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

Potential structure of this section:

1. Higher-order discretisation in computational solid mechanics remains relatively underexplored compared to fluid dynamics, despite its potential for improving accuracy and efficiency.

Comment on 'why' we should care about higher order: potentially cheaper to achieve the same mesh error; potential to reduce the amount of data when running large models on HPC systems (where data is the bottleneck).

2. Recent research has demonstrated that higher-order accuracy can be achieved in cell-centred formulations through gradient reconstruction (e.g. least-squares approaches) and face integration techniques (e.g. Gaussian quadrature); however, their efficiency and ease of implementation in an industrial code are unexplored.

3. The solution strategy is a crucial factor: segregated iterative solvers remain the standard in finite-volume solid mechanics but often suffer from slow convergence and limited robustness, and they are efficient for higher orders, as shown by Demirdzic. Jacobian-free Newton–Krylov (JFNK) methods have emerged as a promising alternative, avoiding the need for explicit Jacobian construction while enabling quadratic convergence through Krylov subspace acceleration. In addition, the JFNK approach is well-suited for integration into existing segregated frameworks. Until the recent introduction of the Jacobian-free Newton–Krylov (JFNK) algorithm as an option for finite volume solid mechanics solvers, the most common solution strategy was based on the segregated approach, which may be interpreted as a quasi-Newton method. Alternatively, some studies employed the Newton method for both linear material behaviour (special case of the Newton procedure) and for nonlinear materials. The primary barrier to the wider adoption of full Newton methods has been the complexity of the required linearization. For non-linear material models, such as those involving plasticity, the derivation, implementation, and assembly of the analytical Jacobian is a notoriously complex. The present work addresses this critical limitation by adopting the JFNK algorithm, which eliminates the need for explicit Jacobian assembly while retaining the quadratic convergence properties of Newton's method.

4. Our previous work established the feasibility and efficiency of JFNK for second-order cell-centred formulations, where a compact-stencil approximate Jacobian proved highly effective as a preconditioner. This study extends JFNK to third- and fourth-order cell-centred finite-volume formulations for linear and nonlinear solid mechanics, combining least-squares gradient reconstruction with Gaussian quadrature integration.

5. We demonstrate that the proposed approach significantly improves accuracy while retaining computational efficiency, and we show that compact-stencil Jacobians remain effective as preconditioners in the higher-order setting.

6. The aim of this paper is therefore twofold: to contribute to the relatively sparse literature on higher-order cell-centred finite-volume methods for solids, and to explore the synergy between such methods and Jacobian-free Newton–Krylov solution strategies.

7. pokazano je da se moze splitat surface topologija cv-a na surface

8. nemamo proracunske tocke na rubu

9. dodali smo stabilizaciju i pokazali da je nuzna

10. SA JFNK ne moramo konstruirati matricu sto moze biti vremenski zathejno i matematicki zajebano isto skica molekule na symmetry planeu skica molekule na dirichlet boundariu sve jedna slika na kojoj je desni rub dirichlet, dole je symmetry a gore je neumman rub!

Philip comment: we should mention the importance of using consistent areas and volumes, e.g when discretising the volumetric terms we should use the same cell decomposition as we use to calculate the cell volume; the same applies for the face area integrals. Ivan noted this is required for non-flat faces to get the expected order of accuracy, e.g. on the irregular hexahedra/polyhedra meshes.

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

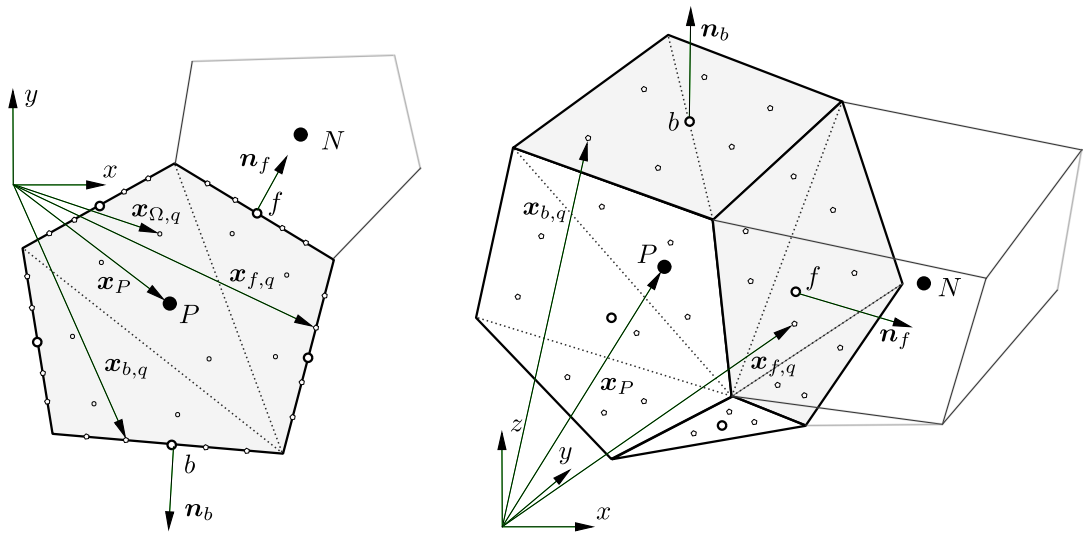
In this work, a finite volume discretization is employed to approximate the strong form of the governing equations. The computational points are located at the cell centroids, where the solution is treated as point-valued rather than cell-averaged. This type of discretization is commonly referred to as the deconvolution finite volume method [1, 2].

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 [3] are applied for triangular subelements, and Shunn and Ham's symmetric quadrature rules [4] 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

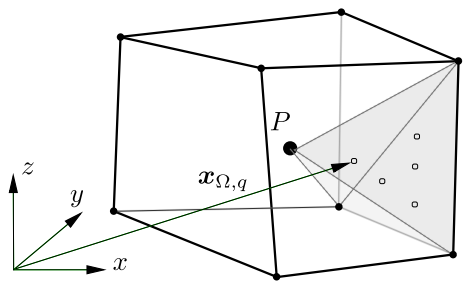


Fig. 2 Volume integration, cell decomposition into tetrahedral elements

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 [1].

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 [5, 6] or interpolation with geometrically weighted coefficients [7–9]. 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 [10], which is second-order accurate [7]. 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 [11], which preserves the designed high-order convergence of the method [5, 6]. 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 [8, 9, 12] 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 [5]. 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 $N_t = \dfrac{(d+p)!}{(d!\,p!)}$ 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 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, see [13, 14].

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

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

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 [15, 16] and solid mechanics problems [13, 14, 17]. 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 [13, 14, 17].

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 [15]. Based on those findings presented in [13, 17], 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, [18] noted that within the $k-$exact least-squares framework, the stencil size is typically chosen in the range $1.5\,|\mathcal{S}|_{\min} \leq |\mathcal{S}| \leq 3\,|\mathcal{S}|_{\min}$.

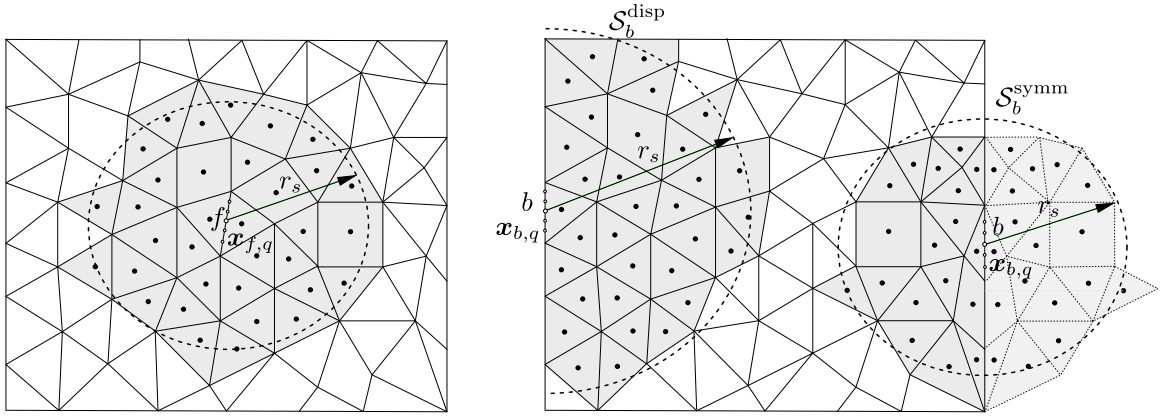
The stencil is constructed using a nearest-neighbour approach, where the set $\mathcal{S}$ comprises the cells closest to the reference point (face or cell centre), enclosed within a sphere of radius r_s. To reduce computational cost, all quadrature points on a given face share the same stencil, and 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 [15, 18]; 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

jedbazyba za tracttion ne treba posebnu obradu - vidi jednazbu surface intehgrals gdje se direktno uzima u obzir Boundary conditions are imposed in a weak sense, i.e. thorugh modifications of the least square systems by introcudin penalty constraints. Note that this work does not consider curved boundaries. This is left for future work and is expected that is enchanched which will not remedy interior accuracy of discretisation bla bla

Note that this work does not consider curved boundary treatement, this is left for future work as it require special care

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 $\mathcal{S}_b^{\text{disp}}$. For symmetry boundary, the contribution of the mirrored (ghost) points is also included, giving:

$$(\nabla \boldsymbol{u})_{b,q} = \sum_{N \in \mathcal{S}_b^{\text{symm,p}}} \boldsymbol{c}_{\boldsymbol{x}_N} \otimes \boldsymbol{u}_N + \sum_{N \in \mathcal{S}_b^{\text{symm,g}}} \boldsymbol{c}_{\boldsymbol{x}_N} \otimes (\boldsymbol{R}_b \cdot \boldsymbol{u}_N), \tag{23}$$

where $\boldsymbol{R}_b = (\mathbf{I} - 2\boldsymbol{n}_b \otimes \boldsymbol{n}_b)$ is reflection tensor [22]. 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 [13, 17] 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:

$$\boldsymbol{R}(\boldsymbol{u}) = 0, \tag{24}$$

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 $k+1$:

$$\boldsymbol{R}(\boldsymbol{u}_{k+1}) = 0, \tag{25}$$

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

$$\boldsymbol{R}(\boldsymbol{u}_{k+1}) \approx \boldsymbol{R}(\boldsymbol{u}_k) \; + \; \boldsymbol{R}'(\boldsymbol{u}_k)(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k), \tag{26}$$

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:

$$\boldsymbol{J}(\boldsymbol{u}_k)\delta\boldsymbol{u} = -\boldsymbol{R}(\boldsymbol{u}_k) \tag{27}$$

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 $\boldsymbol{v}$:

$$\boldsymbol{J}\boldsymbol{v} \approx \frac{\boldsymbol{R}(\boldsymbol{u} + \epsilon\boldsymbol{v}) - \boldsymbol{R}(\boldsymbol{u})}{\epsilon} \tag{28}$$

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:

$$\boldsymbol{J}\boldsymbol{P}^{-1}\boldsymbol{v} \approx \frac{\boldsymbol{R}(\boldsymbol{u} + \epsilon\boldsymbol{P}^{-1}\boldsymbol{v}) - \boldsymbol{R}(\boldsymbol{u})}{\epsilon} \tag{29}$$

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 [9, 19]. Following the classification of Knoll and Keyes [20], 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:

$$\begin{aligned}\boldsymbol{P}_P = \oint_{\Gamma_P} \bar{K}\, \boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\, \mathrm{d}\Gamma_P \approx \sum_{f\in\mathcal{F}_P^{\mathrm{int}}} \bar{K}\,|\boldsymbol{\Delta}_f|\,\frac{\boldsymbol{u}_N - \boldsymbol{u}_P}{|\boldsymbol{d}_{PN}|}\Gamma_f + \sum_{b\in\mathcal{F}_P^{\mathrm{disp}}} \bar{K}\,|\boldsymbol{\Delta}_b|\,\frac{\bar{\boldsymbol{u}}_b - \boldsymbol{u}_P}{|\boldsymbol{d}_{Pb}|}\,|\boldsymbol{\Gamma}_b| \\ + \sum_{b\in\mathcal{F}_P^{\mathrm{symm}}} \bar{K}\,|\boldsymbol{\Delta}_b|\,\frac{\boldsymbol{R}_b\cdot\boldsymbol{u}_P - \boldsymbol{u}_P}{|\boldsymbol{d}_{Pb}|}\Gamma_b,\end{aligned} \tag{30}$$

where $\boldsymbol{\Delta}_f = \boldsymbol{d_{PN}}/(\boldsymbol{d}_{PN}\cdot\boldsymbol{n}_f)$ is *over-relaxed orthogonal* vector [21]. 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) [23] 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 [9, 19, 24], which is built on OpenFOAM-v2312 [25]. An extended discussion with additional implementation details is provided in the authors' previous work [7], where the JFNK solver is coupled with a second-order residual evaluation.

5 Test Cases

Dati napomenu zasto su neki cejsecvi izvrceni sa puno mreza, citirati zadnji niskihavin rad gdje se jasno vide oscilacije. Do koje tocnosti se rjesavalo.

alpha = 0.1 stabilizacija.

root mean square error

$$L_2 = \frac{1}{N_c} \sum_{i=1}^{N_c} |\Delta\phi_i|, \tag{31}$$

and infinity norm representing the maximum absolute error

$$L_{\text{inf}} = \max_{1 \leq i \leq N_c} |\Delta\phi_i|, \tag{32}$$

where $\Delta\phi_i$ is the difference between expected and predicted solutions 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:

$$\sigma_{ekv} = \sqrt{\frac{1}{2}\left[(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2\right] + 3\left(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2\right)} \tag{33}$$

5.1 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 10(a)) is employed in the form or polynomial, trigonometric [26] or mixed form [13]:

- **2D**

$$\boldsymbol{u} = \begin{pmatrix} e^{x^2} \sin(y) \\ \ln(3 + y)\cos(x) + \sin(y) \\ 0. \end{pmatrix} \tag{34}$$

- **3D - polynomial**

$$\boldsymbol{u} = \begin{pmatrix} a_x(x^3 + xy^2) \\ a_y(y^3 + yz^2) \\ a_z(z^3 + zx^2) \end{pmatrix} \tag{35}$$

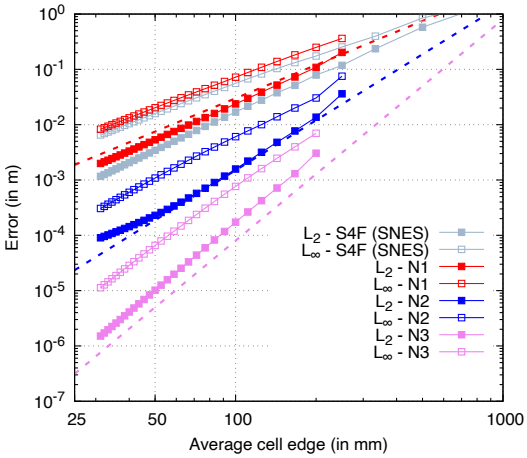
  where $a_x = 1\,\mu$m, $a_y = 1\,\mu$m, and $a_z = 1\,\mu$m.

- **3D - trigonometric**

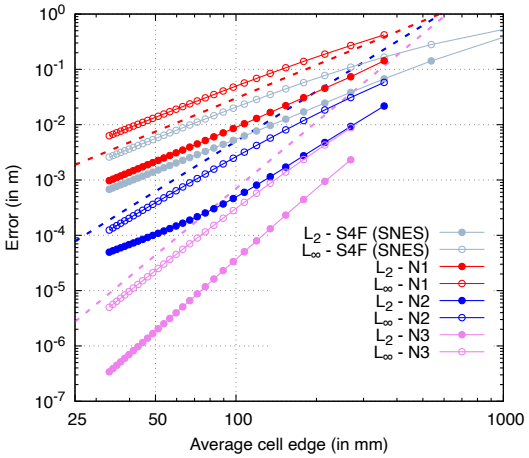
$$\boldsymbol{u} = \begin{pmatrix} a_x \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ a_y \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ a_z \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \end{pmatrix} \tag{36}$$

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

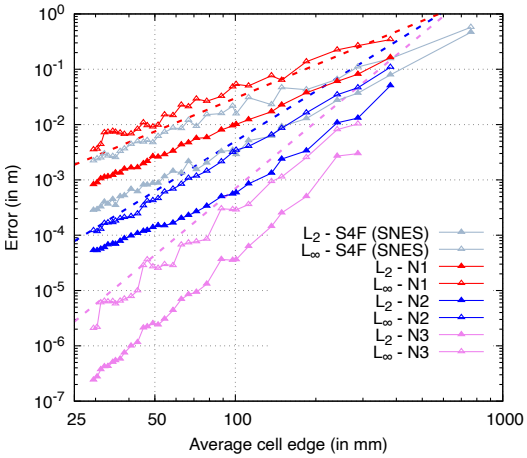
The Cartesian coordinates are given by x, y and z. The corresponding manufactured body force term ($\boldsymbol{f}_b$ in Equation 3) can be found in authors previous publication [7] or obtained by manufactured solution procedure.



(a) Hexahedral mesh

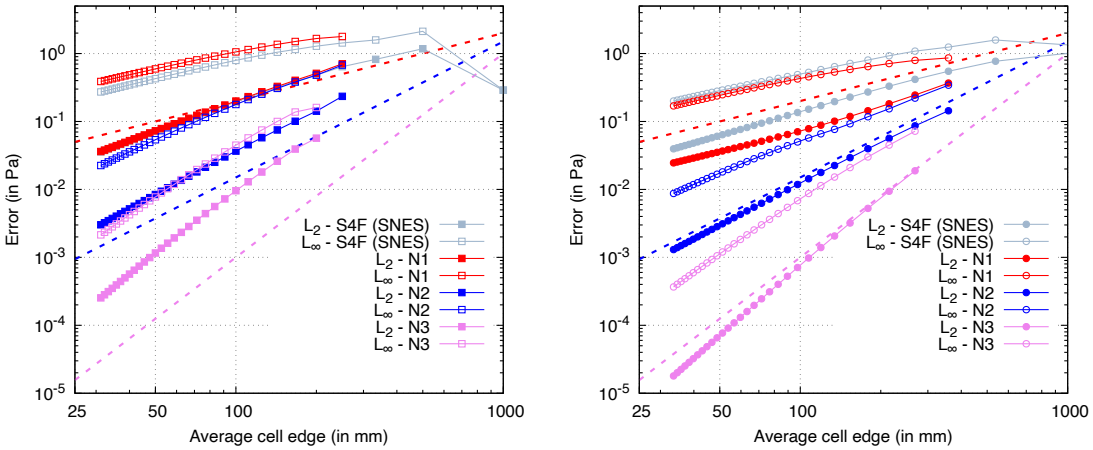


(b) Tetrahedral structured mesh



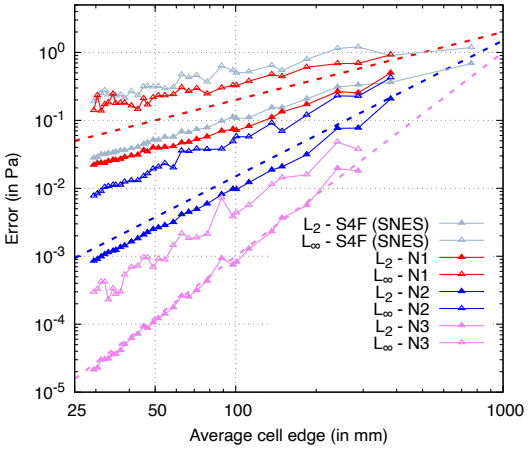
(c) Tetrahedral unstructured mesh

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



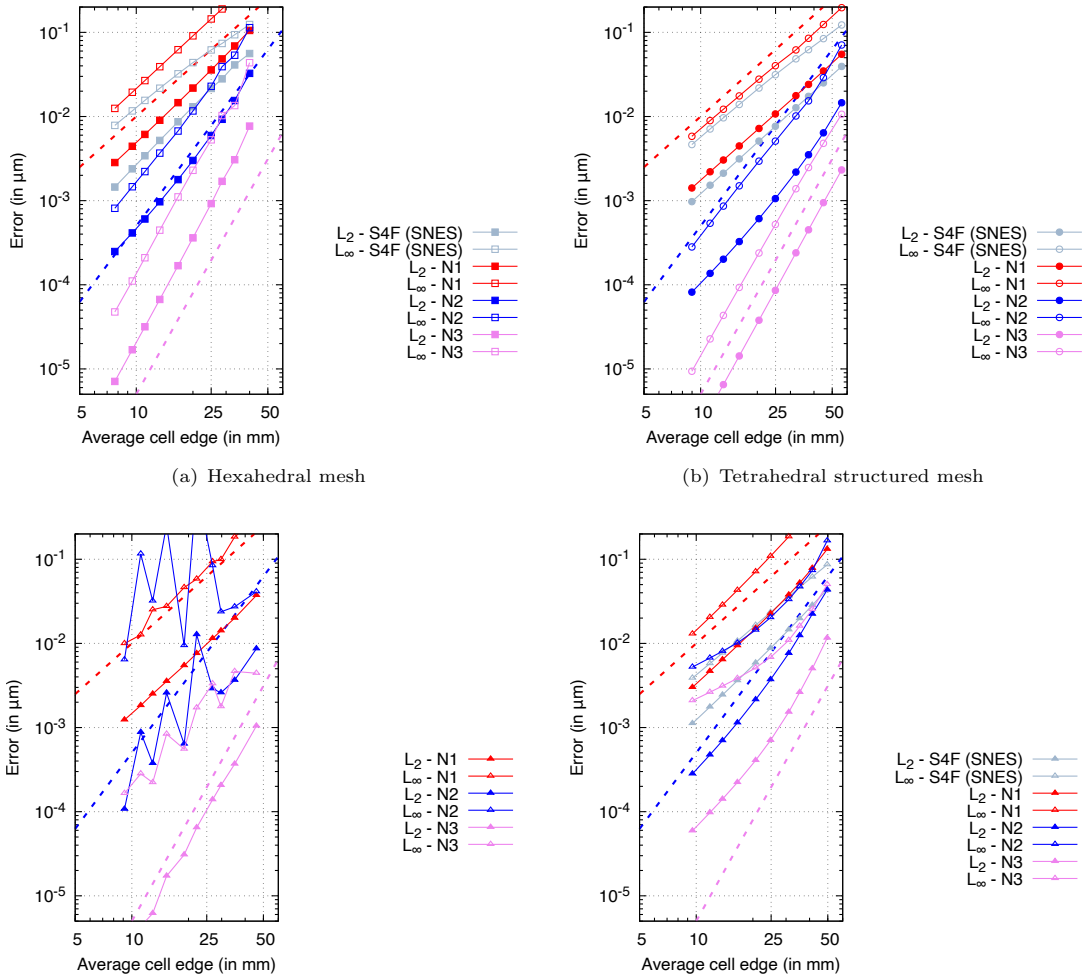
(a) Hexahedral mesh

(b) Tetrahedral structured mesh



(c) Tetrahedral unstructured mesh

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



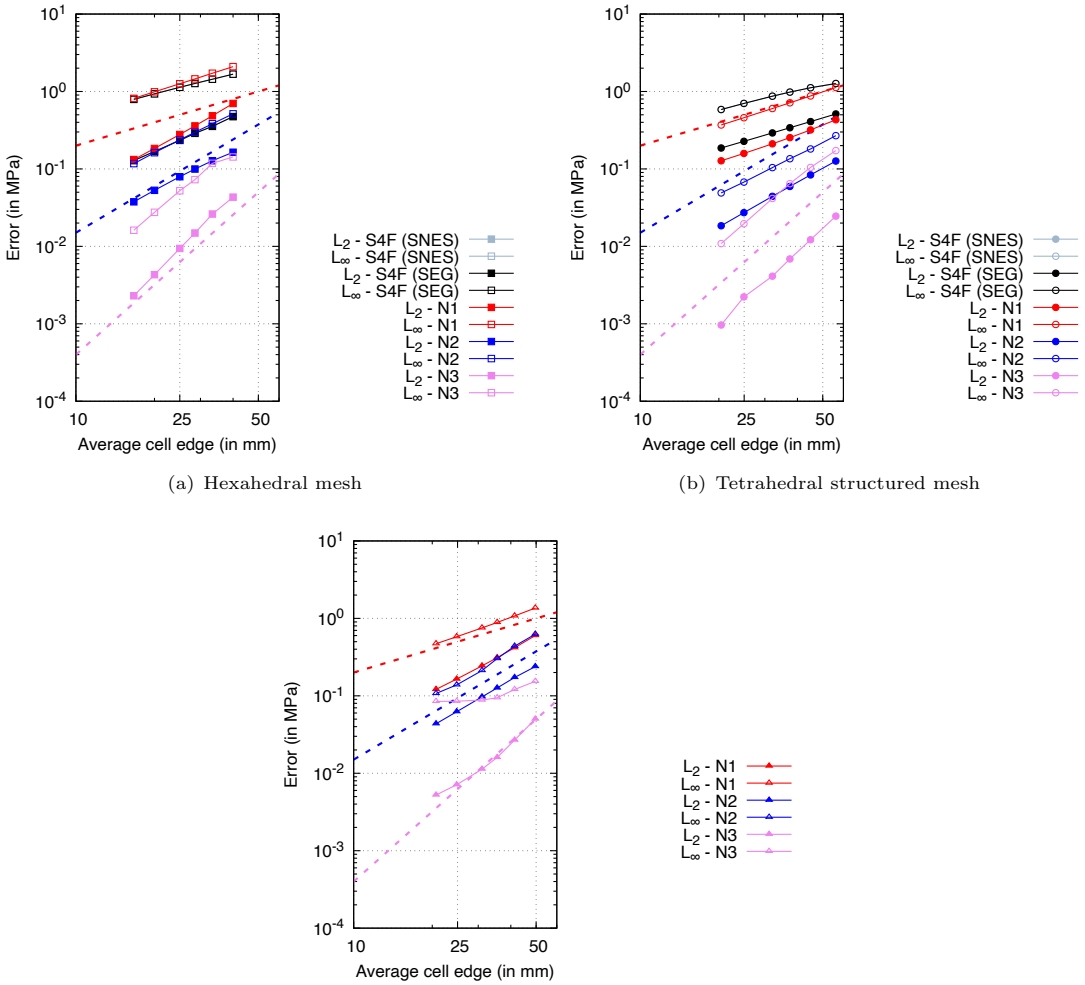
(a) Hexahedral mesh

(b) Tetrahedral structured mesh

(c) Tetrahedral unstructured mesh

(d) Polyhedral mesh

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



(a) Hexahedral mesh

(b) Tetrahedral structured mesh

(c) Polyhedral mesh

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

5.2 Case 2: Cantilever beam

The test case geometry, shown in Fig. 6(a), is a rectangular beam with dimensions of 2 m × 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.

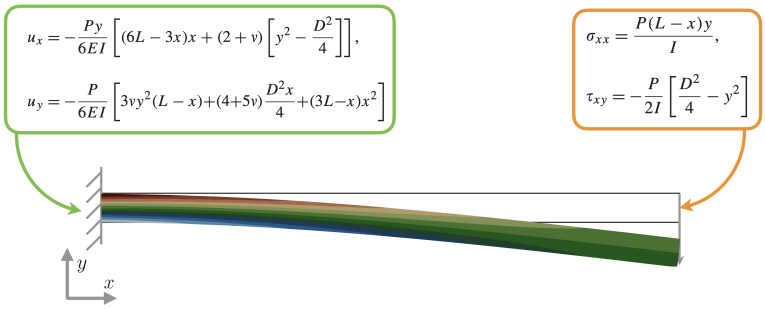
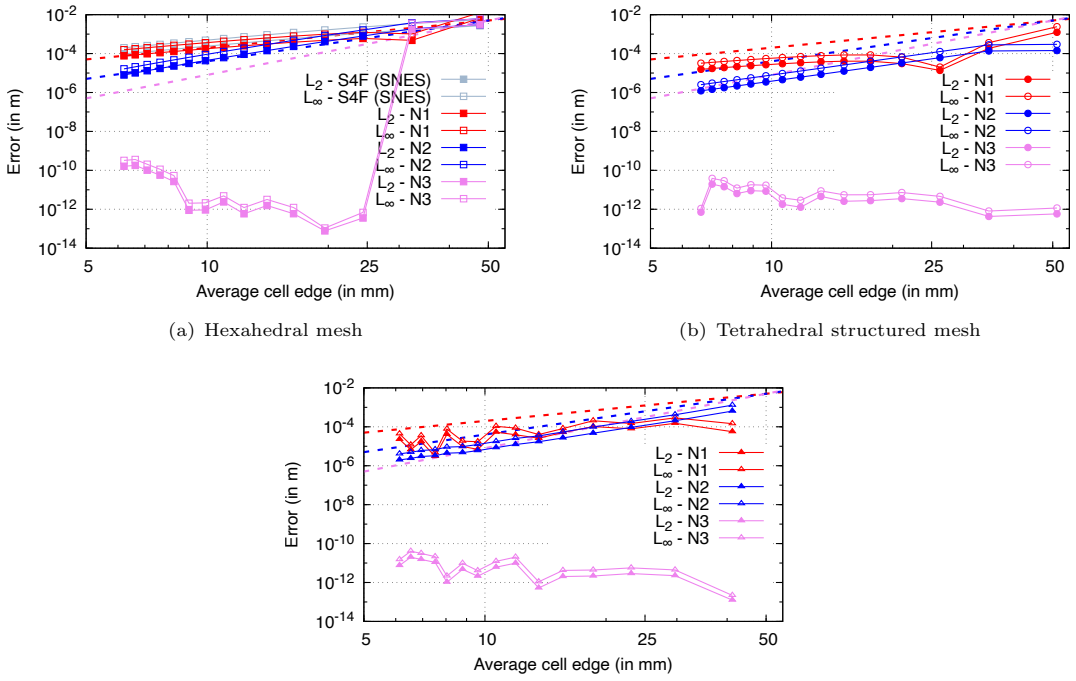


Fig. 8 Cantilever beam case: geometry and boundary conditions

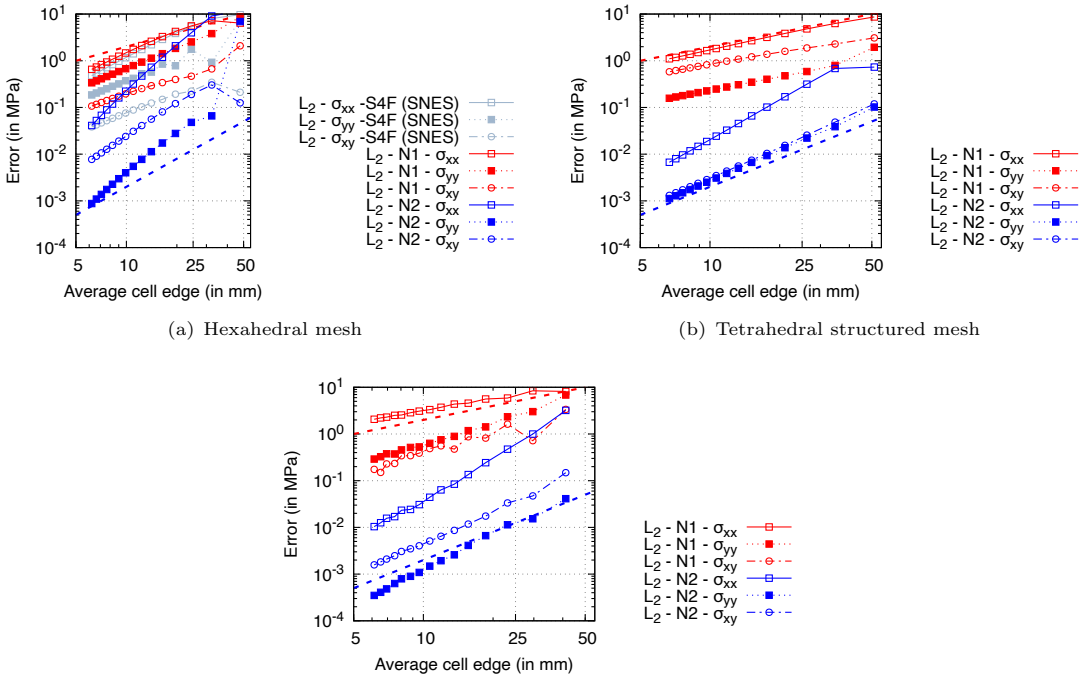


(a) Hexahedral mesh

(b) Tetrahedral structured mesh

(c) Tetrahedral unstructured mesh

Fig. 9 Cantilever beam case: displacement magnitude discretisation errors



(a) Hexahedral mesh

(b) Tetrahedral structured mesh

(c) Tetrahedral unstructured mesh

Fig. 10 Cantilever beam case: σ_{xx}, σ_{yy}, σ_{xy} stress discretisation errors. Results for $N = 3$ are not shown because stress error is around $\sim 10^{-12}$

5.3 Case 3: Internally pressurised thick-walled cylinder

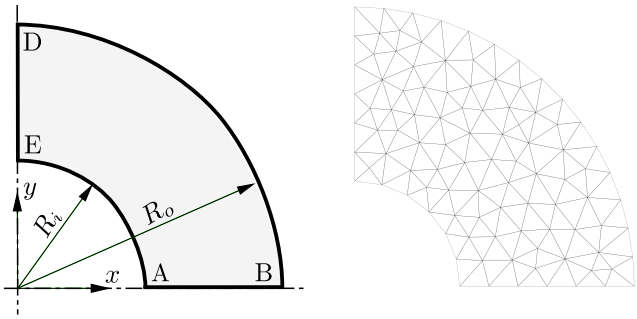
In this case, a homogeneous thick-wall cylindrical pressure vessel with an inner radius $R_i = 7$ m, outer radius $R_o = 18.625$ m, and loaded internally with pressure $p = 100$ MPa is analysed. Two types of material are considered:

i. Small strain, linear-elastic [27]: $E = 10$ GPa, $\nu = 0.3$.

ii. Finite strain, Mooney-Rivlin law [28]: $c_{10} = 80$ MPa, $c_{01} = 20$ MPa, and $c_{11} = 0.0$ MPa and Poisson's ratio $\nu = 0.49$

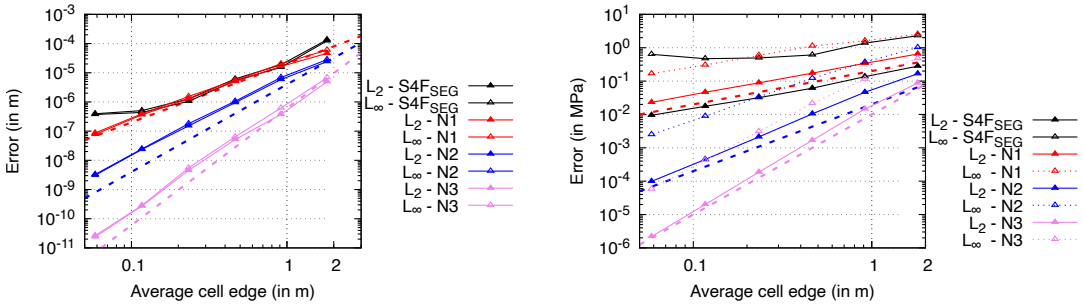
The problem is considered plane stress, with the 2-D computational domain comprising a quarter of the cylinder geometry. The cylinder is discretised with series of unstructured tetrahedral grids. Gravitational and inertial effects are neglected. Linear elastic case is solved using one loading increment while hyperelastic case is solved using 100 equal loading increments. Analytical solutions are available in [29] and [30].

popraviiti slova na slici 8



(a) Case geometry (b) Tetrahedral mesh

Fig. 11 Plate hole case geometry and mesh



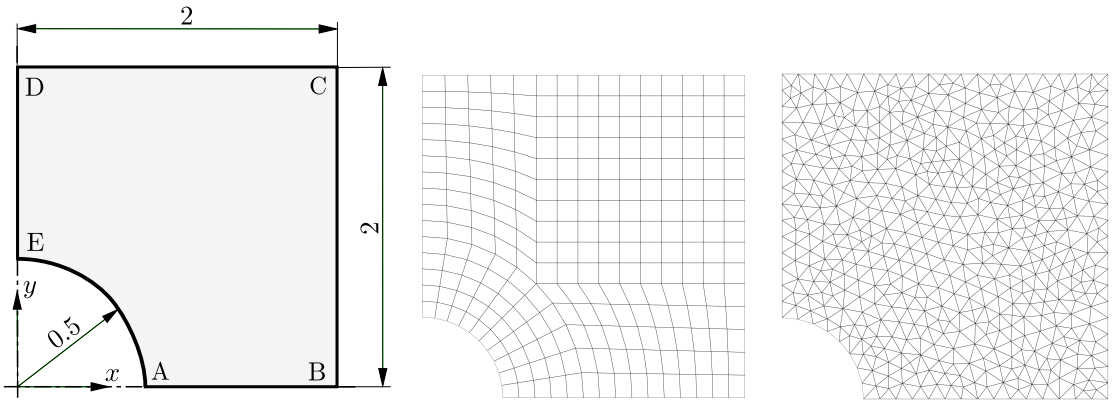
(a) Displacement convergence (b) Stress convergence

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

5.4 Case 4: Cooks membrane

5.5 Case 4: Plate hole

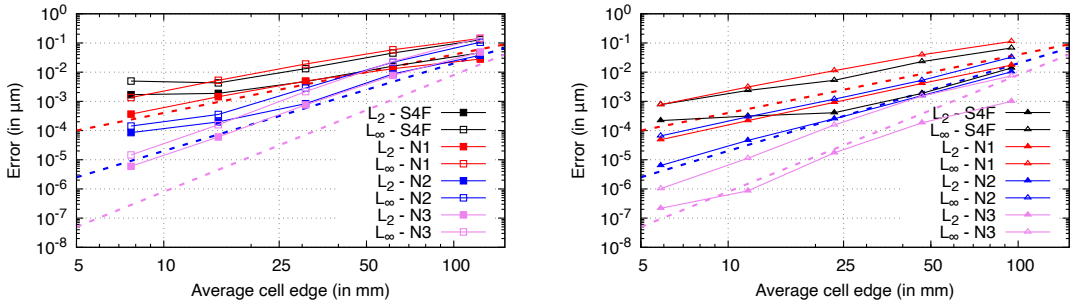
This benchmark problem consists of a thin, infinitely large plate with a circular hole, subjected to uniaxial tension. Owing to the symmetry of the geometry and loading, only one quarter of the plate is modeled as a finite domain. To minimize the influence of the finite computational boundaries, the exact tractions obtained from the analytical solution [31] 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 $\nu = 0.3$.



(a) Case geometry (b) Hexahedral mesh (c) Tetrahedral mesh

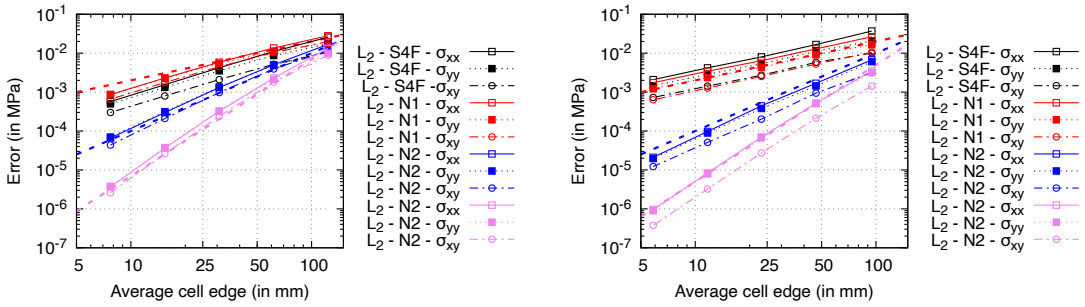
Fig. 13 Plate hole case geometry and mesh

Hexahedral mesh with N3 needs 20 neighbours to avoid ill conditioning. This results in lower slope for displacement but not affecting the stresses. I'm not sure why S4F results flatten on finer meshes.



(a) Hexahedral mesh (b) Tetrahedral mesh

Fig. 14 Plate hole case: displacement magnitude discretisation errors



(a) Hexahedral mesh (b) Tetrahedral mesh

Fig. 15 Plate hole case: σ_{xx}, σ_{yy}, σ_{xy} stress discretisation errors.

5.6 Case 5: Elliptic plate

5.7 Case 6: Spherical cavity

5.8 Case 7: Idealised ventricle

5.9 Error-cost relationship

5.10 Stabilisation scheme effects

5.11 Effects of poor interpolation conditioning

5.12 Different choice for approximate Jacobian

Plate hole sa hex mrezom tu staviti i objasniti sto se desava u 2D-u sa kondicijskim brojem interpolacije

5.13 Code parallelisation

6 Conclusions

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. [32] 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

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

Provjeri sto se desava sa simetrijom i displacement rubom pa nadopisi

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

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 $\mathcal{S}_P$:

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

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

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

Coefficients in tensor $\boldsymbol{c}_{\boldsymbol{xx}_N}$ 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:

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

where $\mathbf{c}_{\boldsymbol{xxx}_N}$ is a symmetric third-order coefficient tensor $\mathbf{c}_{\boldsymbol{xxx}_N} = \left[c_{xxx_N}, c_{xxy_N}, c_{xxz_N}, c_{xyy_N}, c_{xyz_N}, c_{xzz_N}, c_{yyy_N}, c_{yyz_N}, c_{yzz_N}, c_{zzz_N}\right]^{\mathrm{T}}$ 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:

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

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 Body Force for the Method of Manufactured Solutions Case

The body force for the manufactured solution case is [26]:

$$\boldsymbol{f}_b = \begin{pmatrix} \lambda\left[8a_y\pi^2\cos(4\pi x)\cos(2\pi y)\sin(\pi z) \right. \\ \quad +4a_z\pi^2\cos(4\pi x)\cos(\pi z)\sin(2\pi y) \\ \quad \left. -16a_x\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ +\mu\left[8a_y\pi^2\cos(4\pi x)\cos(2\pi y)\sin(\pi z) \right. \\ \quad +4a_z\pi^2\cos(4\pi x)\cos(\pi z)\sin(2\pi y) \\ \quad \left. -5a_x\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ \quad -32a_x\mu_\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ \\ \lambda\left[8a_x\pi^2\cos(4\pi x)\cos(2\pi y)\sin(\pi z) \right. \\ \quad +2a_z\pi^2\cos(2\pi y)\cos(\pi z)\sin(4\pi x) \\ \quad \left. -4a_y\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ +\mu\left[8a_x\pi^2\cos(4\pi x)\cos(2\pi y)\sin(\pi z) \right. \\ \quad +2a_z\pi^2\cos(2\pi y)\cos(\pi z)\sin(4\pi x) \\ \quad \left. -17a_y\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ \quad -8a_y\mu_\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ \\ \lambda\left[4a_x\pi^2\cos(4\pi x)\cos(\pi z)\sin(2\pi y) \right. \\ \quad +2a_y\pi^2\cos(2\pi y)\cos(\pi z)\sin(4\pi x) \\ \quad \left. -a_z\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ +\mu\left[4a_x\pi^2\cos(4\pi x)\cos(\pi z)\sin(2\pi y) \right. \\ \quad +2a_y\pi^2\cos(2\pi y)\cos(\pi z)\sin(4\pi x) \\ \quad \left. -20a_z\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ \quad -2a_z\mu_\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z) \end{pmatrix} \tag{E21}$$

Appendix F Meshes Used with the Method of Manufactured Solutions

References

[1] 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

[2] 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

[3] 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)

[4] 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)

[5] 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

[6] 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

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

[15] 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

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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)

[25] 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)

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

[27] 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)

[28] 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)

[29] Timoshenko, S., Goodier, J.: Theory of Elasticity. McGraw-Hill Company, Inc., New York

(1970)

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

[31] 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)

[32] 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