam` toolbox [32, 40, 47], which is built on OpenFOAM-v2312 [31]. An extended discussion with additional implementation details is provided in the authors' previous work [30], where the JFNK solver is coupled with a second-order residual evaluation.

5 Test Cases

The performance of the proposed high-order solver is verified using several benchmark cases. These are employed to assess both the solution accuracy and the formal order of convergence, and to provide a direct comparison against a standard second-order discretisation. The comparison is evaluated in terms of solution accuracy, computational cost, and memory requirements. In addition, the influence of the stabilisation factor, different choices of preconditioning matrices, and other numerical aspects of the method are examined.

The benchmark cases are carefully selected to cover a broad range of features, including both two- and three-dimensional problems, small- and large-strain regimes, and various material models. The convergence behaviour is studied on a fine-grained sequence of mesh refinements to detect potential oscillatory convergence trends, which may arise in high-order finite-volume formulations [35].

In all simulations, the convergence criterion corresponds to a three-order-of-magnitude reduction in the residual, and a line-search procedure is employed to stabilise the solution update [30]. The number of quadrature points per tetrahedral, triangular, or line element is kept to the minimum required for exact integration of the stress distribution. For example, triangular elements with $p = 1$ and $p = 2$ reconstructions require one integration point, whereas $p = 3$ uses three. The interpolation stencil size for all cases is used as specified in Section 3.5.2. Different mesh types are considered; the open-source mesher Gmsh [50] was employed to generate tetrahedral meshes. For the structured hexahedral meshes, the native OpenFOAM utilities `blockMesh` and `extrudeMesh` were used. Polyhedral meshes were subsequently produced by converting a representative tetrahedral mesh using the `polyDualMesh` utility, which is also available in OpenFOAM.

5.1 Testing the Accuracy and Order of Accuracy

This section presents all benchmark cases considered in the paper. The accuracy and convergence behaviour of the method are evaluated for interpolation orders $p = 1$, $p = 2$, and $p = 3$. For cases where an analytical solution is available, the solution error is quantified using the L_2 and L_∞ norms, where the L_2 norm represents the root mean square error:

$$L_2 = \frac{1}{N_\mathrm{c}} \sum_{i=1}^{N_\mathrm{c}} |\Delta\phi_i|, \tag{32}$$

and L_∞ is infinity norm representing the maximum absolute error:

$$L_\mathrm{inf} = \max_{1 \leq i \leq N_\mathrm{c}} |\Delta\phi_i|, \tag{33}$$

where $\Delta\phi_i$ is the difference between expected and predicted solutions at computational point and N_c is the overall number of computational points, i.e. cells. Both norms are calculated for displacement magnitude $|\boldsymbol{u}| = \sqrt{u_x^2 + u_y^2 + u_z^2}$ and von Mises stress:

$$\sigma_{ekv} = \sqrt{\frac{1}{2}\left[(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2\right] + 3\left(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2\right)}. \tag{34}$$

5.1.1 Case 1: Order Verification via the Manufactured Solution Procedure

The first test case consists of a 0.2×0.2 m square or $0.2 \times 0.2 \times 0.2$ m cube with linear elastic ($E = 200$ GPa, $\nu = 0.3$) properties. A manufactured solution for displacement (Figure **??**) is employed in the form or polynomial, trigonometric [51] or mixed form [7]:

- **2D**

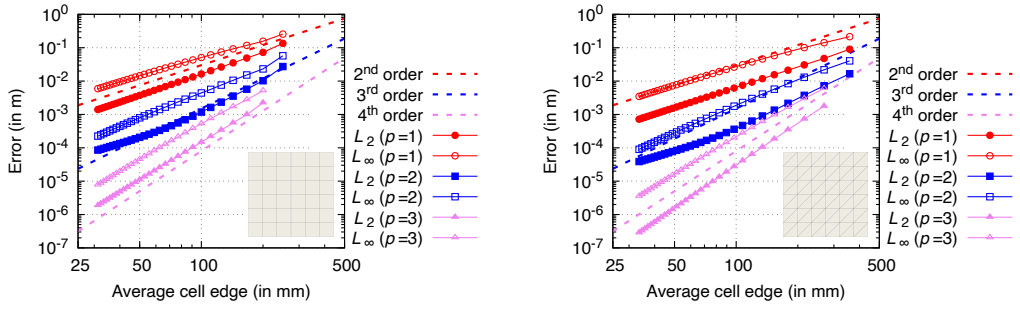
$$\boldsymbol{u} = \begin{pmatrix} e^{x^2}\sin(y) \\ \ln(3+y)\cos(x) + \sin(y) \\ 0. \end{pmatrix} \tag{35}$$

- **3D**

$$\boldsymbol{u} = \begin{pmatrix} a_x \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ a_y \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ a_z \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \end{pmatrix} \tag{36}$$

  where $a_x = 2\,\mu$m, $a_y = 4\,\mu$m, and $a_z = 6\,\mu$m.

The Cartesian coordinates are given by x, y and z. The corresponding manufactured body force term ($\boldsymbol{f}_b$ in Equation 3) can be found in [30] or obtained by manufactured solution procedure. velicine mreza



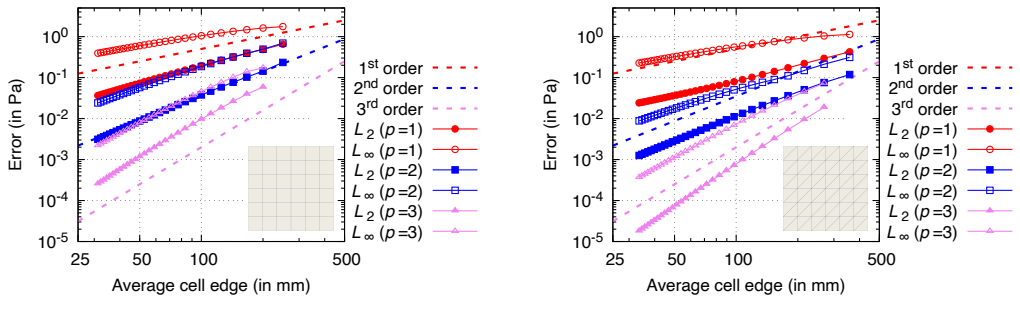
(a) Hexahedral mesh

(b) Tetrahedral structured mesh

(c) Tetrahedral unstructured mesh

(d) Polyhedral structured mesh

Fig. 4 Manufactured solution cube (2D case): error convergence for displacement magnitude.



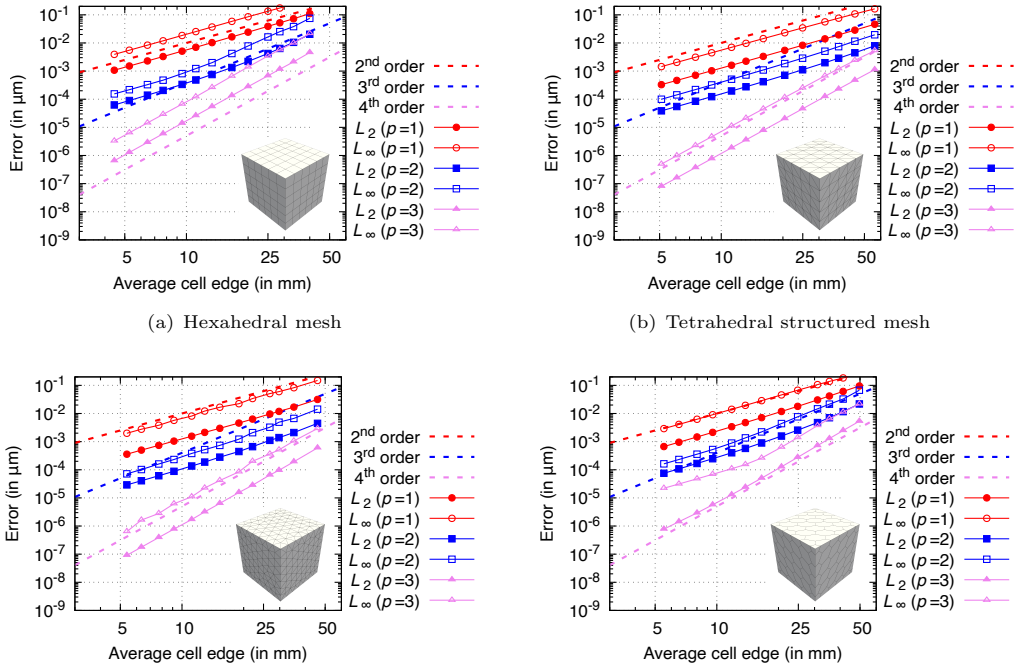
(a) Hexahedral mesh

(b) Tetrahedral structured mesh

(c) Tetrahedral unstructured mesh

(d) Polyhedral structured mesh

Fig. 5 Manufactured solution cube (2D case): error convergence for stress magnitude.



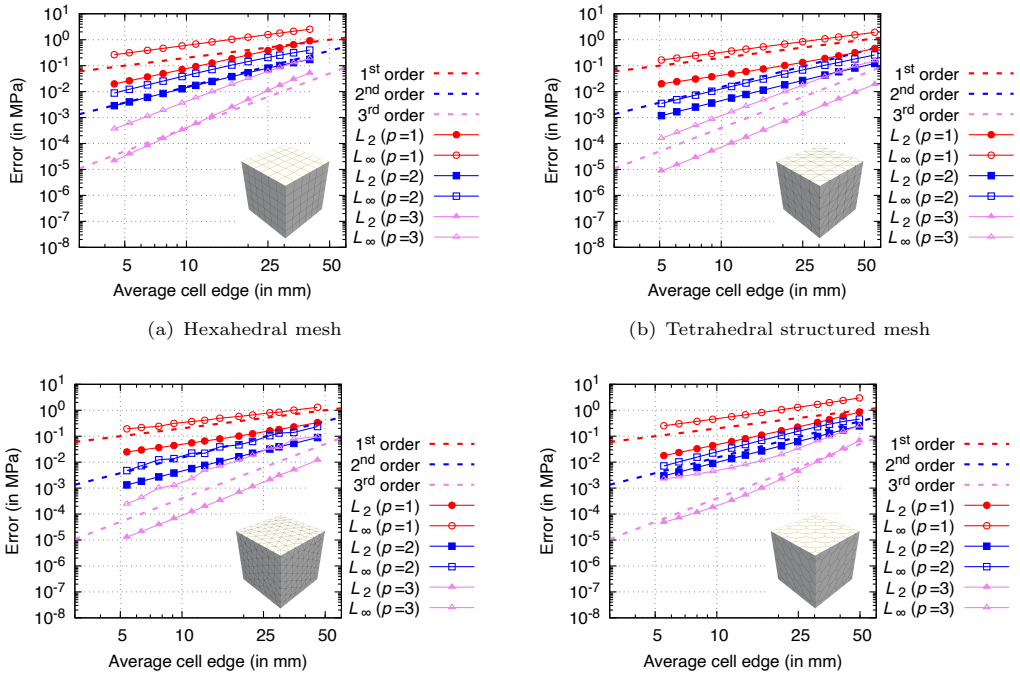
(a) Hexahedral mesh

(b) Tetrahedral structured mesh

(c) Tetrahedral unstructured mesh

(d) Polyhedral structured mesh

Fig. 6 Manufactured solution cube (3D case): error convergence for displacement magnitude.



(a) Hexahedral mesh

(b) Tetrahedral structured mesh

(c) Tetrahedral unstructured mesh

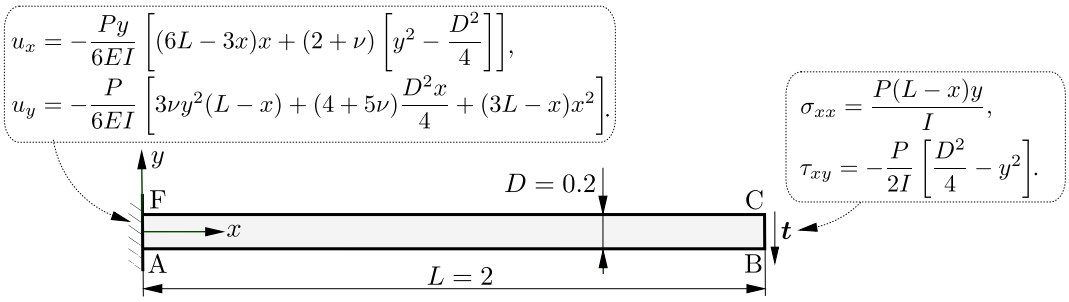
(d) Polyhedral structured mesh

Fig. 7 Manufactured solution cube (3D case): error convergence for stress magnitude.

5.1.2 Case 2: Cantilever Beam

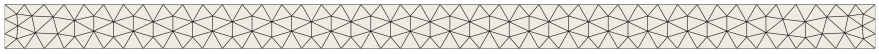
The test case geometry, shown in Fig. 6(a), is a rectangular beam with dimensions of $L \times D = 2$ m $\times$ 0.2 m, a Young's modulus of $E = 200$ GPa, and a Poisson's ratio of $\nu = 0.3$. The top and bottom

boundaries of the beam are traction-free, and plane strain conditions are assumed. The left end of the beam is constrained by the analytical displacement, while the right end is subjected to the corresponding analytical traction of (0, 1) MPa. This setup enables quantification of the difference between the predicted displacement and the analytical solution as a measure of convergence across the entire domain, rather than only at the beam end. Convergence is assessed using 15 successively refined grids.velicine mreza



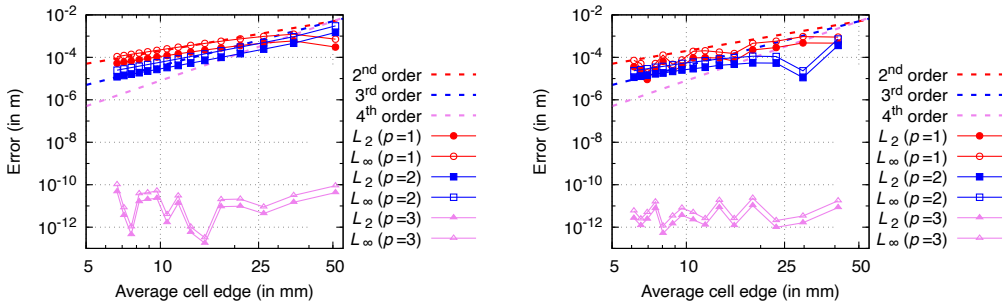
(a) Case geometry and boundary conditions

(b) Tetrahedral unstructured mesh



(c) Tetrahedral unstructured mesh

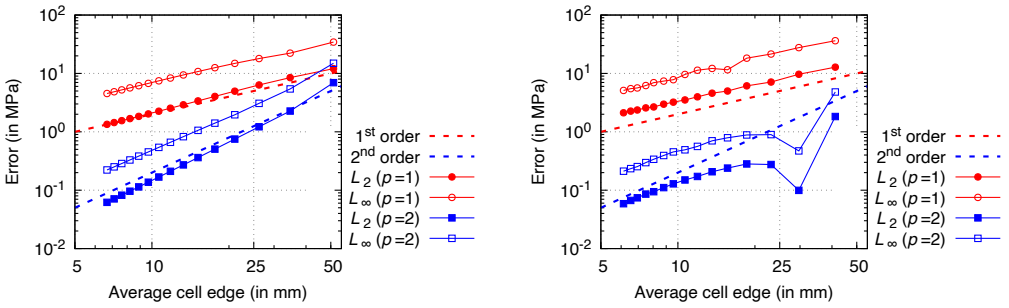
Fig. 8 Cantilever beam case: case geometry, boundary conditions and the coarsest mesh used.



(a) Tetrahedral structured mesh

(b) Tetrahedral unstructured mesh

Fig. 9 Cantilever beam case: displacement magnitude discretisation errors.



(a) Tetrahedral structured mesh (b) Tetrahedral unstructured mesh

Fig. 10 Cantilever beam case: error convergence for stress magnitude. Results for $p = 3$ are omitted for brevity, i.e. stress error for $p = 3$ is below $\sim 10^{-6}$ MPa.

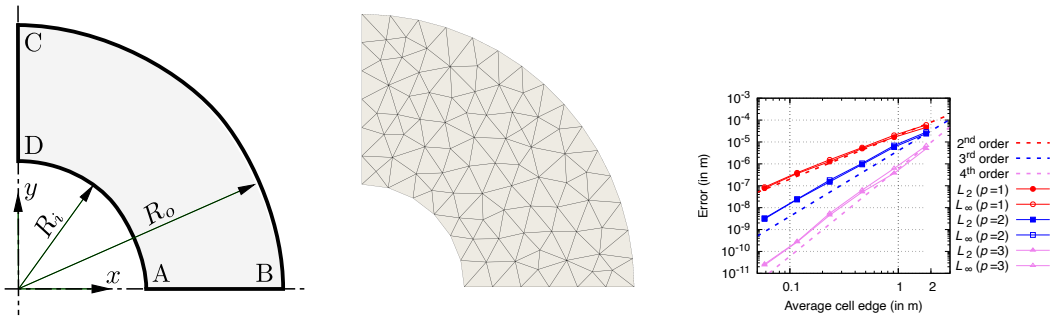
5.1.3 Case 3: Internally Pressurised Thick-walled Cylinder

In this case, a homogeneous thick-wall cylindrical pressure vessel with an inner radius $R_i = 7$ m, outer radius $R_o = 18.625$ m, and loaded internally with pressure $p = 100$ MPa is analysed. Two types of material are considered:

i. Small strain, linear-elastic [52]: $E = 10$ GPa, $\nu = 0.3$.

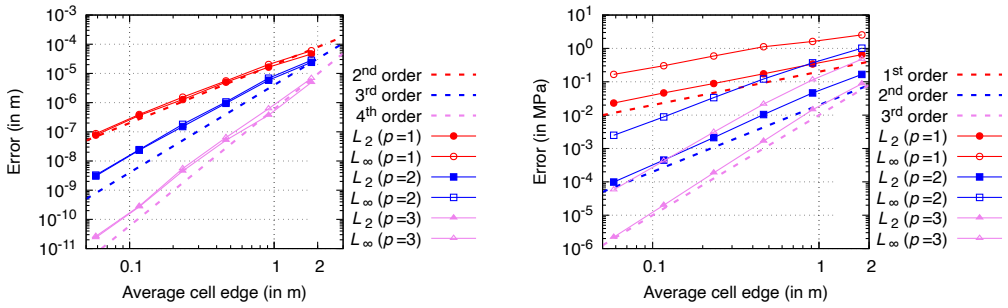
ii. Finite strain, Mooney-Rivlin law [53]: $c_{10} = 80$ MPa, $c_{01} = 20$ MPa, and $c_{11} = 0.0$ MPa and Poisson's ratio $\nu = 0.49$

The problem is considered plane stress, with the 2-D computational domain comprising a quarter of the cylinder geometry. The cylinder is discretised with series of unstructured tetrahedral grids. Gravitational and inertial effects are neglected. Linear elastic case is solved using one loading increment while hyperelastic case is solved using 100 equal loading increments. Analytical solutions are available in [54] and [55].velicine mreza



(a) Case geometry (b) Tetrahedral unstructured mesh (c) Displacement convergence

Fig. 11 Pressurised cylinder case: geometry and the coarsest mesh.



(a) Displacement convergence (b) Stress convergence

Fig. 12 Pressurised cylinder case: displacement and stress magnitude discretisation errors for linear elastic material.

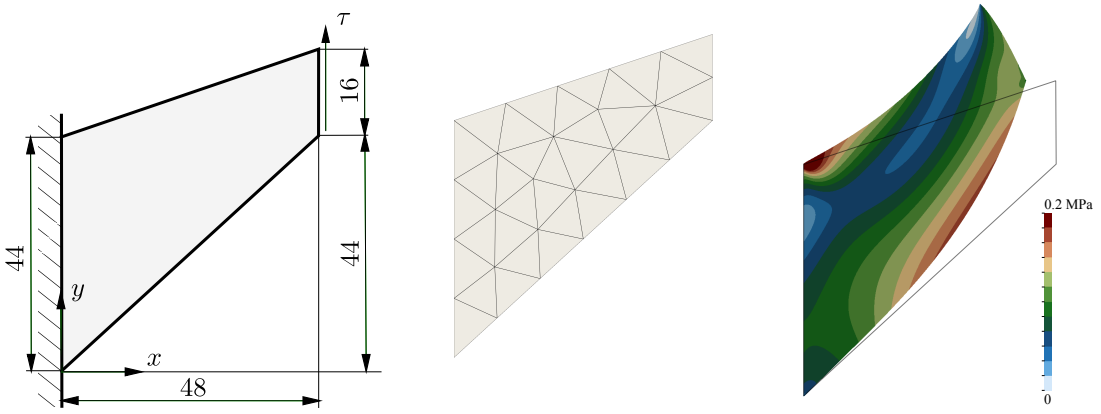
5.1.4 Case 4: Cook's Membrane

The Cook's membrane configuration (Figure 13) is a classic benchmark problem dominated by bending effects. It consists of a two-dimensional trapezoidal panel (plane-strain condition) that is fully constrained along its left edge and loaded by a uniform shear traction applied to the right edge. In this study, two variants of the problem are analysed:

i. Small-strain, linear elastic [56, 57]: $E = 70$ MPa, $\nu = 1/3$, and $\tau = 6.25$ kPa.
ii. Finite-strain, neo-Hookean hyperelastic [58]: $E = 1.0985$ MPa, $\nu = 0.3$, and $\tau = 62.5$ kPa.

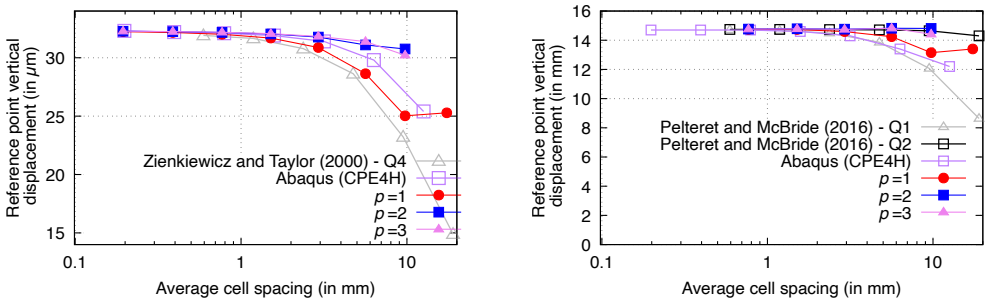
where E denotes Young's modulus, ν the Poisson ratio, and τ is the imposed shear traction. The analysis is performed as quasi-static without body forces. The linear case is solved in a single step, whereas the hyperelastic case is solved using 30 uniform load increments. To validate the results, Figure 14 plots the vertical displacement of a key reference point against the average cell width, comparing the predictions to literature data and Abaqus (C3D8 element) results. The location of this reference point differs by analysis type: it is the top-right corner (48, 60) mm for the elastic case but the midpoint of the loaded edge (48, 52) mm for the hyperelastic case. Eight successively refined tetrahedral meshes are used 11, 35 (Figure 15(b)), 164, 177, 263, 389, 1457, 5637, 22119 and 88012 cells.

Firs order extrapolation for figure 13(c)



(a) Case geometry and dimensions (in mm) (b) Tetrahedral unstructured mesh (c) Equivalent (von Mises) stress distribution for hyperelastic case

Fig. 13 Cook's membrane case: geometry, mesh and deformation in the hyperelastic case.

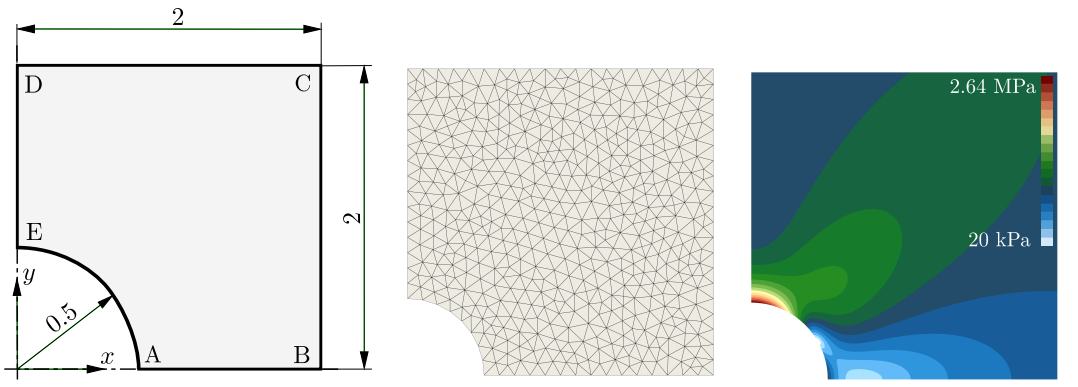


(a) Linear elastic case (b) Hyperelastic case

Fig. 14 Cook's membrane vertical displacement predictions at the reference point as a function of the average cell width. Comparisons are given with the results from the literature [56–58]. Case with $p = 1$ is using $n+ = 7$ instead of 10 get resutls on first mesh.

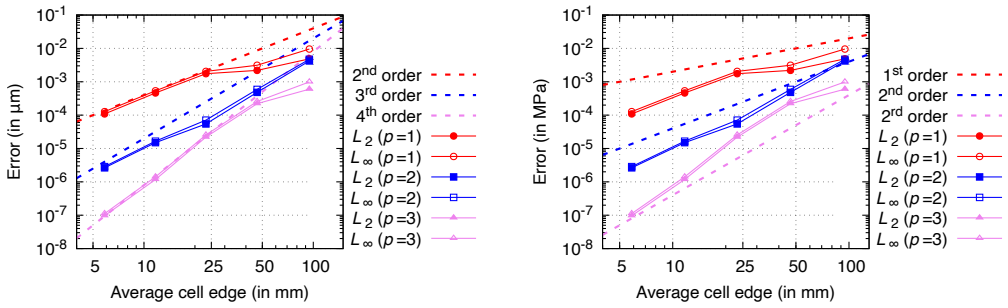
5.1.5 Case 4: Hole in a Infinite Plate Subjected to Remote Stress

This benchmark problem consists of a thin, infinitely large plate with a circular hole, subjected to uniaxial tension. Owing to the symmetry of the geometry and loading, only one quarter of the plate is modeled as a finite domain. To minimize the influence of the finite computational boundaries, the exact tractions obtained from the analytical solution [59] are prescribed on the outer edges BC and CD. Symmetry boundary conditions are applied on boundaries AB and DE, while zero traction is specified on the hole boundary. The material properties are defined by a Young's modulus of $E = 200\,\text{GPa}$ and a Poisson's ratio of $\nu = 0.3$. Five successively refined tetrahedral meshes are used 978 (Figure 15(b)), 4050, 16296, 64257 and 256512 cells.



(a) Case geometry (b) Tetrahedral unstructured mesh (c) Von Mises stress distribution

Fig. 15 Plate hole case: geometry, mesh and resulting stress distribution

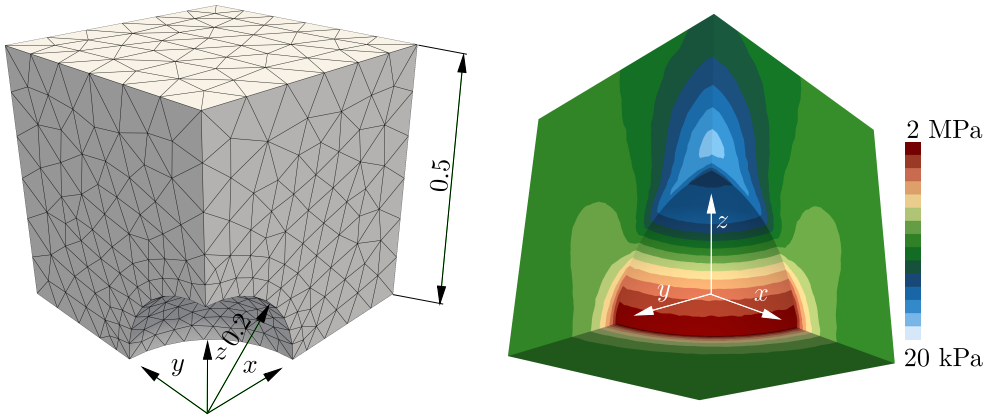


(a) Hexahedral mesh (b) Tetrahedral mesh

Fig. 16 Plate hole case: displacement and stress error convergence

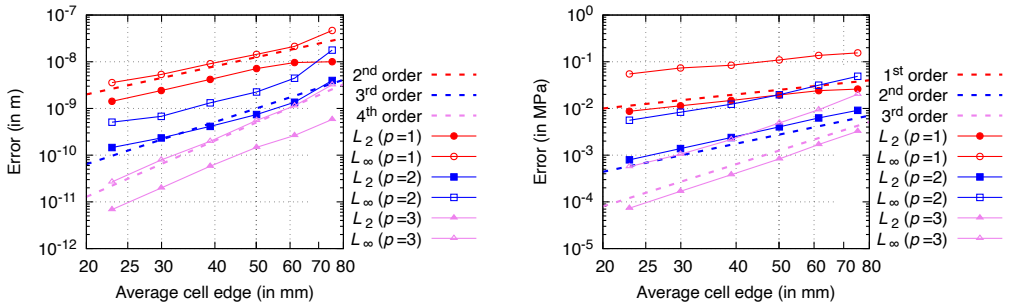
5.1.6 Case 5: Spherical Cavity in an Infinite Solid Subjected to Remote Stress

This problem considers a spherical cavity embedded within an infinite and isotropic medium with linear elastic material properties ($E = 200$ GPa, $\nu = 0.3$). The solid is subjected to a far-field uniaxial tensile stress of $\sigma_{zz} = T = 1$ MPa, with all other stress components being zero. A key feature of this problem is the pronounced stress concentration that forms near the cavity on the plane perpendicular to the applied load; this stress rapidly diminishes further from the cavity. Although the problem is inherently axisymmetric, it is solved here as a 3-D case with a mesh graded towards the cavity. The computational domain is taken as one-eighth of a $1 \times 1 \times 1$ m cube aligned with the Cartesian axes, with one corner at the centre of the sphere. The analytical tractions are applied to the domain's far boundaries to mitigate the finite geometry effects. The analytical expressions for stress [60] and displacement [61] can be found in [30]. The analysis is performed using six different unstructured tetrahedral meshes with cell counts of 3802, 7030, 14891, 32864, 72945 and 158870, with the coarsest mesh shown in Figure 18(b).



(a) Unstructured tetrahedral mesh (b) Von Mises stress distribution

Fig. 17 Spherical cavity case: geometry, mesh and resulting von Mises stress distribution.



(a) Displacement convergence

(b) Stress convergence

Fig. 18 Spherical cavity case: displacement and stress error convergence.

5.1.7 Case 6: Inflation of an Idealised Ventricle

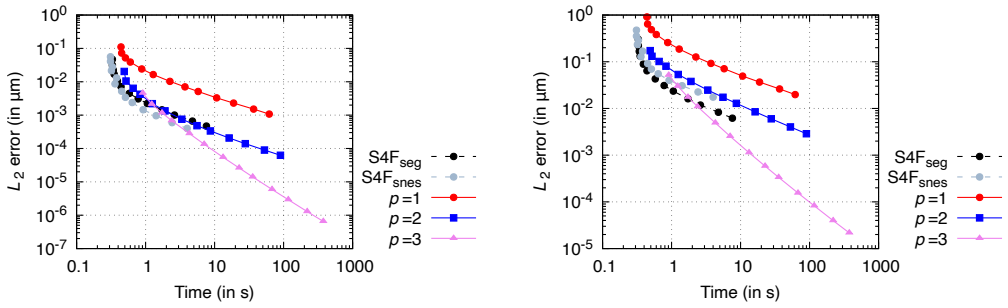
This case considers the idealized ventricle inflation benchmark developed by Land et al. [62] to test cardiac mechanics software. The material is governed by the hyperelastic, transversely isotropic constitutive law from Guccione et al. [63]. The specific parameters used, $C = 10$ kPa and $c_f = c_t = c_{fs} = 1$, are chosen to produce an isotropic material response. The benchmark specifies incompressibility; this is enforced in the current work in a weak sense by setting the bulk modulus ($\kappa = 1000$ kPa) to be two orders of magnitude greater than the shear modulus parameter C. The ventricle geometry is defined as a truncated ellipsoid (see Cardiff et al. [30], Land et al. [62] for geometry definition). The base plane is fixed, the internal surface (endocardial surface) has an internal pressure of 10 kPa, while the outer surface is traction-free. The problem is solved quasi-statically using 100 equal increments on a set of four grids: 1 620, 12 960, 50 000, and 103 680 cells.

5.2 Error-Cost Relationship

[**?**] cell centred grad is second order, stranica 10.

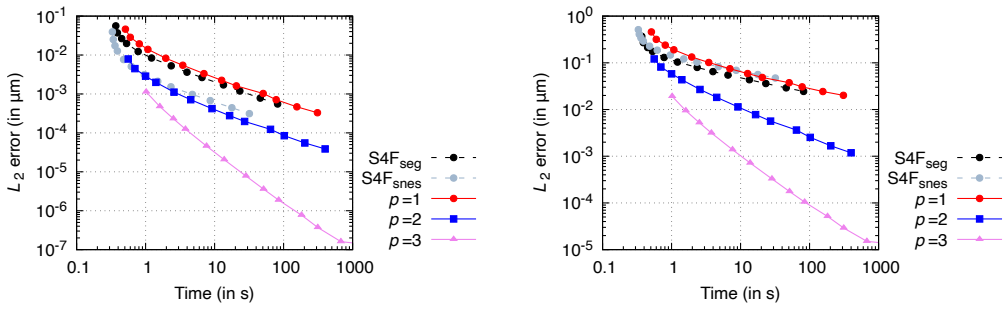
Clock times and memory usage (measured with the GNU time utility) were generated using a Mac Studio with an M1 Ultra CPU on one core, where the code was built with the Clang compiler (version 16.0.0). Examination of multi-CPU-core parallelisation is left to Section

An additional clarification regarding the notation used in the next sections is needed. The abbreviations are as follows: "$S4F^{seg}$" stands for solids4foam segregated solver whereas "$S4F^{snes}$" is the same solver with JFNK algorithm. Both solvers are second-order accurate, meaning that their accuracy is comparable with high-order discretisation with p=1. they reach the same accuracy under different timings and memory requirements MMS case is one iteration + volume source term. Hex from S4F have good timings becouse it is showing "overconvergence" on such regular grids



(a) Displacement error (b) Stress error

Fig. 19 Error-time relationship for 3D MMS case, hexahedral structured mesh



(a) Displacement error (b) Stress error

Fig. 20 Error-time relationship for 3D MMS case, tetrahedral structured mesh

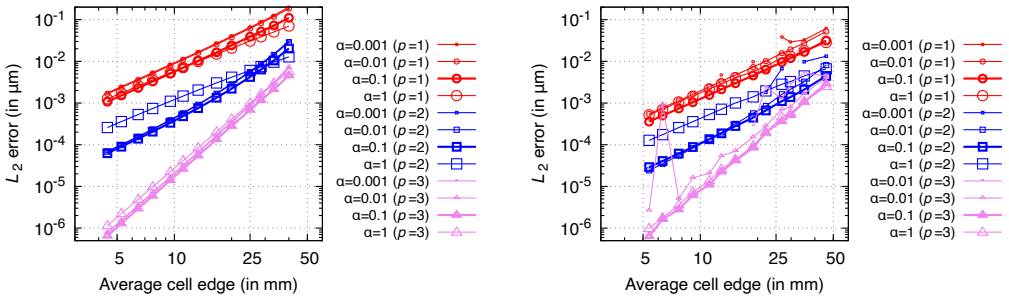
5.3 Stabilisation Scheme Effects

This section analyses the effect of the scaling factor, α, which is a user-defined parameter in the stabilisation scheme (Equation (8)) used to suppress spurious zero-energy modes. While $\alpha = 0.1$ is used as the default value for results in all examples, the investigation here focuses on how a different choice of α impacts solution accuracy and computational timing. The effect on timing is indirect: while the damping calculation itself has a constant computational cost, different α values alter the properties of the system matrix by lowering the matrix condition number, making it easier to precondition. This change in system behavior directly affects the linear solver's performance. Accordingly, selecting α involves a balance between introducing numerical dissipation (which impacts accuracy) and maintaining solver efficiency and robustness. To quantify this trade-off, the Method of Manufactured Solutions case (case 5.1.1) is used with structured hexahedral and unstructured tetrahedral meshes.

Four values for the scaling factor, each differing by an order of magnitude, are investigated: $\alpha = 0.001, 0.01, 0.1$, and 1.0. Figure E1 presents the L_2 error norm convergence for displacement, while Figure 22 presents computational timings. As expected, different values of α do not affect the convergence order; the exception is for $p = 2$ (hexahedral mesh), where $\alpha = 1$ forces an earlier transition to second-order. In terms of robustness, $\alpha = 0.001$ is too low to stabilise the unstructured (irregular) mesh, and solver converges only on some meshes. From these diagrams, one can see that $\alpha = 0.1$ and $\alpha = 0.01$ are the best values in terms of accuracy. However, when looking at Figure 22, it is clear that the recommended value of $\alpha = 0.1$ is the optimal choice for efficiency. It

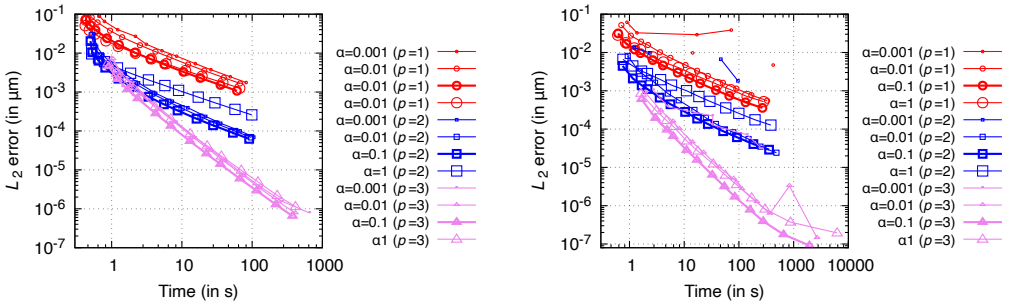
should be emphasised that the recommended value, $\alpha = 0.1$, is the same as the one for Rhie-Chow stabilisation in standard second-order solvers [30].

As an alternative to the proposed α-stabilisation, a comparable stabilising effect can be achieved by increasing the interpolation stencil and employing more integration points than the minimum required—an approach often referred to as *over-integration* or *antialiasing*. This strategy has been adopted in previous studies [7, 8]. However, its main drawbacks are the higher computational cost, arising from both the enlarged stencil and the increased number of integration points. Moreover, while this approach can be effective for structured meshes, its impact is significantly reduced for unstructured meshes, where the present α-stabilisation scheme demonstrates clear advantages (see Appendix E).



(a) Structured hexahedral mesh (b) Unstructured tetrahedral mesh

Fig. 21 Convergence of the L_2 displacement error norm for case 5.1.1 with varying stabilisation parameter: $\alpha = 0.001, 0.01, 0.1$, and 1.0.



(a) Structured hexahedral mesh (b) Unstructured tetrahedral mesh

Fig. 22 L_2 displacement error norm versus computational time for case 5.1.1, comparing the effect of $\alpha = 0.001, 0.01, 0.1$, and 1.0.

5.4 Effect of Poor Interpolation Conditioning

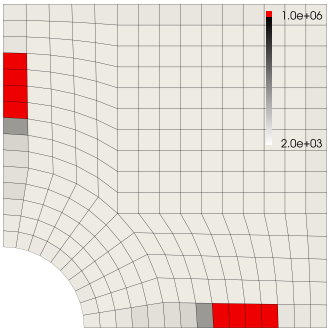
The conditioning of the least-squares system used to compute interpolation coefficients can influence both the accuracy and robustness of the numerical solution. When the interpolation matrix $\bar{\mathbf{M}}$ is highly ill-conditioned, the resulting coefficients may lead to an *artificially stiff discretisation*, which can cause the linear solver to stall or converge very slowly. In this context, *artificially stiff discretisation* means that the residual vector $\boldsymbol{R}(\boldsymbol{u})$ becomes highly sensitive to displacements of some neighbouring cells within the interpolation stencil.

This problem is particularly relevant in high-order schemes, which are generally more suitable for unstructured meshes, since irregular point distributions tend to improve matrix conditioning [**?**

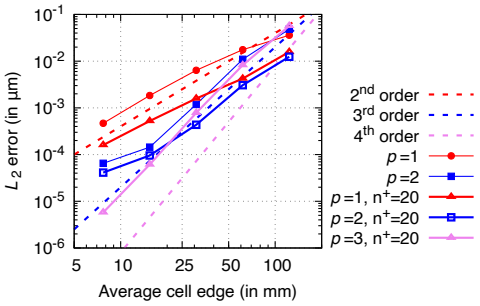
? ?]. On structured grids, however, the regular alignment of points amplifies ill-conditioning (especially in the case of stretched structured grids), leading to numerical difficulties. This limitation is often regarded as one of the key challenges of high-order finite-volume discretisations.

The objective of this section is to illustrate the impact of this phenomenon and to demonstrate a simple remedy to mitigate it. Numerical tests reveal that ill-conditioning is most pronounced in structured two-dimensional meshes, where the absence of a third dimension prevents natural improvement of the condition number. To investigate this effect, the plate hole benchmark from Section 5.1.5 is considered, discretised with hexahedral control volumes (see Figure 23(a)).

Figure 23(b) presents the L_2-error convergence of the displacement field for two values of the stencil-expansion parameter: $n^+ = 10$, the default used in all previous examples, and $n^+ = 20$. Increasing n^+ effectively enlarges the interpolation stencil and reduces the condition number of the least-squares matrix $\bar{\mathbf{M}}$, leading to improved solver robustness. As shown, with $n^+ = 10$ it is not possible to obtain results for $p = 3$ because the linear solver fails to converge, whereas for $n^+ = 20$ convergence is achieved, albeit with a reduction in the accuracy level. It should be noted that in the case of $p = 3$ and $n^+ = 10$, where the JFNK linear solver stalls, a fully implicit discretisation combined with a direct linear solver (as in [7, 8]) can still produce a solution; however, the accuracy of these results is typically affected.



(a) Hexahedral structured mesh, coloured with matrix $\bar{\mathbf{M}}$ condition number. Red coloured cells have condition number $> 1e12$



(b) Displacement L_2 norm convergence for $n^+ = 10$ and $n^+ = 20$

Fig. 23 Plate hole case: effects of high ill-conditioning of matrix $\bar{\mathbf{M}}$.

5.5 Different Choice for Approximate Jacobian

It has already been demonstrated that a compact-stencil *approximate* Jacobian, when used as a preconditioner in the JFNK algorithm, is highly effective for second-order finite-volume discretisations in solid mechanics [30]. In the present work, we confirm that this preconditioning strategy remains effective even when the residual is evaluated using a higher-order discretisation.

To isolate the impact of preconditioner accuracy, a linear elastic test case is employed where the *true* (exact) Jacobian matrix can be assembled explicitly for reference (see Appendix B). Although in a truly linear problem, Newton's method would converge in a single iteration with the exact Jacobian, we deliberately enforce multiple Newton iterations using the assembled *true* Jacobian in this case. This allows a direct comparison of convergence behaviour and computational cost against the standard compact-stencil approximate Jacobian preconditioner. The results show that using a fully assembled true Jacobian as the preconditioner provides no improvement in solver convergence or performance, but dramatically increases the computational time. For example,

in the spherical cavity benchmark (Table 1), employing the *true* Jacobian preconditioner can make the solve from 2 to 92 times slower than using the compact-stencil approximation. These findings make it clear that increasing the Jacobian's accuracy yields no practical benefits; on the contrary, it severely penalises efficiency. This outcome strongly supports continuing to use the compact-stencil *approximate* Jacobian for preconditioning, given its superior efficiency and significant memory savings with no loss of solver effectiveness. Finally, this is consistent with the documented behavior of the *approximate* Jacobians used in high-order finite volume computational fluid dynamics [20–23, 25–29].

	$p = 1$		$p = 2$		$p = 3$	
Mesh size	AJ	TJ	AJ	TJ	AJ	TJ
2068	1.34	2.82	1.78	4.18	4.55	418.33
3802	2.19	5.48	2.89	9.46	8.02	24.07
7030	3.94	11.43	5.23	16.82	14.80	48.75
14891	8.15	30.89	10.96	44.00	32.00	91.12
32864	18.40	90.45	24.95	1896.12	75.64	246.67
72945	43.48	1183.30	58.71	3841.79	189.71	1340.57

Table 1 Spherical cavity case: computational time (in s) for different preconditioning matrices, comparing the approximate Jacobian (AJ) with the high-order true Jacobian (TJ).

5.6 Code parallelisation

To work on after meeting with Philip

6 Conclusions

This work presents a high-order cell-centred finite volume formulation for solid mechanics combined with a Jacobian-free Newton–Krylov nonlinear solution strategy. Third- and fourth-order spatial accuracy is achieved through Gaussian quadrature and a least-squares (LRE) gradient reconstruction procedure. To ensure stability on unstructured meshes, an α-stabilisation scheme is introduced to suppress high-frequency error modes. Solver efficiency is maintained through the use of a compact-stencil second-order approximate Jacobian as a preconditioner within the JFNK solver. The formulation does not require additional boundary unknowns, supports arbitrary polyhedral cells and is demonstrated across a range of linear and nonlinear benchmark problems.
The principal findings of the study are:

- **Accuracy**: The proposed discretisation delivers consistent high-order accuracy across diverse test cases. Linear reconstruction yields second-order displacement and first-order stress accuracy. Quadratic reconstruction achieves second-order accuracy for both displacement and stress; the expected third-order displacement behaviour is not observed, consistent with prior reports in the literature. Cubic reconstruction attains fourth-order displacement and third-order stress accuracy, confirming the effectiveness of the underlying high-order flux integration and LRE reconstruction.
- **Stabilisation scheme**: The introduced α-stabilisation scheme, adapted from high-order CFD, effectively damps spurious high-frequency errors on unstructured meshes. The stabilisation term scales with the order of the method, unlike the Rhie–Chow correction used in second-order finite volume solvers. An optimal value of $\alpha \approx 0.1$ is observed, which is also consistent with recommendations for Rhie-Chow stabilisation [30].

- **Preconditioning choice**: The compact-stencil approximate Jacobian, originating from a second-order semi-implicit discretisation, remains highly effective when used as a preconditioner for high-order residual evaluations. Comparison against a fully assembled true Jacobian (possible for linear elastic tests) shows that increased preconditioner fidelity does not improve convergence, confirming that compact approximate Jacobians offer the best balance of efficiency, robustness, and memory usage.
- **Avoidance of shear locking**: The high-order finite volume formulation naturally avoids shear-locking artefacts, providing an important advantage over conventional second-order discretisations.
- **Robustness**: The method is robust across linear, nonlinear, structured, and unstructured meshes. The main challenge arises from the ill-conditioning of the local least-squares reconstruction, particularly on structured 2-D grids. When conditioning is improved (e.g., via enlarged stencils), robustness is restored.
- **Efficiency**: The OpenFOAM implementation enabled direct comparison with conventional second-order finite volume solvers. Tetrahedral meshes provide the best performance between accuracy and computational cost due to fewer quadrature points. Efficiency decreases with increasing face-edge count because this enlarges the quadrature stencil and increases storage requirements for interpolation coefficients. Despite a non-optimised implementation, the method shows competitive performance and substantial potential.
- **Integration into existing JFNK frameworks**: The high-order formulation requires only modifications to the residual evaluation and therefore integrates seamlessly into existing second-order JFNK infrastructure. The complete implementation is released within `solids4foam`, enabling community use, evaluation, and extension.

Future work will be directed in several key areas. First, the treatment of curved boundaries in the context of high-order finite volume for solid mechanics is a critical and unresolved challenge that warrants special attention. Second, more efficient high-order implementations, potentially using economical approach from [2], should be investigated to maximise computational efficiency. Finally, the successful application of this high-order JFNK algorithm to solid mechanics opens a promising avenue for its extension to a monolithic, high-order, fluid-solid interaction framework.

Data Availability. The codes presented are publicly available at https://github.com/solids4foam/solids4foam, and the cases and plotting scripts are available at https://github.com/solids4foam/solid-benchmarks.

Declaration of generative AI and AI-assisted technologies in the writing process. During the preparation of this work, the authors used ChatGPT and Grammarly as writing assistants. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Acknowledgments. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 101088740). Financial support is gratefully acknowledged from the Irish Research Council through the Laureate programme, grant number IRCLA/2017/45, from Bekaert through the University Technology Centre (UTC phases I and II) at UCD (www.ucd.ie/bekaert), from I-Form, funded by Science Foundation Ireland (SFI) Grant Numbers 16/RC/3872 and 21/RC/10295_P2, co-funded under European Regional Development Fund and by I-Form industry partners, and from NexSys, funded by SFI Grant Number 21/SPP/3756. Additionally, the authors

wish to acknowledge the DJEI/DES/SFI/HEA Irish Centre for High-End Computing (ICHEC) for the provision of computational facilities and support (www.ichec.ie), and part of this work has been carried out using the UCD ResearchIT Sonic cluster which was funded by UCD IT Services and the UCD Research Office.

Appendix A Mechanical Laws

A.1 Linear Elasticity

The definition of engineering stress $\boldsymbol{\sigma}_s$ for linear elasticity can be given as

$$\begin{aligned}\boldsymbol{\sigma}_s &= 2\mu\boldsymbol{\varepsilon} + \lambda \operatorname{tr}(\boldsymbol{\varepsilon})\mathbf{I} \\ &= \mu\boldsymbol{\nabla u} + \mu(\boldsymbol{\nabla u})^{\mathrm{T}} + \lambda(\boldsymbol{\nabla}\cdot\boldsymbol{u})\mathbf{I}\end{aligned} \tag{A1}$$

where λ is the first Lamé parameter, and μ is the second Lamé parameter, synonymous with the shear modulus. The Lamé parameters can be expressed in term of the Young's modulus (E) and Poisson's ratio ν as

$$\mu = \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{A2}$$

A.2 St. Venant-Kirchoff Hyperelasticity

The St. Venant-Kirchoff model defines the second Piola–Kirchhoff stress $\mathbf{S}$ as

$$\boldsymbol{S} = 2\mu\boldsymbol{E} + \lambda \operatorname{tr}(\boldsymbol{E})\mathbf{I} \tag{A3}$$

where, as before, λ is the first Lamé parameter, and μ is the second Lamé parameter. The Lagrangian Green strain $\mathbf{E}$ is defined as

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{\nabla u} + \boldsymbol{\nabla u}^{\mathrm{T}} + \boldsymbol{\nabla u}\cdot\boldsymbol{\nabla u}^{\mathrm{T}}\right) \tag{A4}$$

The true stress can be calculated from the second Piola–Kirchhoff stress as

$$\boldsymbol{\sigma} = \frac{1}{J}\boldsymbol{F}\cdot\boldsymbol{S}\cdot\boldsymbol{F}^{\mathrm{T}} \tag{A5}$$

A.3 Neo-Hookean Hyperelasticity

The definition of true (Cauchy) stress $\boldsymbol{\sigma}$ for neo-Hookean hyperelasticity can be given as

$$\boldsymbol{\sigma} = \frac{\mu}{J}\operatorname{dev}\left(\bar{\boldsymbol{b}}\right) + \frac{\kappa}{2}\frac{J^2-1}{J}\mathbf{I} \tag{A6}$$

where, once again, μ is the shear modulus, and κ is the bulk modulus. The bulk modulus can be expressed in terms of the Young's modulus (E) and Poisson's ratio ν as

$$\kappa = \frac{E}{3(1-2\nu)} \tag{A7}$$

The volume-preserving component of the elastic left Cauchy–Green deformation tensor is $\boldsymbol{b}$ is given as

$$\bar{\boldsymbol{b}} = J^{-2/3}\boldsymbol{b} = J^{-2/3}\boldsymbol{F}\cdot\boldsymbol{F}^{\mathrm{T}} \tag{A8}$$

In the limit of small deformations $\|\nabla \mathbf{u}\| \ll 1$, neo-Hookean hyperelasticity (Equation A6) reduces to linear elasticity (Equation A1).

A.4 Guccione Hyperelasticity

The Guccione et al. [63] hyperelastic law defines the second Piola-Kirchhoff stress as

$$\boldsymbol{S} = \frac{\partial Q}{\partial \boldsymbol{E}} \left(\frac{C}{2} \right) e^{Q} + \frac{\kappa}{2} \frac{J^2 - 1}{J} \mathbf{I} \tag{A9}$$

where

$$Q(I_1, I_2, I_4, I_5) = c_t I_1^2 - 2c_t I_2 + (c_f - 2c_{fs} + c_t) I_4^2 + 2(c_{fs} - c_t) I_5 \tag{A10}$$

$$\begin{aligned} \frac{\partial Q}{\partial \boldsymbol{E}} &= 2c_t \boldsymbol{E} + 2(c_f - 2c_{fs} + c_t) I_4 (\boldsymbol{f_0} \otimes \boldsymbol{f_0}) \\ &\quad + 2(c_{fs} - c_t) \left[\boldsymbol{E} \cdot (\boldsymbol{f_0} \otimes \boldsymbol{f_0}) + (\boldsymbol{f_0} \otimes \boldsymbol{f_0}) \cdot \boldsymbol{E} \right] \end{aligned} \tag{A11}$$

The scalars C, c_f, c_{fs}, and c_t are material parameters and invariants of the Green strain, $\boldsymbol{E} = \boldsymbol{F}^{\mathrm{T}} \cdot \boldsymbol{F}$, are defined as

$$\begin{aligned} I_1 = \mathrm{tr}(\boldsymbol{E}), \quad I_2 = \frac{1}{2} \left[\mathrm{tr}^2(\boldsymbol{E}) - \mathrm{tr}(\boldsymbol{E} \cdot \boldsymbol{E}) \right], \\ I_4 = \boldsymbol{E} : (\boldsymbol{f_0} \otimes \boldsymbol{f_0}), \quad I_5 = (\boldsymbol{E} \cdot \boldsymbol{E}) : (\boldsymbol{f_0} \otimes \boldsymbol{f_0}) \end{aligned} \tag{A12}$$

with $\boldsymbol{f_0}$ representing the unit fibre directions in the initial configuration.

Equation A5 is used to convert the second Piola-Kirchhoff stress to the true stress.

A.5 Mon-Rivlin - add here if used

Appendix B Preconditioner Based on True Jacobian

$$\begin{aligned} &\sum_{f \in \mathcal{F}_P^{\text{int}}} \boldsymbol{n}_f \cdot \left[\sum_{q=1}^{q=N_{f,q}} \alpha_q \, (\boldsymbol{\sigma}_s)_{f,q} \right] \Gamma_f = \\ &= \sum_{f \in \mathcal{F}_P^{\text{int}}} \boldsymbol{n}_f \cdot \left[\sum_{q=1}^{q=N_{f,q}} \alpha_q \, \left(\mu (\nabla \boldsymbol{u})_{f,q} + \mu (\nabla \boldsymbol{u})_{f,q}^T + \lambda \mathrm{tr}((\nabla \boldsymbol{u})_{f,q}) \mathbf{I} \right) \right] \Gamma_f \\ &= \sum_{f \in \mathcal{F}_P^{\text{int}}} \left[\sum_{q=1}^{q=N_{f,q}} \alpha_q \left(\mu \sum_{N \in \mathcal{S}_f} (\boldsymbol{c}_{x_N} \cdot \boldsymbol{n}_f) \mathbf{I} \boldsymbol{u}_N + \mu \sum_{N \in \mathcal{S}_f} (\boldsymbol{c}_{x_N} \boldsymbol{n}_f^T) \boldsymbol{u}_N + \lambda \sum_{N \in \mathcal{S}_f} (\boldsymbol{n}_f \boldsymbol{c}_{x_N}^T) \boldsymbol{u}_N \right) \right] \Gamma_f \end{aligned} \tag{B13}$$

Provjeri sto se desava sa simetrijom i displacement rubom pa nadopisi

$$\begin{aligned} &\oint_{\Gamma_P} \boldsymbol{n} \cdot \boldsymbol{\sigma}_s \, \mathrm{d}\Gamma_P = \sum_{f \in \mathcal{F}_P} \int_{\Gamma_f} \boldsymbol{n} \cdot \boldsymbol{\sigma}_s \, \mathrm{d}\Gamma_f \approx \sum_{f \in \mathcal{F}_P^{\text{int}}} \boldsymbol{n}_f \cdot \left[\sum_{q=1}^{q=N_{f,q}} \alpha_q \, (\boldsymbol{\sigma}_s)_{f,q} \right] \Gamma_f \\ &+ \sum_{b \in \mathcal{F}_P^{\text{non-trac}}} \boldsymbol{n}_b \cdot \left[\sum_{q=1}^{q=N_{f,q}} \alpha_q \, (\boldsymbol{\sigma}_s)_{b,q} \right] \Gamma_b + \sum_{b \in \mathcal{F}_P^{\text{trac}}} \boldsymbol{n}_b \cdot \sum_{q=1}^{q=N_{f,q}} \boldsymbol{T}_{b,q} \Gamma_b, \end{aligned} \tag{B14}$$

Appendix C High-order Derivatives

The reconstruction of higher-order derivatives is performed for each component of the displacement vector field $\boldsymbol{u} = [u_x,\, u_y,\, u_z]^{\mathrm{T}}$. Specifically, the Hessian ($\nabla\nabla\boldsymbol{u}$) and the third-order derivative tensor ($\nabla\nabla\nabla\boldsymbol{u}$) appearing in Equation (9) are obtained by independently reconstructing the corresponding scalar derivatives of u_x, u_y, and u_z.

For a scalar field u representing one displacement component, the Hessian at the cell centre P is reconstructed as a weighted sum of the neighbouring cell values contained in the stencil $\mathcal{S}_P$:

$$(\nabla\nabla\boldsymbol{u})_P^u = \sum_{N\in\mathcal{S}_P} \boldsymbol{c}_{\boldsymbol{x}\boldsymbol{x}_N}\phi_N, \tag{C15}$$

where $\boldsymbol{c}_{\boldsymbol{x}\boldsymbol{x}_N}$ is a symmetric second-order coefficient tensor:

$$\boldsymbol{c}_{\boldsymbol{x}\boldsymbol{x}_N} = \begin{bmatrix} c_{xx_N} & c_{xy_N} & c_{xz_N} \\ c_{xy_N} & c_{yy_N} & c_{yz_N} \\ c_{xz_N} & c_{yz_N} & c_{zz_N} \end{bmatrix}. \tag{C16}$$

Coefficients in tensor $\boldsymbol{c}_{\boldsymbol{x}\boldsymbol{x}_N}$ are derived from the rows of the reconstruction matrix $\bar{\mathbf{A}}$ corresponding to the second-order monomials. The complete displacement Hessian ($\nabla\nabla\boldsymbol{u}$) is assembled from the Hessians of its three scalar components.

For $p = 3$, the third-order derivative tensor $(\nabla\nabla\nabla u)_P$ for each displacement component is computed as:

$$(\nabla\nabla\nabla\boldsymbol{u})_P^u = \sum_{N\in\mathcal{S}_P} \mathbf{c}_{\boldsymbol{x}\boldsymbol{x}\boldsymbol{x}_N} u_N, \tag{C17}$$

where $\mathbf{c}_{\boldsymbol{x}\boldsymbol{x}\boldsymbol{x}_N}$ is a symmetric third-order coefficient tensor $\mathbf{c}_{\boldsymbol{x}\boldsymbol{x}\boldsymbol{x}_N} = \left[c_{xxx_N}, c_{xxy_N}, c_{xxz_N}, c_{xyy_N}, c_{xyz_N}, c_{xzz_N}, c_{yyy_N}, c_{yyz_N}, c_{yzz_N}, c_{zzz_N}\right]^{\mathrm{T}}$ derived from the rows of the reconstruction matrix $\bar{\mathbf{A}}$.

To mitigate ill-conditioning in the reconstruction procedure, the polynomial basis $\mathbf{q}$ in Equation (15) is scaled using the characteristic stencil size $h = 2r_s$:

$$\mathbf{q}^T(\boldsymbol{x} - \tilde{\boldsymbol{x}}) = \left[1, \frac{(x-\tilde{x})}{h}, \frac{(y-\tilde{y})}{h}, \frac{(z-\tilde{z})}{h}, \frac{1}{2}\frac{(x-\tilde{x})^2}{h^2}, \dots\right]. \tag{C18}$$

Accordingly, the monomials and their corresponding coefficients in Equations (C15) and (C17) are scaled by h^{-2} for the second-order (Hessian) terms and by h^{-3} for the third-order derivative tensor. check is 1/2 going into q or a matrix

Appendix D Time integration scheme

For the unsteady problem considered in this study, the second-order backward differentiation formula (BDF2) is employed for time integration. The inertia term (Equation (7)) is discretized using a single quadrature point per cell, located at the cell centre P, with the acceleration at that point

computed as:

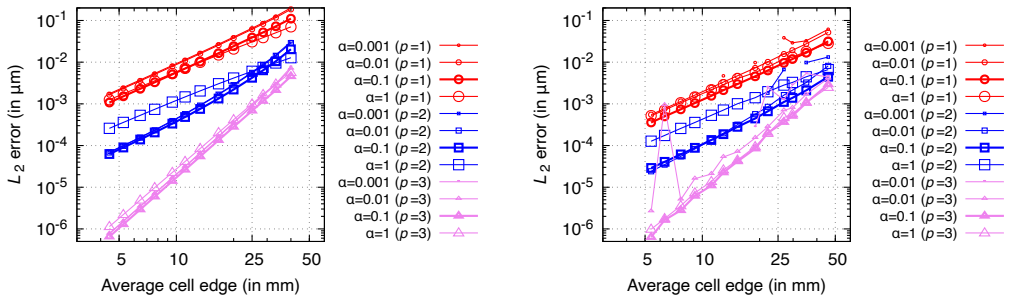
$$\cdot\left(\frac{\partial^2 \boldsymbol{u}_P}{\partial t^2}\right)_P \approx \frac{3\boldsymbol{v}_P^{[t+1]} - 4\boldsymbol{v}_P^{[t]} + \boldsymbol{v}_P^{[t-1]}}{2\Delta t} \approx \frac{3\left(\dfrac{3\boldsymbol{u}_P^{[t+1]} - 4\boldsymbol{u}_P^{[t]} + \boldsymbol{u}_P^{[t-1]}}{2\Delta t}\right) - 4\boldsymbol{v}_P^{[t]} + \boldsymbol{v}_P^{[t-1]}}{2\Delta t} \tag{D19}$$

where Δt is the time increment – assumed constant here. Superscript $[t]$ indicates the time level, with $\boldsymbol{u}_P^{[t+1]}$ corresponding to the unknown displacement at the current time step. The velocity vector $\boldsymbol{v} = \partial \boldsymbol{u}/\partial t$ at the current time step is also updated using the BDF2 scheme as

$$\boldsymbol{v}_P^{[t+1]} \approx \frac{3\boldsymbol{u}_P^{[t+1]} - 4\boldsymbol{u}_P^{[t]} + \boldsymbol{u}_P^{[t-1]}}{2\Delta t}. \tag{D20}$$

Consequently, the displacement and velocity at the two previous time steps must be stored, or alternatively, the displacement at the previous four time steps. The discretisation of the acceleration term in Equation 7 can also be performed using higher-order time integration schemes. However, such formulations require integration over multiple volume quadrature points, rather than a single point, which is sufficient for the second-order scheme employed here.

Appendix E Stabilisation using *over-integration*



(a) Structured hexahedral mesh

(b) Unstructured tetrahedral mesh

Fig. E1 Convergence of the L_2 displacement error norm for case 5.1.1 with varying stabilisation parameter: $\alpha = 0.001, 0.01, 0.1$, and 1.0.

Appendix F List of things to check/discuss/improve in the paper

- The number of integration points is mentioned in section 5, maybe we should mention that before that section?
- Only for quadrature point sum i'm using different notation, i should use $q \in \mathcal{N}_q$ or something like that. Also in that case it make sense to describe what is going into $q \in \mathcal{N}_q$ and how weights are scaled? Now i explain this only by text.
- High-order derivatives appendix is not going into details how product $d : \nabla\nabla\boldsymbol{u}$ and $d :: \nabla\nabla\nabla\boldsymbol{u}$ is calculated.
- Time scheme is mentioned in dynamic example and not before in text.
- add that alpha is used in ucns3

References

[1] Wang, Z.J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H.T., Kroll, N., May, G., Persson, P.-O., Leer, B., Visbal, M.: High-order cfd methods: current status and perspective. International Journal for Numerical Methods in Fluids **72**(8), 811–845 (2013) https://doi.org/10.1002/fld.3767 https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.3767

[2] Nishikawa, H., White, J.A.: An economical third-order nodal-gradient cell-centered finite-volume method for mixed-element grids. Journal of Computational Physics **540**, 114292 (2025) https://doi.org/10.1016/j.jcp.2025.114292

[3] Antoniadis, A.F., Drikakis, D., Farmakis, P.S., Fu, L., Kokkinakis, I., Nogueira, X., Silva, P.A.S.F., Skote, M., Titarev, V., Tsoutsanis, P.: Ucns3d: An open-source high-order finite-volume unstructured cfd solver. Computer Physics Communications **279**, 108453 (2022) https://doi.org/10.1016/j.cpc.2022.108453

[4] Costa, R., Clain, S., Machado, G.J., Nóbrega, J.M.: Very high-order accurate finite volume scheme for the steady-state incompressible navier–stokes equations with polygonal meshes on arbitrary curved boundaries. Computer Methods in Applied Mechanics and Engineering **396**, 115064 (2022) https://doi.org/10.1016/j.cma.2022.115064

[5] Costa, R., Clain, S., Machado, G.J., Nóbrega, J.M., Beirão da Veiga, H., Crispo, F.: Imposing slip conditions on curved boundaries for 3d incompressible flows with a very high-order accurate finite volume scheme on polygonal meshes. Computer Methods in Applied Mechanics and Engineering **415**, 116274 (2023) https://doi.org/10.1016/j.cma.2023.116274

[6] Demirdžić, I.: A fourth-order finite volume method for structural analysis. Applied Mathematical Modelling **40**, 3104–3114 (2016)

[7] High-order finite volume method for linear elasticity on unstructured meshes. Computers & Structures **268**, 106829 (2022) https://doi.org/10.1016/j.compstruc.2022.106829

[8] Castrillo, P., Schillaci, E., Rigola, J.: High-order cell-centered finite volume method for solid dynamics on unstructured meshes. Computers & Structures **295**, 107288 (2024) https://doi.org/10.1016/j.compstruc.2024.107288

[9] Kolev, T., Fischer, P., Min, M., Dongarra, J., Brown, J., Dobrev, V., Warburton, T., Tomov, S., Shephard, M.S., Abdelfattah, A., Barra, V., Beams, N., Camier, J.-S., Chalmers, N., Dudouit, Y., Karakus, A., Karlin, I., Kerkemeier, S., Lan, Y.-H., Medina, D., Merzari, E., Obabko, A., Pazner, W., Rathnayake, T., Smith, C.W., Spies, L., Swirydowicz, K., Thompson, J., Tomboulides, A., Tomov, V.: Efficient exascale discretizations: High-order finite element methods. The International Journal of High Performance Computing Applications **35**(6), 527–552 (2021) https://doi.org/10.1177/10943420211020803 https://doi.org/10.1177/10943420211020803

[10] Bathe, K.J.: Finite Element Procedures. Prentice Hall, ??? (2006). https://books.google.hr/books?id=rWvefGICfO8C

[11] Demirdžić, I., Martinović, D., Ivanković, A.: Numerical simulation of thermal deformation in welded workpiece. Zavarivanje **31**, 209–219 (1988). In Croatian. English translation available at https://www.researchgate.net/profile/Alojz_Ivankovic/publication/296148474_Numerical_simulation_of_thermal_deformation_in_welded_workpiece/links/5d07642ba6fdcc39f12219eb/Numerical-simulation-of-thermal-deformation-in-welded-workpiece.pdf

[12] Cardiff, P., Demirdžić, I.: Thirty years of the finite volume method for solid mechanics. Archives of Computational Methods in Engineering **28**(5), 3721–3780 (2021)

[13] Cueto-Felgueroso, L., Colominas, I., Nogueira, X., Navarrina, F., Casteleiro, M.: Finite volume solvers and moving least-squares approximations for the compressible navier–stokes equations on unstructured grids. Computer Methods in Applied Mechanics and Engineering **196**(45), 4712–4736 (2007) https://doi.org/10.1016/j.cma.2007.06.003

[14] Ramírez, L., Nogueira, X., Khelladi, S., Chassaing, J.-C., Colominas, I.: A new higher-order finite volume method based on moving least squares for the resolution of the incompressible navier–stokes equations on unstructured grids. Computer Methods in Applied Mechanics and Engineering **278**, 883–901 (2014) https://doi.org/10.1016/j.cma.2014.06.028

[15] Pablo Green, C.: High-order finite volume method for solid dynamics in fluid-structure interaction applications. PhD thesis, niversitat Politècnica de Catalunya (2023)

[16] Gear, C.W., Saad, Y.: Iterative solution of linear equations in ODE codes. SIAM Journal on Scientific and Statistical Computing **4**(4), 583–601 (1983) https://doi.org/10.1137/0904040

[17] Chan, T.F., Jackson, K.R.: Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms. SIAM Journal on Scientific and Statistical Computing **5**(3), 533–542 (1984) https://doi.org/10.1137/0905039

[18] Brown, P.N., Hindmarsh, A.C.: Matrix-free methods for stiff systems of ODE's. SIAM Journal on Numerical Analysis **23**(3), 610–638 (1986) https://doi.org/10.1137/0723034

[19] Brown, P.N., Saad, Y.: Hybrid Krylov methods for nonlinear systems of equations. SIAM Journal on Scientific and Statistical Computing **11**(3), 450–481 (1990) https://doi.org/10.1137/0911026

[20] Qin, N., Ludlow, D.K., Shaw, S.T.: A matrix-free preconditioned Newton/GMRES method for unsteady Navier–Stokes solutions. International Journal for Numerical Methods in Fluids **33**(2), 223–248 (2000) https://doi.org/10.1002/(SICI)1097-0363(20000530)33:2⟨223::AID-FLD10⟩3.0.CO;2-V

[21] Geuzaine, P.: Newton-Krylov strategy for compressible turbulent flows on unstructured meshes. AIAA Journal **39**(3), 528–531 (2001) https://doi.org/10.2514/2.1339

[22] Pernice, M., Tocci, M.D.: A multigrid-preconditioned Newton–Krylov method for the incompressible Navier–Stokes equations. SIAM Journal on Scientific Computing **23**(2), 398–418 (2001) https://doi.org/10.1137/S1064827500372250

[23] Nejat, A., Jalali, A., Sharbatdar, M.: A Newton–Krylov finite volume algorithm for the

power-law non-Newtonian fluid flow using pseudo-compressibility technique. Journal of Non-Newtonian Fluid Mechanics **166**(19–20), 1158–1172 (2011) https://doi.org/10.1016/j.jnnfm.2011.07.003

[24] Nakashima, Y., Higo, Y., Nishikawa, H.: Recent algorithmic advances in the scflow unstructured-polyhedral-grid cfd solver. (2025). https://doi.org/10.2514/6.2025-0075

[25] McHugh, P.R., Knoll, D.A.: Comparison of standard and matrix-free implementations of several Newton–Krylov solvers. AIAA Journal **32**(12), 2394–2400 (1994) https://doi.org/10.2514/3.12305

[26] Knoll, D.A., Keyes, D.E.: Jacobian-free Newton–Krylov methods: a survey of approaches and applications. Journal of Computational Physics **193**(2), 357–397 (2004) https://doi.org/10.1016/j.jcp.2003.08.010

[27] Vaassen, J.-M., Vigneron, D., Essers, J.-A.: An implicit high order finite volume scheme for the solution of 3d Navier–Stokes equations with new discretization of diffusive terms. Journal of Computational and Applied Mathematics **215**(2), 595–601 (2008) https://doi.org/10.1016/j.cam.2006.04.066

[28] Lucas, P., Zuijlen, A.H., Bijl, H.: Fast unsteady flow computations with a jacobian-free Newton–Krylov algorithm. Journal of Computational Physics **229**(24), 9201–9215 (2010) https://doi.org/10.1016/j.jcp.2010.08.033

[29] Nishikawa, H.: A hyperbolic poisson solver for tetrahedral grids. Journal of Computational Physics **409**, 109358 (2020) https://doi.org/10.1016/j.jcp.2020.109358

[30] Cardiff, P., Armfield, D., Tuković, Batistić, I.: A Jacobian-free Newton-Krylov method for cell-centred finite volume solid mechanics (2025). https://arxiv.org/abs/2502.17217

[31] Weller, H.G., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object orientated techniques. Computers in Physics **12**, 620–631 (1998)

[32] Cardiff, P., Batistić, I., Tuković: solids4foam: A toolbox for performing solid mechanics and fluid-solid interaction simulations in openfoam. J. Open Source Softw. **10**, 7407 (2025)

[33] Nishikawa, H.: Beyond interface gradient: A general principle for constructing diffusion schemes. In: 40th Fluid Dynamics Conference and Exhibit. AIAA, Chicago, Illinois, USA (2010). https://doi.org/10.2514/6.2010-5093 . Published online: 13 Nov 2012

[34] Nishikawa, H.: Robust and accurate viscous discretization via upwind scheme – i: Basic principle. Computers & Fluids **49**(1), 62–86 (2011) https://doi.org/10.1016/j.compfluid.2011.04.014

[35] Nishikawa, H.: On Second- and Higher-Order Finite-Volume Schemes for Viscous Terms on Unstructured Grids. https://doi.org/10.2514/6.2025-3674 . https://arc.aiaa.org/doi/abs/10.2514/6.2025-3674

[36] Nishikawa, H.: Economically high-order unstructured-grid methods: Clarification and efficient fsr schemes. International Journal for Numerical Methods in Fluids **93**(11), 3187–3214 (2021) https://doi.org/10.1002/fld.5028 https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.5028

[37] Dunavant, D.A.: High degree efficient symmetrical gaussian quadrature rules for the triangle. International Journal for Numerical Methods in Engineering **21**(6), 1129–1148 (1985)

[38] Shunn, L., Ham, F.E.: Symmetric quadrature rules for tetrahedra based on a cubic close-packed lattice arrangement. Journal of Computational and Applied Mathematics **236**(17), 4348–4364 (2012)

[39] Cardiff, P., Tuković, Ž., Jaeger, P.D., Clancy, M., Ivanković, A.: A Lagrangian cell-centred finite volume method for metal forming simulation. International Journal for Numerical Methods in Engineering **109**(13), 1777–1803 (2017) https://doi.org/10.1002/nme.5345 https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.5345

[40] Cardiff, P., Karač, A., Jaeger, P.D., Jasak, H., Nagy, J., Ivanković, A., Tuković, Ž.: An open-source finite volume toolbox for solid mechanics and fluid-solid interaction simulations (2018) https://doi.org/10.48550/arXiv.1808.10736 . https://arxiv.org/abs/1808.10736

[41] Rhie, C.M., Chow, W.L.: Numerical study of the turbulent flow past an airfoil with trailing edge separation. AIAA Journal **21**(11), 1525–1532 (1983) https://doi.org/10.2514/3.8284 https://doi.org/10.2514/3.8284

[42] Tuković, Ž., Ivanković, A., Karač, A.: Finite-volume stress analysis in multi-material linear elastic body. International Journal for Numerical Methods in Engineering **93**(4), 400–419 (2013) https://doi.org/10.1002/nme.4390 https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.4390

[43] Strang, G.: Linear Algebra and Its Applications, 2^{nd} edn. Academic Press, ??? (2012)

[44] Khelladi, S., Nogueira, X., Bakir, F., Colominas, I.: Toward a higher order unsteady finite volume solver based on reproducing kernel methods. Computer Methods in Applied Mechanics and Engineering **200**(29), 2348–2362 (2011) https://doi.org/10.1016/j.cma.2011.04.001

[45] Tsoutsanis, P.: Stencil selection algorithms for weno schemes on unstructured meshes. Journal of Computational Physics **475**, 108840 (2023) https://doi.org/10.1016/j.jcp.2019.07.039

[46] Demirdžić, I., Cardiff, P.: Symmetry plane boundary conditions for cell-centered finite-volume continuum mechanics. Numerical Heat Transfer, Part B: Fundamentals (2022) https://doi.org/10.1080/10407790.2022.2105073

[47] Tuković, Ž., Karač, A., Cardiff, P., Jasak, H., Ivanković, A.: OpenFOAM finite volume solver for fluid-solid interaction. Transactions of FAMENA **42**(3), 1–31 (2018) https://doi.org/10.21278/TOF.42301

[48] Demirdžić, I.: On the discretization of the diffusion term in finite-volume continuum mechanics. Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology **68:1**, 1–10 (2015) https://doi.org/10.1080/10407790.2014.985992

[49] Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Rupp, K., Smith, B.F., Zampini, S., Zhang, H.: PETSc Web page. http://www.mcs.anl.gov/petsc (2015). http://www.mcs.anl.gov/petsc

[50] Geuzaine, C., Remacle, J.-F.: Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. International journal for numerical methods in engineering **79**(11), 1309–1331 (2009) https://doi.org/10.1002/nme.2579

[51] Mazzanti, F.: Coupled vertex-centred finite volume methods for large-strain elastoplasticity. PhD thesis, University College Dublin (2024). For examination

[52] Bijelonja, I., Demirdžić, I., Muzaferija, S.: A finite volume method for incompressible linear elasticity. Computer Methods in Applied Mechanics and Engineering **195**, 6378–6390 (2006)

[53] Bijelonja, I., Demirdžić, I., Muzaferija, S.: A finite volume method for large strain analysis of incompressible hyperelastic materials. International Journal for Numerical Methods in Engineering **64**, 1594–1609 (2005)

[54] Timoshenko, S., Goodier, J.: Theory of Elasticity. McGraw-Hill Company, Inc., New York (1970)

[55] Green, A.E., Zerna, W.: Theoretical Elasticity. Courier Corporation, ??? (1992)

[56] Zienkiewicz, O.C., Taylor, R.L.: The Finite Element Method vol. 2. Butterworth-heinemann, Berlin, Germany (2000)

[57] Areias, P.: Simplas. Portuguese Software Association (ASSOFT). http://www.simplassoftware.com.

[58] Pelteret, J.-P., McBride, A.: The deal.II code gallery: Quasi-Static Finite-Strain Compressible Elasticity. Zenodo (2018). https://doi.org/10.5281/zenodo.1228964 . https://doi.org/10.5281/zenodo.1228964

[59] Demirdžić, I., Muzaferija, S., Perić, M.: Benchmark solutions of some structural analysis problems using finite-volume method and multigrid acceleration. International journal for numerical methods in engineering **40**(10), 1893–1908 (1997)

[60] Southwell, R.V., Gough, H.J.: Vi. on the concentration of stress in the neighbourhood of a small spherical flaw; and on the propagation of fatigue fractures in "statistically isotropic" materials. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **1**(1), 71–97 (1926) https://doi.org/10.1080/14786442608633614 https://doi.org/10.1080/14786442608633614

[61] Goodier, J.N.: Concentration of stress around spherical and cylindrical inclusions and flaws. Journal of Applied Mechanics **1**(2), 39–44 (1933)

[62] Land, S., Gurev, V., Arens, S., Augustin, C.M., Baron, L., Blake, R., Bradley, C., Castro, S., Crozier, A., Favino, M., Fastl, T.E., Fritz, T., Gao, H., Gizzi, A., Griffith, B.E., Hurtado, D.E.,

Krause, R., Luo, X., Nash, M.P., Pezzuto, S., Plank, G., Rossi, S., Ruprecht, D., Seemann, G., Smith, N.P., Sundnes, J., Rice, J.J., Trayanova, N., Wang, D., Wang, Z.J., Niederer, S.A.: Verification of cardiac mechanics software: benchmark problems and solutions for testing active and passive material behaviour **471**(2184), 20150641 (2015) https://doi.org/10.1098/rspa.2015.0641

[63] Guccione, J.M., Costa, K.D., McCulloch, A.D.: Finite element stress analysis of left ventricular mechanics in the beating dog heart. Journal of Biomechanics **28**(10), 1167–1177 (1995) https://doi.org/10.1016/0021-9290(94)00174-3