Assessing the potential of Jacobian-free Newton-Krylov methods for cell-centred finite volume solid mechanics

Philip Cardiff[1,2,3*], Ivan Batistić[4] and Željko Tuković[4]

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

[2]UCD Centre for Mechanics, University College Dublin, Ireland.

[3]SFI I-Form Centre, University College Dublin, Ireland.

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

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

Abstract

This study investigates the efficacy of Jacobian-free Newton-Krylov methods in finite-volume solid mechanics. Traditional Newton-based approaches require explicit Jacobian matrix formation and storage, which can be computationally expensive and memory-intensive. In contrast, Jacobian-free Newton-Krylov methods approximate the Jacobian's action using finite differences, combined with Krylov subspace solvers such as the generalised minimal residual method (GMRES), enabling seamless integration into existing segregated finite-volume frameworks without major code refactoring. This work benchmarks the performance of a compact-stencil Jacobian-free Newton-Krylov method against a conventional segregated approach on a suite of test cases, encompassing varying geometric dimensions, nonlinearities, dynamic responses, and material behaviours. Key metrics, including computational cost, memory efficiency, and robustness, are evaluated, along with the influence of preconditioning strategies and stabilisation scaling. Results show that the proposed Jacobian-free Newton-Krylov method outperforms the segregated approach in all linear and nonlinear elastic cases, achieving order-of-magnitude speedups in many instances; however, divergence is observed in elastoplastic cases, highlighting areas for further development. It is found that preconditioning choice significantly impacts performance: a LU direct solver is fastest in small to moderately-sized cases, while a multigrid method is more effective for larger problems. The findings demonstrate that Jacobian-free Newton-Krylov methods are promising for advancing finite-volume solid mechanics simulations, particularly for existing segregated frameworks where minimal modifications enable their adoption. The described implementations are available in the solids4foam toolbox for OpenFOAM, inviting the community to explore, extend, and compare these procedures.

Keywords: Jacobian-free Newton-Krylov, Finite volume method, GMRES, solids4foam, OpenFOAM

1 Introduction

Finite volume formulations for solid mechanics are heavily influenced by their fluid mechanics counterparts, favouring fully explicit [1–4] or segregated implicit [5–12] methods. Segregated approaches, where the governing equations are temporarily decomposed into scalar component equations, offer memory efficiency and simplicity of implementation, but the outer coupling Picard iterations often suffer from slow convergence. Explicit formulations are straightforward to implement and offer superior robustness but are only efficient for high-speed dynamics, where the physics requires small time increments. In contrast, the finite element community commonly employs Newton-Raphson-type solution algorithms, which necessitate repeated assembly of the Jacobian matrix and solution of the resulting block-coupled non-diagonally dominant linear system. A disadvantage of traditional Newton-based approaches is that they typically require explicit formation and storage of the Jacobian matrix, which can be computationally expensive and memory-intensive. A further disadvantage from a finite volume perspective is that extending existing code frameworks from segregated algorithms to coupled Newton-Raphson-type approaches is challenging in terms of the required assembly, storage, and solution of the resulting block-coupled system. In addition, the derivation of the true Jacobian matrix is non-trivial. Consequently, similar block-coupled solution finite volume methods are rare in the literature [13–15]. The motivation of the current work is to seek the robustness and efficiency of block-coupled Newton-Raphons approaches in a way that can be easily incorporated into existing segregated solution frameworks. To this end, the current article examines the efficacy of *Jacobian-free* Newton-Krylov methods, where the quadratic convergence of Newton methods can potentially be achieved without deriving, assembling and storing the exact Jacobian.

Jacobian-free Newton-Krylov methods circumvent the need for the Jacobian matrix by combining the Newton-Raphson method with Krylov subspace iterative linear solvers, such as the generalised minimal residual method (GMRES), and noticing that such Krylov solvers do not explicitly require the Jacobian matrix. Instead, only the action of the Jacobian matrix on a solution-type vector is required. The key step in Jacobian-free Newton-Krylov methods is the approximation of products between the Jacobian matrix and a vector using the finite difference method; that is

$$\boldsymbol{J}\boldsymbol{v} \approx \frac{\boldsymbol{F}(\boldsymbol{x} + \epsilon\boldsymbol{v}) - \boldsymbol{F}(\boldsymbol{x})}{\epsilon} \tag{1}$$

where $\boldsymbol{J}$ is the Jacobian matrix, $\mathbf{x}$ is the current solution vector (e.g. nodal displacements), $\mathbf{v}$ is a vector (e.g., from a Krylov subspace), and ϵ is a small scalar perturbation. With an appropriate choice of ϵ (balancing truncation and round-off errors), the characteristic quadratic convergence of Newton methods can be achieved without the Jacobian, hence the modifier *Jacobian-free*. This approach promises significant memory savings over Jacobian-based methods, especially for large-scale problems, but also potentially for execution time, with appropriate choice of solution components.

A crucial aspect of ensuring the efficiency and robustness of the Jacobian-free Newton-Krylov method is the choice of a suitable preconditioner for the Krylov iterations. This preconditioner is often derived from the exact Jacobian matrix in traditional Newton methods. However, the Jacobian-free approach does not allow direct access to the full Jacobian matrix, necessitating an

alternative strategy to approximate its action. To this end, and to extend existing segregated frameworks, this work proposes using a compact-stencil *approximate* Jacobian as the preconditioner. This approximate Jacobian corresponds to the matrix typically employed in segregated solid mechanics approaches; similar approaches are successful in fluid mechanics applications [16–25]; however, it is unclear if such an approach is suitable for solid mechanics - a question this work aims to answer. By leveraging this compact-stencil approximate Jacobian, it is aimed to effectively precondition the Krylov iterations, enhancing convergence while maintaining the memory and computational savings that define the Jacobian-free and segregated methods. Similarly, if such an approach is efficient, it would naturally fit into existing segregated frameworks, as existing matrix storage and assembly can be reused.

As noted by Knoll and Keyes [20], Jacobian-Free Newton–Krylov methods first appeared in the 1980s and early 1990s for the solution of ordinary and partial differential equations [26–29]. McHugh and Knoll [16] and co-workers demonstrated the potential of Jacobian-free Newton-Krylov approaches for the solution of steady, incompressible, Navier-Stokes problems using a staggered finite volume formulation. They found the GMRES linear solver to be faster than a conjugate solver, however, the true Jacobian matrix was still evalauted by finite differencing in the construction of the preconditioning matrix. Qin et al. [17] later extended Jacobian-Free Newton–Krylov methods to unsteady compressible Reynolds-averaged Navier–Stokes equations. Once again it was found that GMRES was more robust than a conjugate gradient linear solver. For preconditioning, a fully matrix-free approach was proposed, based on the approximate factorization procedure of Badcock and Gaitonde [30]. Geuzaine [18] examined the use of lower and higher-order discretisations for the preconditioning matrix for steady, compressible high Reynolds number flows. They found that the lower-order compact stencil preconditioner gave the best performance when combined with the GMRES linear solver and ILU(k) preconditioner. As is common with other Newton approaches, a pseudo transient algorithm and mesh sequencing strategy were used to improve the convergence. With the aim of exploiting mature segregated solution approaches, Pernice and Tocci [19] recast the SIMPLE segregated algorithm as a method which operators on residuals, allowing it to act as the preconditioner for a Jacobian-free Newton-Krylov approach. It was found that the SIMPLE-preconditioned Jacobian-free Newton-Krylov substantially accelerated convergence for the incompressible Navier–Stokes problems examined. GMRES was the adopted linear solver, where the importance of choosing an appropriate restart parameter was noted. They also found a multigrid approach to be more efficient than ILU(k) for preconditioning, where the implementation was based on the PETSc [31] package.

Over the subsequent two decades, Jacobian-free Newton-Krylov methods have seen increasing application in the field of finite volume computational fluid dynamics, albeit they are still far from widespread. The use of low-order approximate Jacobians for preconditioning has been established as an efficient and robust approach, e.g. [21, 22, 24, 25]. A particularly attractive application of the low-order preconditioning approach is for higher-order discretisations; that is, spatial discretisation of order greater than two. Nejat and Ollivier-Gooch [21] demonstrated such a higher order approach for steady, inviscid compressible flows, with up to fourth order accuracy. GMRES was the adopted linear solver with the ILU(k) preconditioner and fill-in values (k) ranging from 2 to 4. Nejat et al. [24] later applied the second-order variant of the approach to non-Newtonian flows and found Newton-GMRES with ILU(1) to be the most efficient combination. Like Vaassen et al. [22] and

many other authors, Nejat and co-workers [21, 24] proposed a special start-up phase to encourage the solution to efficiently reach the Newton method's domain of quadratic convergence. The use of a pseudo-transient approach combined with an adaptive time step (e.g., based on the convergence of the Newton iterations) is common. A further promotion for the benefit of Jacobian-free Newton-Krylov methods in fluid dynamics was provided by Lucas et al. [23], who demonstrated order of magnitude speedups over nonlinear multigrid methods. A particular benefit was the insensitivity of the Jacobian-free Newton-Krylov methods to changes in mesh aspect ratio, density, time step and Reynolds number. A final relevant point is the effect of numerical stabilisation (damping) on the convergence: Nishikawa et al. [32] demonstrated that a Jacobian-free Newton-Krylov approach was less sensitive to the magnitude of the damping and remained stable for lower values of damping compared with a segregated approach.

Although they are increasingly seen in finite volume computational fluid dynamics, Jacobian-free Newton-Krylov methods have yet to be used for finite volume solid mechanics. This article aims to address this point by assessing the efficacy of Jacobian-free Newton-Krylov cell-centred finite volume formulations applied to static, dynamic, linear and nonlinear solid mechanics problems. Particular focus is given to the use of a compact-stencil approximate Jacobian preconditioner, inspired from existing segregated solid mechanics procedures, and ease of implementation into an existing segregated finite volume framework – OpenFOAM [33]. The remainder of the paper is structured as follows: Section 2 summarises a typical solid mechanics mathematical model and its cell-centred finite volume discretisation. Section 3 presents the solution algorithms, starting with the classic segregated solution algorithm, followed by the proposed Jacobian-free Newton-Krylov solution algorithm. The performance of the proposed Jacobian-free Newton-Krylov approach is compared with the segregated approach on several varying benchmark cases in Section 4, where the effect of several factors are examined. Finally, the article ends with a summary of the main conclusions of the work.

2 Mathematical Model and Numerical Methods

2.1 Governing Equations

In this work, interest is restricted to Lagrangian formulations of the conservation of linear momentum. Assuming small strains, the linear geometry formulation is expressed in strong integral form as:

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

where Ω is the volume of an arbitrary body bounded by a surface Γ with outwards pointing normal $\boldsymbol{n}$. The density is ρ, $\boldsymbol{u}$ is the displacement vector, $\boldsymbol{\sigma}_s$ is the engineering (small strain) stress tensor, and $\boldsymbol{f}_b$ is a body force per unit volume, e.g., $\rho \boldsymbol{g}$, where $\boldsymbol{g}$ is gravity.

More generally, linear momentum conservation can be expressed in a nonlinear geometry form, which is suitable for finite strains. Two equivalent nonlinear geometry forms are common: the *total*

Lagrangian form:

$$\int_{\Omega_o} \rho_o \frac{\partial^2 \boldsymbol{u}}{\partial t^2} d\Omega_o = \oint_{\Gamma_o} \left(J \boldsymbol{F}^{-\mathrm{T}} \cdot \boldsymbol{n}_o \right) \cdot \boldsymbol{\sigma} \; d\Gamma_o + \int_{\Omega_o} \boldsymbol{f}_b \, d\Omega_o \tag{3}$$

and the *updated* Lagrangian form:

$$\int_{\Omega_u} \frac{\partial}{\partial t} \left(\rho_u \frac{\partial \boldsymbol{u}}{\partial t} \right) d\Omega_u = \oint_{\Gamma_u} \left(j \boldsymbol{f}^{-\mathrm{T}} \cdot \boldsymbol{n}_u \right) \cdot \boldsymbol{\sigma} \; d\Gamma_u + \int_{\Omega_u} \boldsymbol{f}_b \, d\Omega_u \tag{4}$$

where subscript o indicates quantities in the initial reference configuration, and subscript u indicates quantities in the updated configuration. The true (Cauchy) stress tensor is indicated by $\boldsymbol{\sigma}$. The deformation gradient is defined as $\boldsymbol{F} = \mathbf{I} + (\boldsymbol{\nabla u})^{\mathrm{T}}$ and its determinant as $J = \det(\boldsymbol{F})$. Similarly, the *relative* deformation gradient is given in terms of the displacement *increment* as $\boldsymbol{f} = \mathbf{I} + [\boldsymbol{\nabla}(\Delta \boldsymbol{u})]^{\mathrm{T}}$ and its determinant as $j = \det(\boldsymbol{f})$. The displacement increment is the change in displacement between the current time step and the previous time step when the time interval is discretised into a finite number of steps.

The governing equations are complemented by boundary conditions, with three types considered here: prescribed displacement, prescribed traction, and symmetry. The definition of the engineering stress ($\boldsymbol{\sigma}_s$) and true stress ($\boldsymbol{\sigma}$) in Equations 2, 3 and 4 is given by a chosen mechanical law. Five mechanical laws are considered in this work, as briefly outlined in Appendix A: linear elasticity (Hooke's law), three forms of hyperelasticity (St. Venant-Kirchhoff, neo-Hookean, and Guccione), and neo-Hookean J_2 hyperelastoplasticity.

2.2 Newton-Type Solution Methods

To facilitate the comparison between the classic segregated solution algorithm and the proposed Jacobian-free Newton-Krylov algorithm, the governing linear momentum conservation (Equations 2, 3 and 4) is expressed in the general form:

$$\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{0} \tag{5}$$

where $\boldsymbol{R}$ represents the *residual* (imbalance) of the equation, which is a function of the primary unknown field. For example, in the linear geometry case, the residual is given as

$$\boldsymbol{R}(\boldsymbol{u}) \;=\; \oint_{\Gamma} \boldsymbol{n} \cdot \boldsymbol{\sigma}_s(\boldsymbol{u}) \, d\Gamma + \int_{\Omega} \rho \boldsymbol{g} \, d\Omega - \int_{\Omega} \rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} \, d\Omega \;=\; \boldsymbol{0} \tag{6}$$

where the dependence of the stress tensor on the solution vector is made explicitly clear: $\boldsymbol{\sigma}_s(\boldsymbol{u})$.

In Newton-type methods, a Taylor expansion about a current point $\boldsymbol{u}_k$ can be used to solve Equation 5 [20]:

$$\boldsymbol{R}(\boldsymbol{u}_{k+1}) = \boldsymbol{R}(\boldsymbol{u}_k) \;+\; \boldsymbol{R}'(\boldsymbol{u}_k)(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) \;+\; \text{H.O.T.} = \boldsymbol{0} \tag{7}$$

Neglecting the higher-order terms (H.O.T.) yields the strict Newton method in terms of an iteration over a sequence of linear systems:

$$\begin{aligned} \boldsymbol{J}(\boldsymbol{u}_k)\delta\boldsymbol{u} &= -\boldsymbol{R}(\boldsymbol{u}_k), \\ \boldsymbol{u}_{k+1} &= \boldsymbol{u}_k + s\,\delta\boldsymbol{u}, \\ k &= 0, 1, ... \end{aligned} \tag{8}$$

where $\boldsymbol{J} \equiv \boldsymbol{R}' \equiv \partial\boldsymbol{R}/\partial\boldsymbol{u}$ is the Jacobian matrix. Starting the Newton procedure requires the specification of $\boldsymbol{u}_0$. The scalar $s > 0$ can be chosen to improve convergence, for example, using a line search or under-relaxation/damping procedure, and is equal to unity in the classic Newton-Raphson approach. Iterations are performed over this system until the residual $\boldsymbol{R}(\boldsymbol{u}_k)$ and solution correction $\delta\boldsymbol{u}$ are sufficiently small, with appropriate normalisation.

For problems with N scalar equations and N scalar unknowns, the residual $\boldsymbol{R}$ and solution $\boldsymbol{u}$ vectors have dimensions of $N \times 1$. The components of the $N \times N$ Jacobian are

$$J_{ij} = \frac{\partial R_i(\boldsymbol{u})}{\partial u_j} \tag{9}$$

The current work focusses on vector problems, where the governing momentum equation is formulated in terms of the unknown displacement solution vector. In this case, Equation 9 refers to the individual scalar components of the residual, solution, and Jacobian. That is, for 3-D analyses, the residual takes the form

$$\boldsymbol{R}(\boldsymbol{u}) = \{R_1^x, R_1^y, R_1^z, R_2^x, R_2^y, R_2^z, ..., R_n^z\} \tag{10}$$

and the solution takes the form

$$\boldsymbol{u} = \{u_1^x, u_1^y, u_1^z, u_2^x, u_2^y, u_2^z, ..., u_n^z\} \tag{11}$$

In practice, it is often more practical and efficient to form and store the residual, solution and Jacobian in a *blocked* manner, where the residual and solution can be considered as vectors of vectors. Similarly, the Jacobian can be formed in terms of sub-matrix block coefficients.

In the strict Newton procedure, the residuals converge at a quadratic rate when the current solution is close to the true solution; that is, the iteration error decreases proportionally to the square of the error at the previous iteration. Once the method gets sufficiently close to the true solution, the number of correct digits in the approximation roughly doubles with each iteration. However, quadratic convergence is only possible when using the exact Jacobian. In contrast, a quasi-Newton method uses an approximation to the Jacobian, sacrificing strict quadratic convergence in an attempt to produce an overall more computationally efficient procedure. From this perspective, the segregated solution algorithm commonly employed in finite volume solid mechanics can be viewed as a quasi-Newton method, where an approximate Jacobian replaces the exact Jacobian:

$$\tilde{\boldsymbol{J}}(\boldsymbol{u}_k)\ \delta\boldsymbol{u} = -\boldsymbol{R}(\boldsymbol{u}_k) \tag{12}$$

In this case, the approximate Jacobian $\tilde{\boldsymbol{J}}$ comes from the compact stencil discretisation of a simple diffusion (Laplacian) term. A benefit of this approach is that the inter-component coupling is removed from the Jacobian, allowing the solution of three smaller scalar systems rather than one larger vector system in 3-D (or two smaller systems in 2-D).

A fully explicit procedure can also be viewed from this perspective by selecting a *diagonal* approximate Jacobian $\tilde{\boldsymbol{D}}$, making the solution of the linear system trivial:

$$\tilde{\boldsymbol{D}}(\boldsymbol{u}_k)\ \delta\boldsymbol{u} = -\boldsymbol{R}(\boldsymbol{u}_k) \tag{13}$$

2.3 Cell-Centred Finite Volume Discretisation

In this work, a nominally second-order cell-centred finite volume discretisation is employed. The solution domain is discretised in both space and time. The total simulation period is divided into a finite number of time increments, denoted as Δt, and the discretised governing momentum equation is solved iteratively in a time-marching fashion. The spatial domain is partitioned into a finite set $\mathcal{P}$ of contiguous convex polyhedral cells, where each cell is denoted by $P \in \mathcal{P}$. The number of cells in the mesh is indicated by $|\mathcal{P}|$. A representative cell P is shown in Figure 1. The set of all faces of cell P is denoted by $\mathcal{F}_P$. This set is further subdivided into two disjoint subsets:

- **Internal Faces** ($\mathcal{F}_P^{\text{int}}$): Faces that are shared with neighbouring cells.
- **Boundary Faces** ($\mathcal{F}_P^{\text{bnd}}$): Faces that lie on the boundary of the spatial domain. The boundary faces of cell P are further classed into three disjoint sets, $\mathcal{F}_P^{\text{bnd}} := \mathcal{F}_P^{\text{disp}} \cup \mathcal{F}_P^{\text{trac}} \cup \mathcal{F}_P^{\text{symm}}$, representing boundary faces where displacement ($\mathcal{F}_P^{\text{disp}}$), traction ($\mathcal{F}_P^{\text{trac}}$) and symmetry ($\mathcal{F}_P^{\text{symm}}$) conditions are prescribed.

Each internal face $f_i \in \mathcal{F}_P^{\text{int}}$ corresponds to a neighbouring cell $N_{f_i} \in \mathcal{N}_P$, where $\mathcal{N}_P$ is the set of all neighbouring cells of P. The outward unit normal vector associated with an internal face f_i is denoted by $\mathbf{n}_{f_i}$, while the outward unit normal vector associated with a boundary face b_i is denoted by $\mathbf{n}_{b_i}$. The vector $\mathbf{d}_{f_i}$ connects the centroid of cell P with the centroid of the neighbouring cell N_{f_i}, whereas the vector $\mathbf{d}_{b_i}$ connects the centroid of cell P with the centroid of boundary face b_i. For convenience, we will also define the set of all faces of cell P excluding those on a traction boundary as $\mathcal{F}_P^{\text{non-trac}} := \mathcal{F}_P^{\text{int}} \cup \mathcal{F}_P^{\text{disp}} \cup \mathcal{F}_P^{\text{symm}}$.

The conservation equation (Equations 2, 3, or 4) is applied to each cell $\mathcal{P}$ and discretised in terms of the displacement at the centroid of the cell $\boldsymbol{u}_P$ and the displacements $\boldsymbol{u}_{N_{f_i}}$ at the centroids of the neighbouring cells. Proceeding with the discretisation, the volume integrals and surface integrals in the governing equation are approximated by algebraic equations as described below.

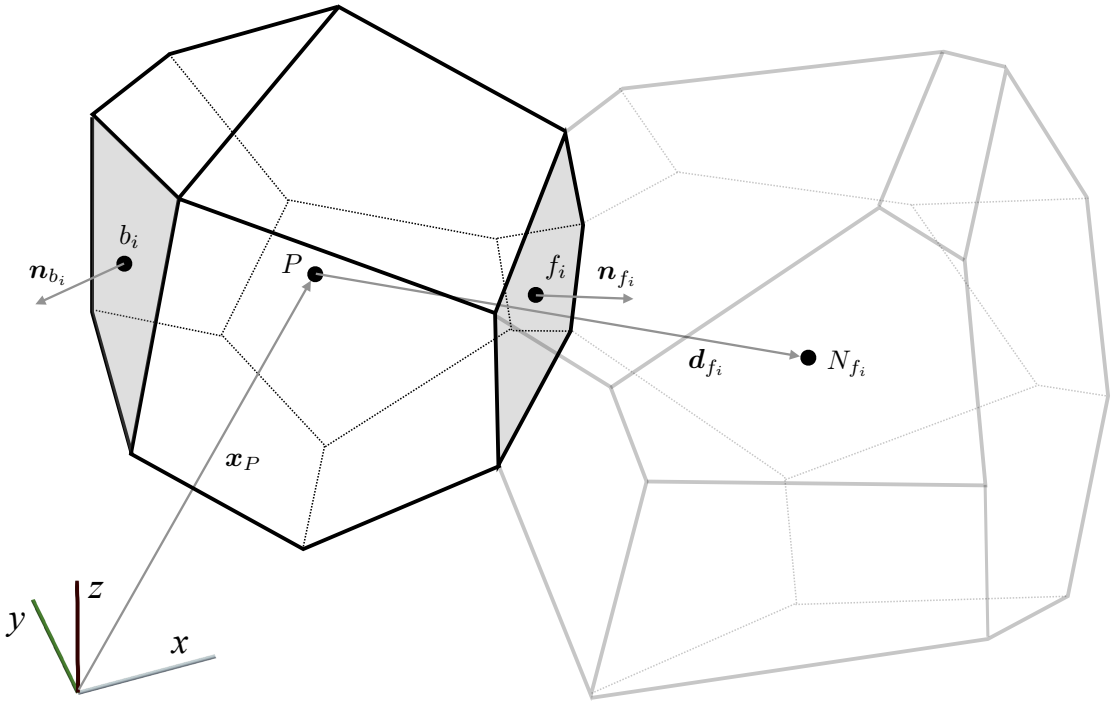


Fig. 1 Representative convex polyhedral cell P and neighbouring cell N_{f_i}, which share a face f_i.

2.3.1 Volume Integrals

To discretise the volume integrals, the integrand $\boldsymbol{\phi}$ is assumed to locally vary accoding to a truncated Taylor series expansion about the centroid of cell P:

$$\boldsymbol{\phi}(\boldsymbol{x}) \approx \boldsymbol{\phi}_P + (\boldsymbol{x} - \boldsymbol{x}_P) \cdot \nabla \boldsymbol{\phi}_P \tag{14}$$

where subscript P indicates a value at the centroid of the cell P. Consequently, volume integrals over a cell P can approximated to second order accuracy as

$$\begin{aligned} \int_{\Omega_P} \boldsymbol{\phi}\, d\Omega_P &\approx \int_{\Omega_P} [\boldsymbol{\phi}_P + (\boldsymbol{x} - \boldsymbol{x}_P) \cdot \nabla \boldsymbol{\phi}_P]\, d\Omega_P \\ &\approx \boldsymbol{\phi}_P \Omega_P \end{aligned} \tag{15}$$

where Ω_P is the volume of cell P and $\int_{\Omega_P} (\boldsymbol{x} - \boldsymbol{x}_P) d\Omega_P \equiv 0$ by definition of the cell centroid. This approximation corresponds to the midpoint rule and one point quadrature.

Using Equation 15, the inertia term (e.g. left-hand side term of Equation 2) becomes

$$\int_{\Omega_P} \rho \frac{\partial \boldsymbol{u}}{\partial t} d\Omega_P \approx \rho_P \left(\frac{\partial^2 \boldsymbol{u}}{\partial t^2} \right)_P \Omega_P \tag{16}$$

Similarly, the body force term (e.g. the second term on the right-hand side of Equation 2) becomes:

$$\int_{\Omega_P} \rho\, \boldsymbol{g}\, d\Omega_P \approx \rho_P\, \boldsymbol{g}\, \Omega_P \tag{17}$$

The discretisation of the acceleration in Equation 16 can be achieved using one of many finite difference schemes, e.g. first-order Euler, second-order backwards, or second-order Newmark-beta. In the current work, the second-order backwards (BDF2) scheme is used:

$$\begin{aligned}\left(\frac{\partial^2 \boldsymbol{u}_P}{\partial t^2}\right)_P &\approx \frac{3\boldsymbol{v}_P^{[t+1]} - 4\boldsymbol{v}_P^{[t]} + \boldsymbol{v}_P^{[t-1]}}{2\Delta t} \\ &\approx \frac{3\left(\dfrac{3\boldsymbol{u}_P^{[t+1]} - 4\boldsymbol{u}_P^{[t]} + \boldsymbol{u}_P^{[t-1]}}{2\Delta t}\right) - 4\boldsymbol{v}_P^{[t]} + \boldsymbol{v}_P^{[t-1]}}{2\Delta t}\end{aligned} \tag{18}$$

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

$$\boldsymbol{v}_P^{[t+1]} \approx \frac{3\boldsymbol{u}_P^{[t+1]} - 4\boldsymbol{u}_P^{[t]} + \boldsymbol{u}_P^{[t-1]}}{2\Delta t} \tag{19}$$

Consequently, the displacement and velocity at the two previous time steps must be stored, or alternatively the displacement at the previous four time steps.

2.3.2 Surface Integrals

The surface integral term can be discretised as using one-point quadrature at each face f_i as

$$\begin{aligned}\oint_{\Gamma_P} \boldsymbol{n}\cdot\boldsymbol{\sigma}\, d\Gamma_P &= \sum_{f_i\in\mathcal{F}_P} \int_{\Gamma_{f_i}} \boldsymbol{n}\cdot\boldsymbol{\sigma}\, d\Gamma_{f_i} \\ &\approx \sum_{f_i\in\mathcal{F}_P^{\text{int}}} \boldsymbol{\Gamma}_{f_i}\cdot\boldsymbol{\sigma}_{f_i} + \sum_{d_i\in\mathcal{F}_P^{\text{disp}}} \boldsymbol{\Gamma}_{d_i}\cdot\boldsymbol{\sigma}_P + \sum_{s_i\in\mathcal{F}_P^{\text{symm}}} \boldsymbol{\Gamma}_{s_i}\cdot\boldsymbol{\sigma}_{s_i} + \sum_{t_i\in\mathcal{F}_P^{\text{trac}}} |\boldsymbol{\Gamma}_{t_i}|\bar{\boldsymbol{T}}_{t_i}\end{aligned} \tag{20}$$

where Γ_P indicates the surface of cell P, and $\boldsymbol{\Gamma}_{f_i}$ indicates the area vector of face f_i, and vector $\bar{\boldsymbol{T}}_{t_i}$ represents the prescribed traction on the traction boundary face t_i.

It is noted that the displacement is assumed to vary linearly within each cell; hence, the displacement gradient and stress are constant within each cell. In the current work, a unique definition of stress $\boldsymbol{\sigma}_{f_i}$ at each cell face f_i is given as a weighted-averaged of values in the two cells ($\boldsymbol{\sigma}_P$, $\boldsymbol{\sigma}_{N_{f_i}}$) straddling the face [34]:

$$\boldsymbol{\sigma}_{f_i} = w_{f_i}\boldsymbol{\sigma}_P + (1 - w_{f_i})\boldsymbol{\sigma}_{N_{f_i}} \tag{21}$$

where the interpolation weight is defined as $w_{f_i} = (\boldsymbol{n}_{f_i}\cdot[\boldsymbol{x}_{N_{f_i}} - \boldsymbol{x}_{f_i}])/(\boldsymbol{n}_{f_i}\cdot[\boldsymbol{x}_{N_{f_i}} - \boldsymbol{x}_P])$; however, achieving second-order accuracy of the displacement field is independent of the value of the weights, and, for example, $w_{f_i} = 1/2$ would also be sufficient. Similarly, the stress $\boldsymbol{\sigma}_{s_i}$ at a symmetry boundary face is calculated as

$$\begin{aligned}\boldsymbol{\sigma}_{s_i} &= \frac{1}{2}\left(\boldsymbol{\sigma}_P + \boldsymbol{R}_{s_i}\cdot\boldsymbol{\sigma}_P\right) \\ &= \left(\mathbf{I} - \boldsymbol{n}_{s_i}\otimes\boldsymbol{n}_{s_i}\right)\cdot\boldsymbol{\sigma}_P\end{aligned} \tag{22}$$

where $\boldsymbol{R}_{s_i} \cdot \boldsymbol{\sigma}_P$ represents the mirror reflection of $\boldsymbol{\sigma}_P$ across the symmetry boundary face s_i. The reflection tensor is $\boldsymbol{R}_{s_i} = \mathbf{I} - 2\boldsymbol{n}_{s_i} \otimes \boldsymbol{n}_{s_i}$ [35], with $\boldsymbol{n}_{s_i}$ indicating the unit normal of the symmetry boundary face s_i. From Equation 22, it is clear that shear stresses are zero on a symmetry plane boundary face. Note from Equation 20 that the stress on displacement boundary faces is assumed to be equal to the stress $\boldsymbol{\sigma}_P$ at the centroid of cell P.

The cell-centred stress $\boldsymbol{\sigma}_P$ is calculated as a function of the displacement gradient according to the chosen mechanical law, where the cell-centred displacement gradients are determined using an inverse-distance-weighted first-neighbours least squares method [34],

$$\begin{aligned}(\boldsymbol{\nabla u})_P = & \sum_{f_i \in \mathcal{F}_P^{\text{int}}} \frac{\boldsymbol{G}_P^{-1} \cdot \boldsymbol{d}_{f_i}}{\boldsymbol{d}_{f_i} \cdot \boldsymbol{d}_{f_i}} \otimes \left(\boldsymbol{u}_{N_{f_i}} - \boldsymbol{u}_P\right) \\ & + \sum_{d_i \in \mathcal{F}_P^{\text{disp}}} \frac{\boldsymbol{G}_P^{-1} \cdot \boldsymbol{d}_{d_i}}{\boldsymbol{d}_{d_i} \cdot \boldsymbol{d}_{d_i}} \otimes \left(\boldsymbol{u}_{d_i} - \boldsymbol{u}_P\right) \\ & + \sum_{s_i \in \mathcal{F}_P^{\text{symm}}} \frac{\boldsymbol{G}_P^{-1} \cdot \boldsymbol{d}_{s_i}}{\boldsymbol{d}_{s_i} \cdot \boldsymbol{d}_{s_i}} \otimes \left(\boldsymbol{R}_{s_i} \cdot \boldsymbol{u}_P - \boldsymbol{u}_P\right)\end{aligned} \tag{23}$$

which is exact for linear functions. The vector $\boldsymbol{u}_{b_i}$ indicates the displacement at the centroid of boundary face b_i, while vector $\boldsymbol{d}_{d_i}$ connects the centroid of cell P to the centroid of displacement boundary face d_i. The quantity $\boldsymbol{R}_{s_i} \cdot \boldsymbol{u}_P$ represents the mirror reflection of $\boldsymbol{u}_P$ across the symmetry boundary face s_i. The vector $\boldsymbol{d}_{s_i}$ connects the centroid $\boldsymbol{x}_P$ of cell P with its mirror reflection $\boldsymbol{R}_{s_i} \cdot \boldsymbol{x}_P$ through boundary face s_i. Traction boundary faces are excluded in Equation 23, as the displacement is unknown there; this is in contrast to the default approach in OpenFOAM [36]. The $\boldsymbol{G}_P$ tensor for cell P is calculated as

$$\boldsymbol{G}_P = \sum_{f_i \in \mathcal{F}_P^{\text{int}}} \frac{\boldsymbol{d}_{f_i} \otimes \boldsymbol{d}_{f_i}}{\boldsymbol{d}_{f_i} \cdot \boldsymbol{d}_{f_i}} + \sum_{d_i \in \mathcal{F}_P^{\text{disp}}} \frac{\boldsymbol{d}_{d_i} \otimes \boldsymbol{d}_{d_i}}{\boldsymbol{d}_{d_i} \cdot \boldsymbol{d}_{d_i}} + \sum_{s_i \in \mathcal{F}_P^{\text{symm}}} \frac{\boldsymbol{d}_{s_i} \otimes \boldsymbol{d}_{s_i}}{\boldsymbol{d}_{s_i} \cdot \boldsymbol{d}_{s_i}} \tag{24}$$

where the $\boldsymbol{G}_P^{-1} \cdot \boldsymbol{d}/(\boldsymbol{d} \cdot \boldsymbol{d})$ vectors are purely a function of the mesh and can be computed once and stored. Equation 23 approximates the cell-centre gradients to at least a first-order accuracy, increasing to second-order accuracy on certain smooth grids [37]; first-order accurate gradients are sufficient to preserve second-order accuracy of the cell-centre displacements.

As noted, boundary conditions are enforced through the discretised surface integral terms at the boundary faces. If required, the displacement $\boldsymbol{u}_{t_i}$ on a traction boundary face t_i can be calculated by extrapolation from the centre of cell P as

$$\boldsymbol{u}_{t_i} = \boldsymbol{u}_P + \boldsymbol{d}_{t_i} \cdot (\boldsymbol{\nabla u})_P \tag{25}$$

where $\boldsymbol{d}_{t_i}$ represents the vector from the centroid of cell P to the centroid of the traction boundary face t_i. Similarly, if required, the displacement $\boldsymbol{u}_{s_i}$ at a symmetry plane face s_i is calculated using the same approach as Equation 22:

$$\begin{aligned}\boldsymbol{u}_{s_i} &= \frac{1}{2}\left(\boldsymbol{u}_P + \boldsymbol{R}_{s_i} \cdot \boldsymbol{u}_P\right) \\ &= \left(\mathbf{I} - \boldsymbol{n}_{s_i} \otimes \boldsymbol{n}_{s_i}\right) \cdot \boldsymbol{u}_P\end{aligned} \tag{26}$$

2.3.3 Rhie-Chow Stabilisation

To quell zero-energy solution modes (i.e. checkerboarding oscillations), a Rhie-Chow-type stabilisation term [38] is added to the residual (Equation 5). The Rhie-Chow stabilisation term $\mathcal{D}_P^{\text{Rhie-Chow}}$ for a cell P takes the following form:

$$\begin{aligned}\mathcal{D}_P^{\text{Rhie-Chow}} = &\sum_{f_i \in \mathcal{F}_P^{\text{int}}} \alpha \bar{K}_{f_i} \left[|\boldsymbol{\Delta}_{f_i}| \frac{\boldsymbol{u}_{N_{f_i}} - \boldsymbol{u}_P}{|\boldsymbol{d}_{f_i}|} - \boldsymbol{\Delta}_{f_i} \cdot (\boldsymbol{\nabla u})_{f_i} \right] |\boldsymbol{\Gamma}_{f_i}| \\ &+ \sum_{d_i \in \mathcal{F}_P^{\text{disp}}} \alpha \bar{K}_{d_i} \left[|\boldsymbol{\Delta}_{d_i}| \frac{\bar{\boldsymbol{u}}_{d_i} - \boldsymbol{u}_P}{|\boldsymbol{d}_{d_i}|} - \boldsymbol{\Delta}_{d_i} \cdot (\boldsymbol{\nabla u})_P \right] |\boldsymbol{\Gamma}_{d_i}| \\ &+ \sum_{s_i \in \mathcal{F}_P^{\text{symm}}} \alpha \bar{K}_{s_i} \left[|\boldsymbol{\Delta}_{s_i}| \frac{\boldsymbol{R}_{s_i} \cdot \boldsymbol{u}_P - \boldsymbol{u}_P}{|\boldsymbol{d}_{s_i}|} - \boldsymbol{\Delta}_{s_i} \cdot (\boldsymbol{\nabla u})_{s_i} \right] |\boldsymbol{\Gamma}_{s_i}| \end{aligned} \tag{27}$$

where $\alpha > 0$ is a user-defined parameter for globally scaling the amount of stabilisation. Parameter $\bar{K}$ is a stiffness-type parameter that gives the stabilisation an appropriate scale and dimension. Here, $\bar{K} = \frac{4}{3}\mu + \kappa = 2\mu + \lambda$ following previous work [8, 10, 39], where μ is the shear modulus (first Lamé parameter), κ is the bulk modulus, and λ is the second Lamé parameter. Vector $\bar{\boldsymbol{u}}_{d_i}$ represents the prescribed displacement at the centroid of displacement boundary face d_i. The quantities $\boldsymbol{\Delta} = \boldsymbol{d}/\boldsymbol{d}\cdot\boldsymbol{n}$ are termed the *over-relaxed orthogonal* vectors [34]. The displacement gradient $(\boldsymbol{\nabla u})_{f_i}$ at the internal face f_i is calculated by interpolation from adjacent cell centres (like in Equation 21). Similarly, the displacement gradient $(\boldsymbol{\nabla u})_{s_i}$ at a symmetry boundary face s_i is averaged from cell P and its mirror reflection across face s_i, as in Equation 22. Note that the displacement gradient at the displacement boundary d_i is assumed to be equal to the displacement gradient at the centroid of cell P. In addition, no stabilisation term is applied on a traction boundary face t_i.

Within the square braces on the right-hand side of Equation 27, the first terms represent a compact stencil (two-node) approximation of the face normal gradient, while the second terms represent a larger stencil approximation. These two terms cancel out in the limit of mesh refinement (or if the solution varies linearly); otherwise, they produce a stabilisation effect that tends to smooth the solution fields. The magnitude of the stabilisation is shown in Appendix B to reduce at a second-order rate (not a third-order rate as stated previously [7]), and hence does not affect the overall scheme's second-order accuracy. It is informative to note that the first term in the square bracket in Equation 27 can be expressed, assuming $(\boldsymbol{d}_{f_i} \cdot \boldsymbol{n}_{f_i}) > 0$, as

$$\begin{aligned}\alpha \left[|\boldsymbol{\Delta}_{f_i}| \frac{\boldsymbol{u}_{N_{f_i}} - \boldsymbol{u}_P}{|\boldsymbol{d}_{f_i}|} - \boldsymbol{\Delta}_{f_i} \cdot (\boldsymbol{\nabla u})_{f_i} \right] &= \alpha \left| \frac{\boldsymbol{d}_{f_i}}{\boldsymbol{d}_{f_i} \cdot \boldsymbol{n}_{f_i}} \right| \frac{\boldsymbol{u}_{N_{f_i}} - \boldsymbol{u}_P}{|\boldsymbol{d}_{f_i}|} - \alpha \frac{\boldsymbol{d}_{f_i}}{\boldsymbol{d}_{f_i} \cdot \boldsymbol{n}_{f_i}} \cdot (\boldsymbol{\nabla u})_{f_i} \\ &= \frac{\alpha}{\boldsymbol{d}_{f_i} \cdot \boldsymbol{n}_{f_i}} \left[\left(\boldsymbol{u}_{N_{f_i}} - \boldsymbol{u}_P \right) - \boldsymbol{d}_{f_i} \cdot (\boldsymbol{\nabla u})_{f_i} \right] \end{aligned} \tag{28}$$

Noting that $(\boldsymbol{\nabla u})_{f_i} = \left[(\boldsymbol{\nabla u})_{N_{f_i}} + (\boldsymbol{\nabla u})_P \right] / 2$, Equation 28 can be expressed as

$$\alpha \left[|\boldsymbol{\Delta}_{f_i}| \frac{\boldsymbol{u}_{N_{f_i}} - \boldsymbol{u}_P}{|\boldsymbol{d}_{f_i}|} - \boldsymbol{\Delta}_{f_i} \cdot (\boldsymbol{\nabla u})_{f_i} \right] = \frac{\alpha}{\boldsymbol{d}_{f_i} \cdot \boldsymbol{n}_{f_i}} \left(\boldsymbol{u}^*_{N_{f_i}} - \boldsymbol{u}^*_P \right) \tag{29}$$

where

$$\boldsymbol{u}_P^* = \boldsymbol{u}_P + (\boldsymbol{d}_{f_i}/2) \cdot (\boldsymbol{\nabla u})_P \tag{30}$$

$$\boldsymbol{u}_{N_{f_i}}^* = \boldsymbol{u}_{N_{f_i}} - (\boldsymbol{d}_{f_i}/2) \cdot (\boldsymbol{\nabla u})_{N_{f_i}} \tag{31}$$

So, this Rhie-Chow stabilisation can be seen to take the form of a *jump* term; that is, it is a function of the jump in solution between neighbouring cells. The two states, $\boldsymbol{u}_P^*$ and $\boldsymbol{u}_{N_{f_i}}^*$, are evaluated halfway between P and N_{f_i}, and it may not be at a face centre; however, as noted by Nishikawa [40], this will not affect the second-order accuracy of the method; consequently, the user is free to choose the value of α for stabilisation purposes.

3 Solution Algorithms

3.1 Segregated Solution Algorithm

include alpha and check its effect The classic segregated solution algorithm can be viewed as a quasi-Newton method, where an approximate Jacobian is derived from the inertia term and a compact-stencil discretisation of a diffusion term:

$$\tilde{\boldsymbol{J}} = \frac{\partial}{\partial \boldsymbol{u}} \left[\oint_{\Gamma_P} \bar{K}\, \boldsymbol{n} \cdot \boldsymbol{\nabla u}\, d\Gamma_P \; - \; \int_{\Omega_P} \rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2}\, d\Omega_P \right] \tag{32}$$

The inertia term is discretised as described in Equations 16 and 18, while the diffusion term is discretised in the same manner as the compact-stencil component of the Rhie-Chow stabilisation term:

$$\begin{aligned} \oint_{\Gamma_P} \bar{K}\, \boldsymbol{n} \cdot \boldsymbol{\nabla u}\, d\Gamma_P \approx & \sum_{f_i \in \mathcal{F}_P^{\text{int}}} \bar{K}\, |\boldsymbol{\Delta}_{f_i}| \, \frac{\boldsymbol{u}_{N_{f_i}} - \boldsymbol{u}_P}{|\boldsymbol{d}_{f_i}|} \, |\boldsymbol{\Gamma}_{f_i}| \\ & + \sum_{d_i \in \mathcal{F}_P^{\text{disp}}} \bar{K}\, |\boldsymbol{\Delta}_{d_i}| \, \frac{\bar{\boldsymbol{u}}_{d_i} - \boldsymbol{u}_P}{|\boldsymbol{d}_{d_i}|} \, |\boldsymbol{\Gamma}_{d_i}| \\ & + \sum_{s_i \in \mathcal{F}_P^{\text{symm}}} \bar{K}\, |\boldsymbol{\Delta}_{s_i}| \, \frac{\boldsymbol{R}_{s_i} \cdot \boldsymbol{u}_P - \boldsymbol{u}_P}{|\boldsymbol{d}_{s_i}|} \, |\boldsymbol{\Gamma}_{s_i}| \end{aligned} \tag{33}$$

When a diffusion term is typically discretised using the cell-centre finite volume method, non-orthogonal corrections are included in a deferred correction manner to preserve the order of accuracy on distorted grids. However, in the current quasi-Newton method form, the approximate Jacobian's exact value does not affect the final converged solution, but only the convergence behaviour. Consequently, non-orthogonal corrections are not included in the approximate Jacobian here. However, grid distortion is appropriately accounted for in the calculation of the residual. Nonetheless, as a result, it is expected that the convergence behaviour of the segregated approach may degrade as mesh non-orthogonality increases.

The linearised system (Equation 12) is formed for each cell in the domain, resulting in a system of algebraic equations:

$$\left[\tilde{\boldsymbol{J}}\right]\ [\delta \boldsymbol{u}] = -\left[\boldsymbol{R}(\boldsymbol{u}_k)\right] \tag{34}$$

where $\left[\tilde{\boldsymbol{J}}\right]$ is a symmetric, $M \times M$ stiffness matrix, where $M = 3|\mathcal{P}|$ in 3-D and $M = 2|\mathcal{P}|$ in 2-D. If $\Delta t < \infty$ or $\mathcal{F}_P^{\text{disp}} \neq \emptyset$, matrix $\left[\tilde{\boldsymbol{J}}\right]$ is strongly diagonally dominant; otherwise, it is weakly diagonally dominant. The block (3×3 for 3-D, 2×2 for 2-D) diagonal coefficient for cell P (row P, column P) can be expressed as

$$\begin{aligned}\left[\tilde{\boldsymbol{J}}\right]_{PP} = &- \sum_{f_i \in \mathcal{F}_P^{\text{int}}} \bar{K} \frac{|\boldsymbol{\Delta}_{f_i}|}{|\boldsymbol{d}_{f_i}|} |\boldsymbol{\Gamma}_{f_i}| \mathbf{I} \quad - \sum_{d_i \in \mathcal{F}_P^{\text{disp}}} \bar{K} \frac{|\boldsymbol{\Delta}_{d_i}|}{|\boldsymbol{d}_{d_i}|} |\boldsymbol{\Gamma}_{d_i}| \mathbf{I} \\ &- \sum_{s_i \in \mathcal{F}_P^{\text{symm}}} \bar{K} \frac{|\boldsymbol{\Delta}_{s_i}|}{|\boldsymbol{d}_{s_i}|} |\boldsymbol{\Gamma}_{s_i}| \boldsymbol{n}_{s_i} \otimes \boldsymbol{n}_{s_i} \quad - \quad \frac{9}{4} \frac{\rho_P \Omega_P}{\Delta t^2} \mathbf{I}\end{aligned} \tag{35}$$

while the off-diagonal coefficients (row P, column Q) can be expressed as

$$\left[\tilde{\boldsymbol{J}}\right]_{PQ} = \bar{K} \frac{|\boldsymbol{\Delta}_{f_{PQ}}|}{|\boldsymbol{d}_{f_{PQ}}|} |\boldsymbol{\Gamma}_{f_{PQ}}| \mathbf{I} \tag{36}$$

where subscript PQ indicates a quantity associated with the internal face f shared between cells P and Q.

By design, matrix $\left[\tilde{\boldsymbol{J}}\right]$ contains no inter-component coupling, i.e., each block coefficient is diagonal; consequently, three equivalent smaller linear systems can be formed in 3-D (or two linear systems in 2-D) and solved for the Cartesian components of the displacement correction, e.g.

$$\left[\tilde{\boldsymbol{J}}_x\right]\ [\Delta \boldsymbol{u}_x] = -\left[\mathcal{R}_x(\boldsymbol{u}_k)\right] \tag{37}$$

$$\left[\tilde{\boldsymbol{J}}_y\right]\ [\Delta \boldsymbol{u}_y] = -\left[\mathcal{R}_y(\boldsymbol{u}_k)\right] \tag{38}$$

$$\left[\tilde{\boldsymbol{J}}_z\right]\ [\Delta \boldsymbol{u}_z] = -\left[\mathcal{R}_z(\boldsymbol{u}_k)\right] \tag{39}$$

where $\bullet_x$ represents the components in the x direction, $\bullet_y$ represents the components in the y direction, and $\bullet_z$ represents the components in the z direction. Matrices $\left[\tilde{\boldsymbol{J}}_x\right]$, $\left[\tilde{\boldsymbol{J}}_y\right]$ and $\left[\tilde{\boldsymbol{J}}_z\right]$ have the size $|\mathcal{P}| \times |\mathcal{P}|$. An additional benefit from a memory and assembly perspective, is that matrices $\left[\tilde{\boldsymbol{J}}_x\right]$, $\left[\tilde{\boldsymbol{J}}_y\right]$ and $\left[\tilde{\boldsymbol{J}}_z\right]$ are identical, except for the effects from including boundary conditions ($\mathcal{F}_P^{\text{symm}}$ terms). From an implementation perspective, this allows a single scalar matrix to be formed and stored, where the boundary condition contributions are inserted before solving a particular component.

The *inner* linear sparse systems (Equations 37, 38 and 39) can be solved using any typical direct or iterative linear solver approach; however, an incomplete Cholesky pre-conditioned conjugate gradient method [41] is often preferred as the diagonally dominant characteristic leads to good convergence characteristics. Algebraic multigrid can also be used to accelerate convergence.

In literature, the segregated solution algorithm is typically formulated in terms of the total displacement vector (or its difference between time steps) as the primary unknown; in contrast, in the quasi-Newton interpretation presented here, the primary unknown is the correction to the displacement vector, which goes to zero at convergence. Nonetheless, both approaches are equivalent and neither formulation might be expected to display superior performance.

3.2 Jacobian-free Newton-Krylov Algorithm

As noted in the introduction, the Jacobian-free Newton-Krylov method avoids the need to explicitly construct the Jacobian matrix by approximating its action on a solution vector using the finite difference method, repeated here:

$$\boldsymbol{J}\boldsymbol{v} \approx \frac{\boldsymbol{F}(\boldsymbol{x}+\epsilon\boldsymbol{v}) - \boldsymbol{F}(\boldsymbol{x})}{\epsilon} \tag{40}$$

The derivation of this approximation can be shown for a 2×2 system as [20]:

$$\begin{aligned}
\frac{\mathbf{F}(\mathbf{x}+\epsilon\mathbf{v}) - \mathbf{F}(\mathbf{x})}{\epsilon} &= \begin{pmatrix} \dfrac{F_1(x_1+\epsilon v_1, x_2+\epsilon v_2) - F_1(x_1,x_2)}{\epsilon} \\ \dfrac{F_2(x_1+\epsilon v_1, x_2+\epsilon v_2) - F_2(x_1,x_2)}{\epsilon} \end{pmatrix} \\
&\approx \begin{pmatrix} \dfrac{F_1(x_1,x_2) + \epsilon v_1 \dfrac{\partial F_1}{\partial u_1} + \epsilon v_2 \dfrac{\partial F_1}{\partial u_2} - F_1(x_1,x_2)}{\epsilon} \\ \dfrac{F_2(x_1,x_2) + \epsilon v_1 \dfrac{\partial F_2}{\partial u_1} + \epsilon v_2 \dfrac{\partial F_2}{\partial u_2} - F_2(x_1,x_2)}{\epsilon} \end{pmatrix} \\
&\approx \begin{pmatrix} v_1 \dfrac{\partial F_1}{\partial u_1} + v_2 \dfrac{\partial F_1}{\partial u_2} \\ v_1 \dfrac{\partial F_2}{\partial u_1} + v_2 \dfrac{\partial F_2}{\partial u_2} \end{pmatrix} \\
&\approx \boldsymbol{J}\boldsymbol{v}
\end{aligned} \tag{41}$$

where a first-order truncated Taylor series expansion about $\boldsymbol{u}$ was used to approximate $\boldsymbol{F}(\boldsymbol{x}+\epsilon\boldsymbol{v})$. As noted above, choosing an appropriate value for ϵ is non-trivial, and care must be taken to balance truncation error (reduced by decreasing ϵ) and round-off error (increased by decreasing ϵ).

The purpose of preconditioning the Jacobian-free Newton-Krylov method is to reduce the number of inner linear solver iterations. In the current work, the GMRES linear solver is used for the inner system. Using right preconditioning, the finite difference approximation of Equation 40 becomes

$$\boldsymbol{J}\boldsymbol{P}^{-1}\boldsymbol{v} \approx \frac{\boldsymbol{F}(\boldsymbol{x}+\epsilon\boldsymbol{P}^{-1}\boldsymbol{v}) - \boldsymbol{F}(\boldsymbol{x})}{\epsilon} \tag{42}$$

where $\boldsymbol{P}$ is the preconditioning matrix or process. In practice, only the action of $\boldsymbol{P}^{-1}$ on a vector is required, and $\boldsymbol{P}^{-1}$ need not be explicitly formed. Concretely, the preconditioner needs to approximately solve the linear system $\boldsymbol{y} = \boldsymbol{P}^{-1}\boldsymbol{v}$.

In the current work, we propose to use the compact-stencil approximate Jacobian from the segregated algorithm $\tilde{\boldsymbol{J}}$ as the preconditioning matrix $\boldsymbol{P}$ for the preconditioned Jacobian-free Newton-Krylov method. This preconditioning approach can be considered as a "physics-based" preconditioner in the classifications of Knoll and Keyes [20]. The approach is conceptually similar to an approximation of the Jacobian of a higher-order large-stencil scheme by a compact-stencil lower-order scheme. A benefit of the proposed approach is that existing segregated frameworks can re-use their existing matrix assembly and storage implementations. Concretely, the Jacobian-free Newton-Krylov method requires only a procedure for forming this preconditioning matrix and a procedure for explicitly evaluating the residual. Both routines are easily implemented – and are likely already available – in an existing segregated framework. The only additional required procedure is an interface to an existing Jacobian-free Newton-Krylov implementation. In the current work, the PETSc package [31] is used as the nonlinear solver, driven by a finite volume solver implemented in the solids4foam toolbox [11, 39] for OpenFOAM toolbox [33] (version OpenFOAM-v2312).

Several preconditioners are available in the literature, incomplete Cholesky or ILU(k) being popular; however, multigrid methods offer the greatest potential for large-scale problems. As noted by Knoll and Keyes [20], algorithmic simplifications within a multigrid procedure, which may result in loss of convergence for multigrid as a solver, have a much weaker effect when multigrid is used as a preconditioner. In this work, three preconditioners are considered:

1. ILU(k): incomplete lower-upper decomposition with fill-in k.
2. Multigrid: the HYPRE Boomerang [42] multigrid implementation.
3. LU: the MUMPS [43, 44] lower-upper decomposition direct solver.

A challenge with Newton-type methods, including Jacobian-free versions, is poor convergence when far from the true solution, and divergence is often a real possibility. Globalisation refers to steering an initial solution towards the quadratic convergence range of the Newton method. Several strategies are possible, and it is common to combine approaches [20]. In the current work, a line search procedure is used to select the s parameter in the solution update step (the second line in Equations 8). Line search methods assume the Newton update direction is correct and aim to find a scalar $s > 0$ that decreases the residual $\boldsymbol{R}(\boldsymbol{u}_k + s\delta\boldsymbol{u}) < \boldsymbol{R}(\boldsymbol{u}_k)$. In addition to a line search approach, a *transient continuation* globalisation approach is used in the current work, where the displacement $\boldsymbol{u}_P$ for cell P at time $t + \Delta t$ is predicted at the start of a new time (or loading) step, based on a truncated second-order Taylor series expansion:

$$\boldsymbol{u}_P^{[t+\Delta t]} = \boldsymbol{u}_P^{[t]} + \Delta t\, \boldsymbol{v}_P^{[t]} + \frac{1}{2}\Delta t^2 \left(\frac{\partial \boldsymbol{v}}{\partial t}\right)_P^{[t]} \tag{43}$$

where $\boldsymbol{v}_P^{[t]}$ is the velocity of cell P at time t, and $\left(\frac{\partial \boldsymbol{v}}{\partial t}\right)_P^{[t]}$ is the acceleration. In this way, for highly nonlinear problems, the user can decrease the time step size Δt as a globalisation approach to improve the performance of the Newton method. The predictor step in Equation 43 has been chosen to be consistent with the discretisation of the inertia term (Equation 18).

A final comment on the Jacobian-free Newton-Krylov solution algorithm is the potential importance of *oversolving*. Here, oversolving refers to solving the linear system to too tight a tolerance during the early Newton iterations, essentially wasting time when the solution is far from the true

solution. In addition, some authors [20] have shown Newton convergence to be worse when the earlier iterations are solved to too tight a tolerance. The concept of oversolving also applies to segregated solution procedures and has been well-known since the early work of Demirdžić and co-workers [7], where the residuals are typically reduced by one order of magnitude in the inner linear system. In the current work, the residual is reduced by one order of magnitude in the segregated approach and three orders in the Jacobian-free Newton-Krylov approach.

4 Test Cases

This section compares the performance of the proposed Jacobian-free Newton-Krylov solution approach with the segregated approach on several benchmark cases. The cases have been chosen to exhibit a variety of characteristics in terms of

- Geometric dimension (2-D vs. 3-D),
- Geometric nonlinearity (small strain vs. large strain),
- Geometric complexity (basic geometric shapes vs. complex geometry),
- Statics vs. dynamics, and
- Material behaviour (elasticity, elastoplasticity, hyperelasticity).

In all cases, the residuals are dropped by six orders of magnitude. Although the presented analyses aim to be extensive, several common features of modern solid mechanics procedures are left for future work, including nonlinear boundary conditions (e.g., contact, fracture) and mixed formulations (e.g., incompressibility).

The remainder of this section is structured as follows: The proposed discretisation's accuracy and order are assessed on several test cases of varying dimensions and phenomena. Subsequently, the efficiency of the Jacobian-free Newton-Krylov approach is compared with the standard segregated procedure in terms of time, memory and robustness. Finally, the effect of preconditioning strategy and stabilisation scaling is examined.

4.1 Testing the Accuracy and Order of Accuracy

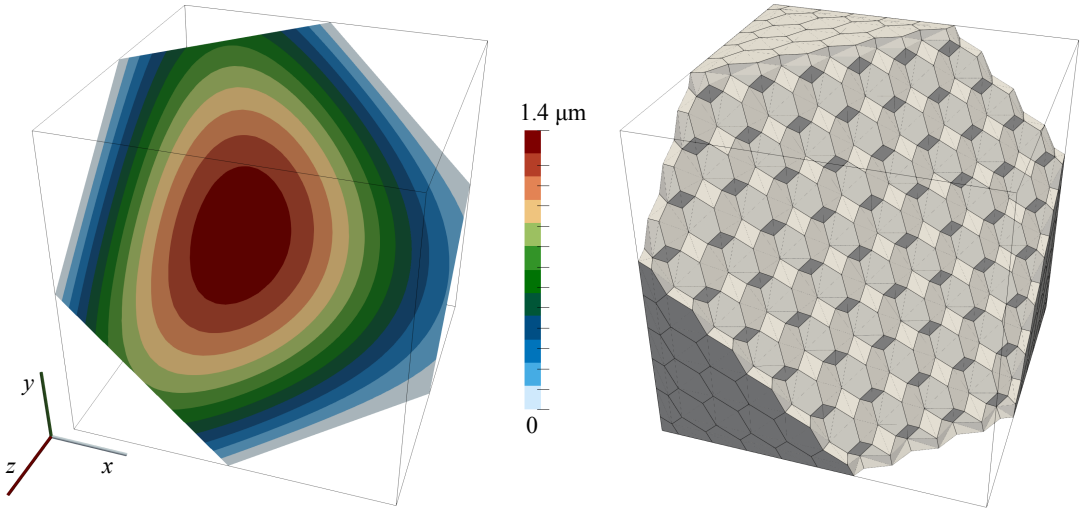
This section solely focuses on assessing the accuracy and order of accuracy of the discretisation. Assessment of the computational efficiency regarding time and memory requirements is left to the subsequent sections. In all cases where convergence is achieved, the differences between the Jacobian-free Newton-Raphson and segregated approaches are minimal; this is expected since both approaches use the same discretisation, and the solution tolerances have been chosen to ensure the iteration errors are small. Consequently, the results presented below (Section 4.1) have been solely generated using the Jacobian-free Newton-Raphson solution algorithm, unless stated otherwise.

Case 1: Order Verification via the Manufactured Solution Procedure

The first test case consists of a $0.2 \times 0.2 \times 0.2$ m cube with linear elastic ($E = 200$ GPa, $\nu = 0.3$) properties. A manufactured solution for displacement (Figure 2(a)) is employed of the form [45]

$$\boldsymbol{u} = \begin{pmatrix} a_x \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ a_y \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ a_z \sin(4\pi x)\sin(2\pi y)\sin(\pi z) \end{pmatrix} \tag{44}$$

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 2) is given in Appendix C.



(a) Magnitude of the manufactured displacement solution (b) Regular polyhedral mesh with 1 000 cells

Fig. 2 A cut plane through the cube case geometry showing the magnitude of the manufactured displacement solution (left) and a polyhedral mesh (right). The cut plane passes through the centre of the cube and has the unit normal $\boldsymbol{n} = (1/\sqrt{3} \quad 1/\sqrt{3} \quad 1/\sqrt{3})$.

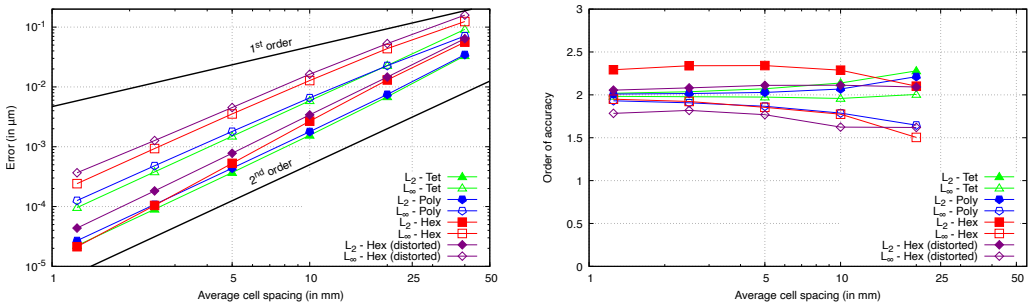
The manufactured displacement solution is applied at the domain's boundaries (prescribed displacements), and inertial effects are neglected. Three mesh types are examined: (i) *regular* tetrahedra, (ii) *regular* polyhedra (shown in Figure 2(b)), (iii) *regular* hexahedra, and (iv) *distorted* hexahedra. The second coarsest meshes are shown in Appendix D. The tetrahedral meshes are created using Gmsh [46], while the polyhedral meshes are created by converting the tetrahedral meshes to their dual polyhedral representations using the OpenFOAM `polyDualMesh` utility. The regular hexahedral meshes are created using the OpenFOAM `blockMesh` utility, and the distorted hexahedral meshes are created by perturbing the points of the regular hexahedra by a random vector of magnitude less than 0.3 times the local minimum edge length. Starting from an initial mesh spacing of 0.04 m, six meshes of each type are created by successively halving the spacing. The cell numbers for the hexahedral and polyhedral meshes are 125, 1 000, 8 000, 64 000, 512 000, and 4 096 000, while the tetrahedral mesh cell counts are 384, 4 374, 41 154, 355 914, 2 958 234, and 24 118 074. For the same average cell width, the cell counts show that the tetrahedral meshes have more cells by a factor of 3 to 6. Table 1 lists the average and maximum face non-orthogonality for

the meshes of different cell types, where face non-orthogonality is defined as the angle made by the vector between the two adjacent cell centres across the common face and the face normal.

Cell type	Average	Maximum
Hexahedra	0	0
Tetrahedra	32.71	35.26
Polyhedra	30.65	38.45

Table 1 Average and maximum face non-orthogonality (in °) for the meshes of different cell types and sizes

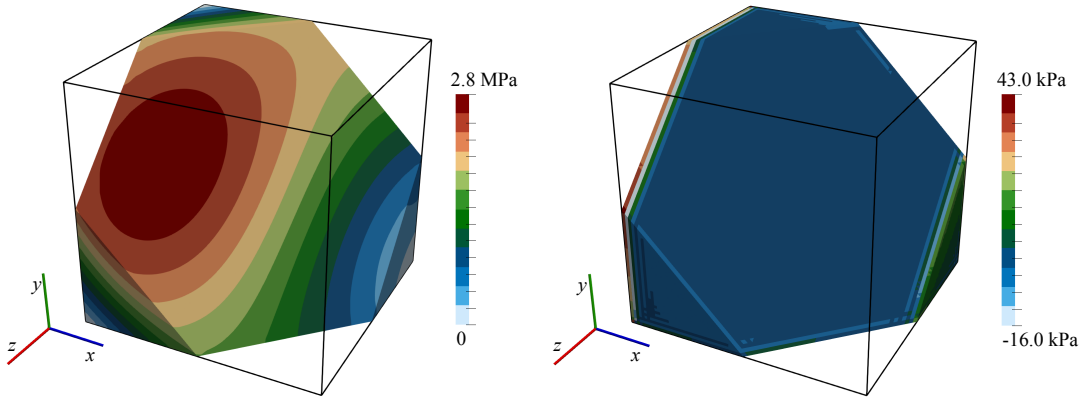
Figure 3(a) shows the displacement magnitude discretisation errors (L_2 and L_∞) as a function of the average cell width for the three mesh types (hexahedral, tetrahedral and polyhedral), while Figure 3(b) shows the corresponding order of accuracy plots. For ease of interpretation, the symbol shapes in the figures have been chosen to correspond to the cell shapes: a square for hexahedra, a triangle for tetrahedra and a pentagon for polyhedra. The maximum (L_∞) and average (L_2) displacement discretisation errors are seen to reduce at an approximately second-order rate for all mesh types, except for the L_2 error on the hexahedral meshes, which is approximately 2.3 on the finest grid.



(a) Displacement magnitude discretisation errors (b) Displacement discretisation error order of accuracy

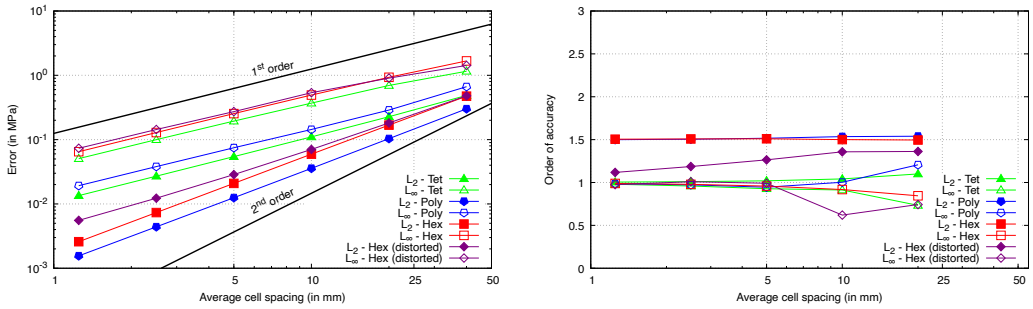
Fig. 3 Manufactured solution cube case: the accuracy and order of accuracy for displacement magnitude on the *regular* meshes

The predicted σ_{xx} stress distribution for a hexahedral mesh with 512 000 cells is shown in Figure 4(a). The corresponding cell-wise σ_{xx} error distribution is shown in Figure 4(b). The errors of greatest magnitude (43 kPa) occur at the boundaries, corresponding to where the local truncation error is higher. The discretisation errors in the stress magnitude as a function of cell size are shown in Figure 5(a), and the order of accuracy in Figure 5(b). The order of accuracy for the average (L_2) stress error is seen to be approximately 1.5 for the hexahedral and polyhedral meshes. In contrast, the average stress error order for the tetrahedral meshes is 1. Similarly, the maximum (L_∞) stress order of accuracy is seen to approach 1 for all three mesh types.



(a) Predicted σ_{xx} stress distribution (b) Cell-wise σ_{xx} error distribution

Fig. 4 Manufactured solution cube case: the predicted σ_{xx} stress distribution on a cut-plane for the regular hexahedral mesh with 512 000 cells (left).



(a) Stress magnitude discretisation errors (b) Stress discretisation error order of accuracy

Fig. 5 Manufactured solution cube case: the accuracy and order of accuracy for stress magnitude

Case 2: Out-of-plane bending of an elliptic plate

This 3-D, static, linear elastic test case (Figure 6) consists of a thick elliptic plate (0.6 m thick) with a centred elliptic hole, with the inner and outer ellipses given as

$$\left(\frac{x}{2}\right)^2 + \left(\frac{y}{1}\right)^2 = 1 \text{ inner ellipse} \tag{45}$$

$$\left(\frac{x}{3.25}\right)^2 + \left(\frac{y}{2.75}\right)^2 = 1 \text{ outer ellipse} \tag{46}$$

The case has been described by the National Agency for Finite Element Methods and Standards (NAFEMS) [47], and analysed using finite volume procedures by Demirdžić et al. [48] and Cardiff et al. [49]. Symmetry allows one-quarter of the geometry to be simulated. A constant pressure of 1 MPa is applied to the upper surface, and the outer surface is fully clamped. The mechanical properties are: $E = 210$ GPa, $\nu = 0.3$. Six successively refined hexahedral meshes are used, with cell counts of 45, 472 (Figure 6), 4 140, 34 968, 287 280 and 2 438 242.

The prediction for the equivalent (von Mises) stress in the domain are shown in Figure 7(a), and the values along the line $r = \sqrt{x^2 + y^2} = 2.1$ m, $z = 0.3$ m are shown in Figure 7(b). The results from the finest grid in Demirdžić et al. [48] are given for comparison, where good agreement is seen; the small offset between the finest mesh predictions and those from Demirdžić et al. [48] are likely due to errors introduced when extracting the Demirdžić et al. [48] results using

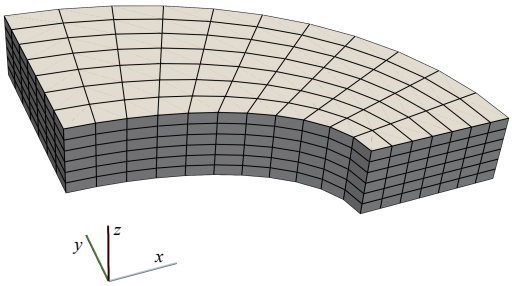
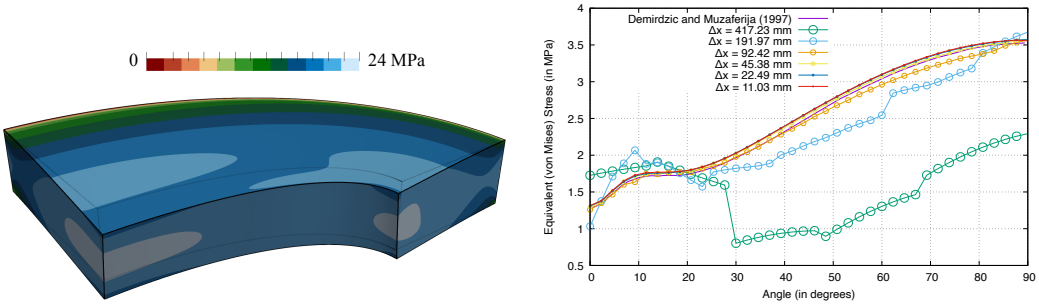


Fig. 6 Elliptic Plate geometry and mesh containing 472 hexahedral cells

WebPlotDigitizer [50]. The sample values have been calculated by generating 40 uniformly spaced angles between 0 and $\pi/2$ along the line, finding the cell of each sampled point, and extrapolating from the cell-centred value using the field gradient.



(a) Equivalent stress distribution for the mesh with 2 438 242 cells. The deformation is scaled by 1 000 and a translucent line indicates the outline of the undeformed geometry

(b) Equivalent stress along the line $r = 2.1$, $z = 0.3$ m

Fig. 7 Equivalent (von Mises) stress distribution in the elliptic plate

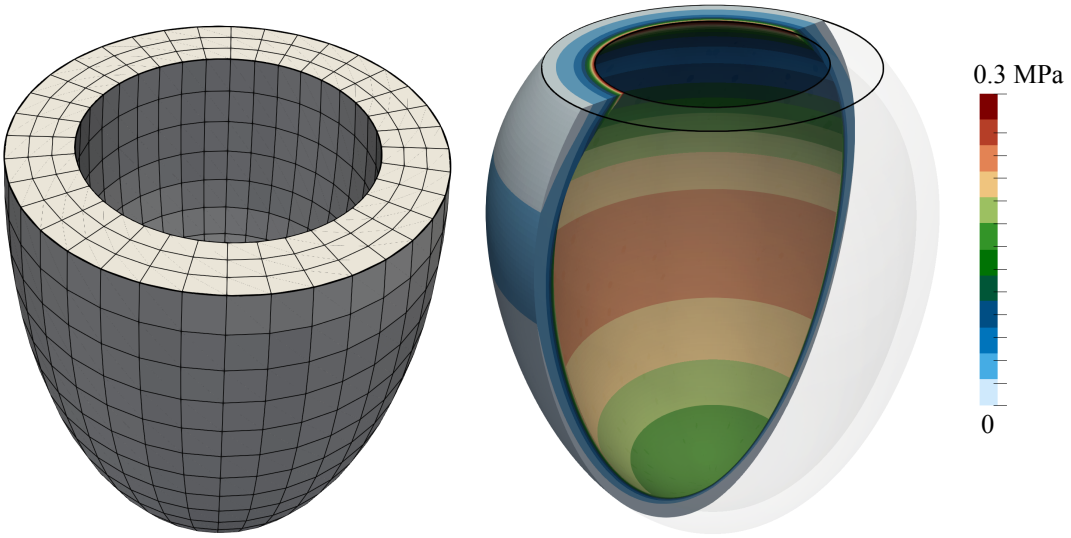
Case 3: Inflation of an idealised ventricle

Inflation of an idealised ventricle (Figure 8) was proposed by Land et al. [51] as a benchmark problem for cardiac mechanics software. The case is 3-D, static, with finite hyperelastic strains. The initial geometry is defined as a truncated ellipsoid:

$$x = r_s \sin(u)\cos(v), \quad y = r_s \sin(u)\sin(v), \quad z = r_l \cos(u) \tag{47}$$

where on the inner (endocardial) surface $r_s = 7$ mm, $r_l = 17$ mm, $u \in \left[-\pi, -\arccos\left(\frac{5}{17}\right)\right]$ and $v \in [-\pi, \pi]$, while on the outer (epicardial) surface $r_s = 10$ mm, $r_l = 20$ mm, $u \in \left[-\pi, -\arccos\left(\frac{5}{20}\right)\right]$ and $v \in [-\pi, \pi]$. The base plane $z = 5$ mm is implicitly defined by the ranges for u. The hyperelastic material behaviour is described by the transversely isotropic constitutive law proposed by Guccione et al. [52] law, where the parameters are $C = 10$ kPa, $c_f = c_t = c_{fs} = 1$; the specified parameters produce isotropic behaviour. The benchmark specifies an incompressible material; in the current work, incompressibility is enforced weakly using a penalty approach, where the bulk modulus parameter is chosen to be two orders of magnitude greater ($\kappa = 1000$ kPa) than the greatest shear modulus parameter. A pressure of 10 kPa is applied to the inner surface, applied here over 100 equal load increments, and the base plane is fixed. The geometry is meshed using a structured

approach and is predominantly composed of hexahedra, with prism cells forming the apex. The OpenFOAM utilities `blockMesh` and `extrudeMesh` are used to create the meshes. Four successively refined meshes are examined: 1 620 (shown in Figure 8(a)), 12 960, 103 680, and 829 440 cells. The case can be simulated as 2-D axisymmetric but is simulated here using the full 3-D geometry.



(a) Ventricle undeformed geometry, showing the mesh with 1,620 cells

(b) Cross-section showing the equivalent (von Mises) stress distribution for the mesh with 829,440 cells

Fig. 8 Idealised ventricle case

The equivalent (von Mises) stress at full inflation is shown for the mesh with 829,440 cells in Figure 8(b), where the high stresses on the inner (endocardial) surface quickly drop off through the wall thickness towards the outer (endocardial) surface. The predicted deformed configuration of the ventricle wall midline is shown in Figure 9, where a side-by-side comparison is given with the round-robin benchmark results from Land et al. [51]. The predictions are seen to become quickly mesh-independent and fall within the benchmark ranges.

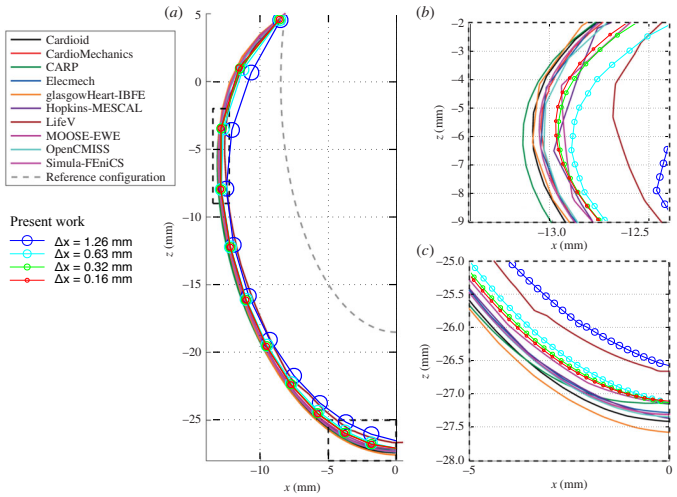
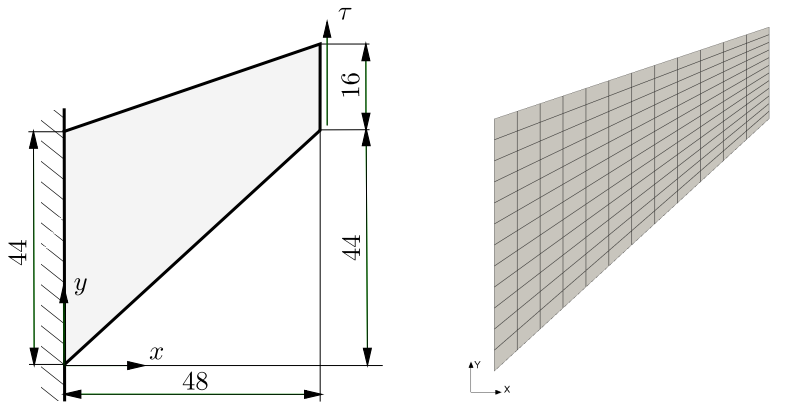


Fig. 9 Idealised ventricle: predictions for the four meshes from the present work overlaid on the round-robin predictions from Land et al. [51]

Case 4: Cook's membrane

Cook's membrane (Figure 10) is a well-known bending-dominated benchmark case used in linear and non-linear analysis. The 2-D plane strain tapered panel (trapezoid) is fixed on one side and subjected to uniform shear traction on the opposite side. The vertices of the trapezoid (in mm) are (0, 0), (48, 44), (48, 60), and (0, 44). The problem is solved quasi-statically, using 30 equally-sized loading increments, and there are no body forces.



(a) Case geometry and dimensions (in mm)

(b) Quadrilateral mesh with 144 cells

Fig. 10 Cook's membrane case geometry and mesh

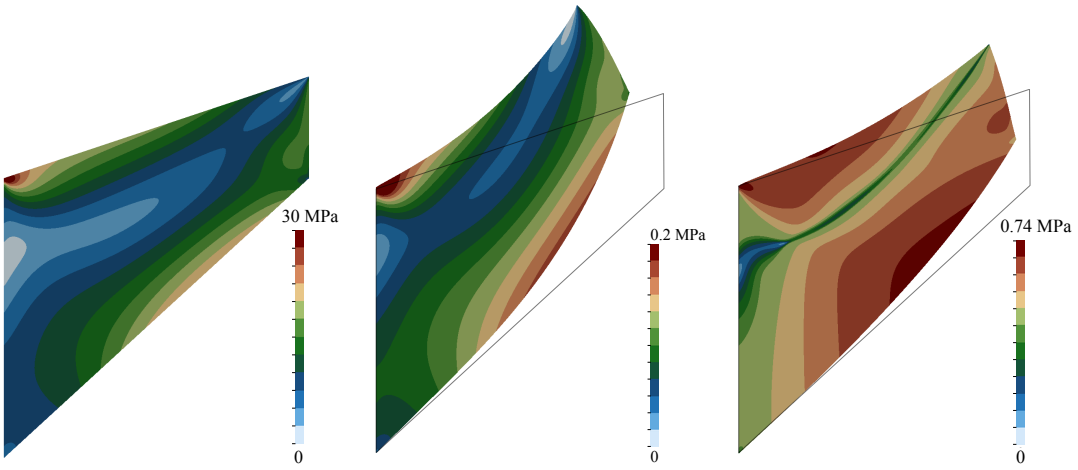
The current work considers three forms of the problem:

i. Small strain, linear elastic [53, 54]: $E = 70$ MPa, $\nu = 1/3$, and $\tau = 6250$ Pa.
ii. Finite strain, neo-Hookean hyperelastic [55]: $E = 1.0985$ MPa, $\nu = 0.3$, and $\tau = 0.0625$ MPa.
iii. Finite strain, neo-Hookean hyperelastoplastic [54, 56, 57]: $E = 206.9$ MPa, $\nu = 0.29$, $\sigma_y = 0.45 + 0.12924\bar{\varepsilon}_p + (0.715 - 0.45)(1 - e^{-16.93\bar{\varepsilon}_p})$ MPa, and $\tau = 0.3125$ MPa.

where E is the Young's modulus, ν is the Poisson's ratio, σ_y is the yield strength, $\bar{\varepsilon}_p$ is the equivalent plastic strain, and τ is the prescribed shear traction. Eight successively refined quadrilateral meshes are considered, where the cell counts are 9, 36, 144 (Figure 10(b)), 576, 2 304, 9 216, 36 864, and 147 456.

Figure 11 shows the predicted equivalent stress distribution on the mesh with 36 842 cells for linear elastic, hyperelastic and hyperelastoplastic cases. The stress distribution is consistent with bending, with regions of high stress near the upper and lower surfaces and a line of relatively unstressed material in the centre. The greatest equivalent stresses occur at the top-left corner and the lower surface in all versions of the case. In the hyperelastoplastic case, almost the entire domain is plastically yielding with only a small thin region remaining elastic, indicated by the blue line in Figure 11(c).

Figure 12 compares the predicted vertical displacement at the reference point as a function of the average cell widths with results from the literature. The reference point is taken as the top right point – $(48, 60)$ mm – in the elastic and hyperelastoplastic cases, while it is taken as the midway point of the loading surface – $(48, 52)$ mm – in the hyperelastic case.



(a) Linear elastic case kPa (b) Hyperelastic case (c) Hyperelastoplastic case

Fig. 11 Equivalent (von Mises) stress distribution for the three Cook's membrane cases using the mesh with 36 842 cells

The results shown for the hyperelastoplastic case were generated using the segregated solution procedure as the Jacobian-free Newton-Krylov approach diverged for five of the eight mesh densitities, as discussed further in Section 4.2.

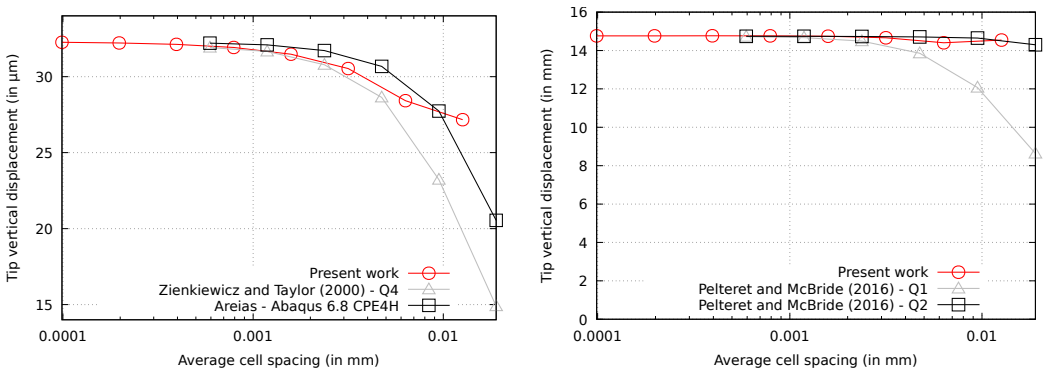
Case 5: Vibration of a 3-D Cantilevered Beam

This 3-D, dynamic, finite strain case geometry consists of a $2 \times 0.2 \times 0.2$ m cuboid column and was proposed in its initial form by Tuković and Jasak [58]. A sudden, constant traction $\boldsymbol{T} = (80, 80, 0)$ kPa is applied to the upper surface. A neo-Hookean hyperelastic material is assumed with $E = 15.293$ MPa, $\nu = 0.3$ and density $\rho = 1000$ kg m^{-3}, while the area $A = 0.04$ m^2 and the second moment of area $I = 1/7\,500$ m^4. The geometric and material parameters were chosen for the first natural frequency to be 1 Hz in the small strain limit. The magnitude of the applied traction is chosen here to test the robustness of the solution procedures when large rotations and strains are present. Four succesively refined hexahedral meshes are generated using the OpenFOAM `blockMesh` utility, with cell counts of 270, 2 160, 17 280 and 138 240. The time step size is 1 ms, and the total period is 1 s.

The deformed configuration of the beam at six time steps is shown in Figure 13(a) for the mesh with 2 160 cells, where the upper surface is seen to drop below the horizontal from approximately t = 0.2 s to 0.32 s. The corresponding displacement magnitude of the centre of the upper surface of the beam vs time is shown in Figure 13(b), where the predictions are seen to converge to the results from finite element software Abaqus (element code C3D8) using the 138 240 mesh.

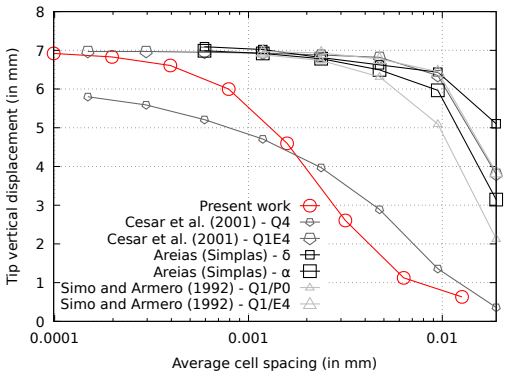
Case 6: Elastic plate behind a rigid cylinder

The final test case, introduced by Turek and Hron [59], extends the classic flow around a cylinder in a channel problem [60] into a well-established fluid-solid interaction benchmark. The `FSI3` variant of the case examined here (Figure 14) consists of a horizontal channel (0.41 m in height, 2.5 m in length) with a rigid cylinder of radius 0.05 m, where the cylinder centre is 0.2 m from the bottom



(a) Linear elastic case

(b) Hyperelastic case

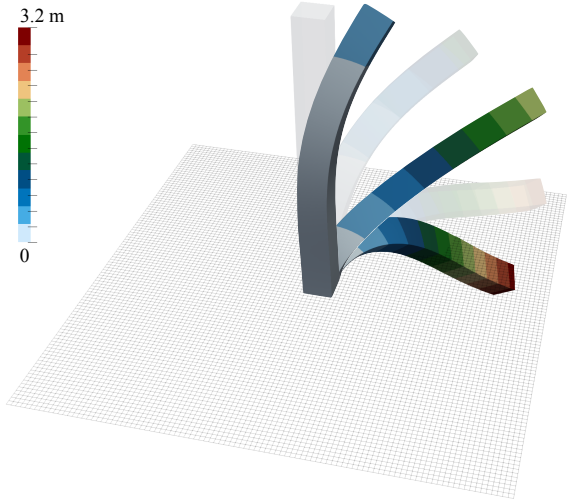


(c) Hyperelastoplastic case

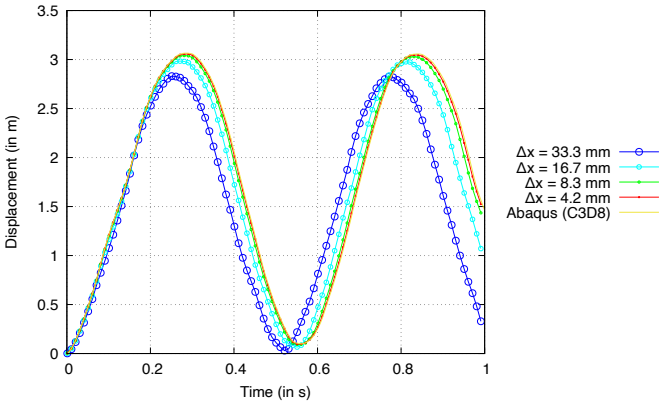
Fig. 12 Cook's membrane vertical displacement predictions at the reference point as a function of the average cell width. Comparisons are given with the results from the literature [53–56]. Ref NOT tip

and inlet (left) boundaries. A parabolic velocity is prescribed at the inlet velocity with a mean value of 2 m/s. A St. Venant-Kirchhoff hyperelastic plate ($E = 5.6$ MPa, $\nu = 0.4$) of 0.35 m in length and 0.02 m in height is attached to the right-hand side of the rigid cylinder. The fluid model is assumed to be isothermal, incompressible, and laminar and adopts the segregated PIMPLE solution algorithm. The fluid's kinematic viscosity is 0.001 m^2/s and density is 1000 kg m^{-3}, while the solid's density is 10 000 kg m^{-3}. The interface quasi-Newton coupling approach with inverse Jacobian from a least-squares model [61] is employed for the fluid-solid interaction coupling. The fluid-solid interface residual is reduced by four orders of magnitude within each time step. Further details of the fluid-solid interaction procedure are found in Tuković et al. [11]. Three successively refined quadrilateral meshes are employed: The fluid region meshes have 1 252, 5 008, and 20 032 cells, while the solid region meshes have 156, 624 and 2 496 cells. The total simulation time is 20 s, and the time-step size is 0.5 ms.

Figure 15 shows the velocity field in the fluid region and displacement magnitude in the solid region at $t = 4.42$ s, corresponding to a peak in the vertical displacement of the plate free-end oscillation. The fluid is seen to accelerate as it passes the cylinder, with regular vortices being thrown off the plate, causing it to oscillate. The predicted mean, amplitude and frequency of the vertical and horizontal displacement of the end of the plate are compared with the results from Turek and Hron [59] in Table 2. As directed in Turek and Hron [59], the maximum and minimum

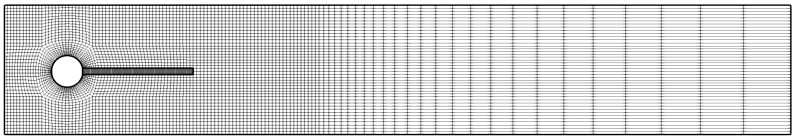


(a) updateDisplacement magnitude for t = {0, 0.1, 0.2} s (translucent) and t = {0.3, 0.4, 0.5} s (opaque) for the mesh with 138 240 cells

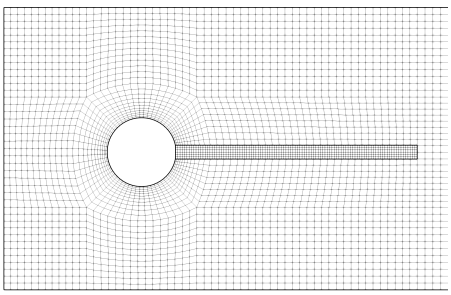


(b) Displacement magnitude of the centre of the upper surface of the beam vs time

Fig. 13 Deflection of a 3-D Cantilevered Beam



(a) Mesh containing 5 336 cells in the fluid region and 630 cells in the solid region



(b) Close-up of the mesh around the plate

Fig. 14 Elastic plate behind a rigid cylinder case geometry and mesh

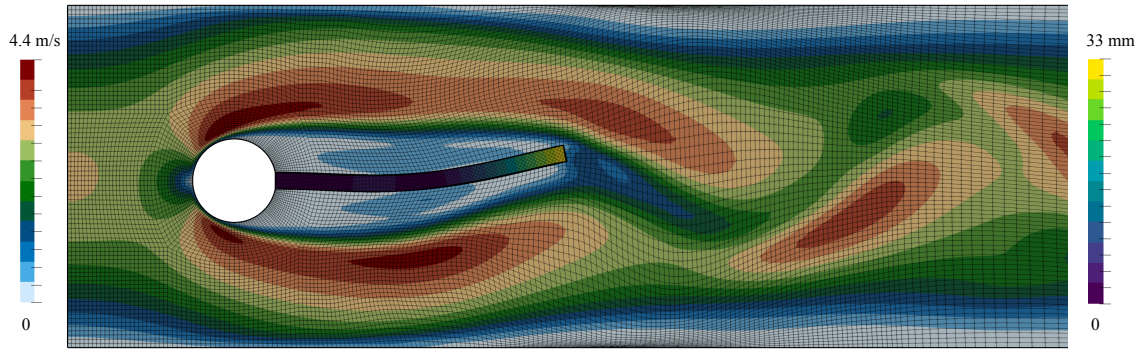


Fig. 15 Elastic plate behind a rigid cylinder case results at $t = 20$ s for the mesh with 20 032 cells in the fluid region and 2 496 in the solid region. The velocity magnitude is shown in the fluid region while the displacement magnitude is shown in the solid region.

values for the last period are used to calculate the mean and amplitude:

$$\text{mean} = \frac{1}{2}(\max + \min), \qquad \text{amplitude} = \frac{1}{2}(\max - \min) \tag{48}$$

while the frequency is calculated as the inverse of the period.

Mesh (number of fluid and solid cells)	u_x (in mm)	u_y (in mm)
1 (1 252 + 156)	-0.1 ± 0.001 [47.6]	1.58 ± 0.002 [5.88]
2 (5 008 + 624)	-2.16 ± 2.00 [10.2]	1.54 ± 29.56 [5.6]
3 (20 032 + 2 496)	-2.51 ± 2.37 [12.1]	0.87 ± 33.08 [5.6]
Turek and Hron [59]	-2.69 ± 2.53 [10.9]	1.48 ± 34.38 [5.3]

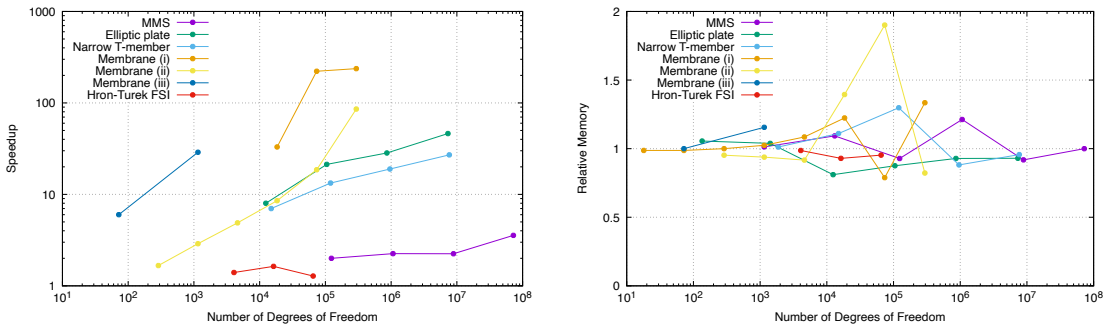
Table 2 Average predicted displacements (mean $\pm$ amplitude [frequency]) of the free end of the plate. Note that large-scale plate oscillations in mesh 1 (coarsest mesh) die out within the first 10 s.

4.2 Resource Requirements and Robustness

This section compares resource requirements and the robustness of the Jacobian-free Newton-Krylov and segregated approaches on the cases presented in Section 4.1. Specifically, time and memory requirements are analysed, and whether the solution converged for all meshes and loading steps. In all cases, the Jacobian-free Newton-Krylov approach used the generalised minimal residual method (GMRES) linear solver with the direct LU preconditioner for 2-D and the Hypre BoomerAMG multigrid preconditioner for 3-D, unless stated otherwise. A restart value of 30 is used for the GMRES solver, where the *loose* variant of the GMRES solver is used, which retains a small number of extra solution vectors across restarts (2 in this case). In contrast, the segregated approach used the conjugate gradient linear solver and the Hypre BoomerAMG multigrid preconditioner for all cases. Time and memory requirements are implementation and hardware-specific; nonetheless, it is insightful to see the relative performances of the Jacobian-free Newton-Krylov and segregated approaches on the same hardware and within the same implementation framework. 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 4.5.

Table 3 lists the wall clock times (time according to a clock on the wall) and maximum memory usage for all cases and all meshes, where diverged cases are indicated by the symbol †. Figure 17(a) plots the corresponding speedup on all cases as a function of the number of degrees of freedom, where the speedup is defined as the segregated clock time divided by the Jacobian-free Newton-Krylov clock time. The number of degrees of freedom is 2× the cell count in 2-D cases and 3× in 3-D cases, except in the fluid-solid interaction case where the fluid domain has three degrees of freedom (two velocity components and pressure) in 2-D. Speedup is only calculated for clock times greater than 2 seconds to avoid the comparison of small numbers. In all cases where convergence was achieved, the Jacobian-free Newton-Krylov approach was faster (speedup > 1), with speedups of one or two orders of magnitude in many cases. Additionally, it can be observed from Figure 17(a) that the speedup increases as the number of degrees of freedom increases. The largest speedup was 237 on the finest mesh of the membrane i (2-D, linear elastic) case, while the lowest speed was 1.09 and occurred on the coarsest mesh in the fluid-solid interaction case. Excluding the fluid-solid interaction, where the majority of degrees of freedom are placed in the fluid region, the next lowest speed-up was 1.66 and occurred on a coarse mesh in the membrane ii (2-D, hyperelastic) case.

Figure 16(b) shows the *relative memory* usage, defined here as the maximum memory usage of the segregated approach divided by that of the Jacobian-free Newton-Krylov approach. The relative memory usage is close to unity in most cases, indicating there is no significant increase in memory requirements when switching from a segregated approach to a Jacobian-free Newton-Krylov approach. In addition, unlike for the speedup, there is no general trend in the relative memory usage as the number of degrees of freedom increases. The maximum value for relative memory was 1.9 and occurred in the second-finest mesh for the membrane ii (2-D, hyperelastic) case, although for all other meshes in this case, the value was less than one (between 0.82 and 0.95). The maximum memory usage is primarily attributed to the linear solver and preconditioner; the choice of preconditioner and linear solver settings are examined further in Section 4.3.



(a) Speedup

(b) Relative memory

Fig. 16 The speedup (segregated clock time divided by the Jacobian-free Newton-Krylov clock time) and relative memory usage (segregated maximum memory usage divided by the Jacobian-free Newton-Krylov maximum memory usage) as a function of degrees of freedom

Regarding robustness, the Jacobian-free Newton-Krylov approach converged in all cases bar one, requiring less than five outer Newton iterations on average in all converged cases. The only case where the Jacobian-free Newton-Krylov approach failed to converge was for the membrane iii problem, the only elastoplastic case examined. For the failed meshes, approximately 80% of the total

Case	Cell Count	JFNK		Segregated	
		Time (in s)	**Memory** (in MB)	**Time** (in s)	**Memory** (in MB)
MMS - regular	384	0	78	0	79
3-D, static,	4 374	0	86	0	94
linear elastic	41 154	1	195	2	181
cell type	355 914	4	1 063	9	1 289
	2 958 234	41	6 724	92	6 174
	24 118 074	486	30 606	1 732	30 601
Elliptic	45	0	72	0	76
Plate	472	0	78	0	81
3-D, static,	4 140	1	153	8	124
linear elastic	34 968	7	754	149	660
	287 280	95	3 943	2 694	3 661
	2 438 242	988	21 884	45 692	20 348
Ventricle	1 620	54	133	†	†
3-D, static,	12 960	576	553	†	†
hyperelastic	103 680	8 362	3 435	†	†
	829 440	150 023	11 746	†	†
Membrane i	9	0	77	0	76
2-D, static,	36	0	77	0	76
linear elastic	144	0	78	0	78
	576	0	82	1	84
	2 304	0	94	4	102
	9 216	1	152	33	186
	36 864	2	497	444	392
	147 456	10	1 461	2 371	1 950
Membrane ii	9	1	81	1	78
2-D, static,	36	1	81	2	78
hyperelastic	144	3	84	5	80
	576	9	97	26	91
	2 304	34	144	166	132
	9 216	161	256	1 374	357
	36 864	860	554	15 991	1 053
	147 456	1 585	2 847	135 851	2 341
Membrane iii	9	0	78	3	78
2-D, static,	36	1	79	6	79
hyperelastoplastic	144	†	†	27	82
	576	8	90	231	104
	2 304	†	†	1 379	180
	9 216	†	†	8 321	451
	36 864	†	†	60 698	1 396
	147 456	†	†	341 212	3 466
Cantilever	270	11	85	†	†
3-D, dynamic,	2 160	71	181	†	†
hyperelastic	17 280	854	752	†	†
	138 240	11 800	5 790	†	†
FSI	1 252 + 156	3 830	748	5 360	738
2-D, dynamic,	5 008 + 624	32 074	834	52 372	775
hyperelastic	20 032 + 2 496	170 919	785	218 797	748

Table 3 Execution times (rounded to the nearest second) and maximum memory usage for Jacobian-free Newton-Krylov (JFNK) and segregated methods (rounded to the nearest MB). † indicates the solver diverged.

load was reached before the linear solver diverged. The cause of this divergence is likely related to the proposed compact preconditioner matrix – which is formulated based on small *elastic* strains – being a poor approximation of the true Jacobian for large plastic strains. Interestingly, convergence problems were not encountered in the hyperelastic cases (membrane ii, ventricle, cantilever, fluid-solid interaction), demonstrating that the proposed compact *linear elastic* preconditioner matrix is suitable for such large strain, large rotation hyperelastic cases. In contrast, the segregated approach – which uses the same matrix – had no problems with the elastoplastic case (membrane iii) but failed to converge on two of the hyperelastic cases (ventricle, cantilever). In both cases where the segregated approach failed, large elastic strains and large rotations were observed, and the segregated solver failed in the early time steps. It is interesting to note that the same linear elastic

preconditioner matrix works well with the Jacobian-free Newton-Krylov approach for hyperelastic cases but not for the segregated approach, while the opposite is true for elastoplastic cases.

Regarding geometric dimension, the same general trends are observed in 2-D and 3-D, with no major distinctions in behaviour. The same can be said for geometric nonlinearity (small strain vs. large strain) with similar speed-ups for both linear elastic and hyperelastic cases. One observation worth highlighting is the lower speedup seen for the method of manufactured solutions case (3-D, linear elastic): a possible explanation is that the segregated approach has previously been seen to be efficient on geometry with low aspect ratios (ratio of maximum to minimum dimensions); in this case, the aspect ratio is at its minimum (unity). In contrast, in the other 3-D linear elastic cases (elliptic plate, membrane i), the aspect ratios are greater than unity, and the segregated approach performs worse relative to the Jacobian-free Newton-Krylov approach.

4.3 Effect of the Preconditioner Choice

This section examines the effect of preconditioning strategy on the performance of the Jacobian-free Newton Krylov approach. Three choices of preconditioning procedure are compared:

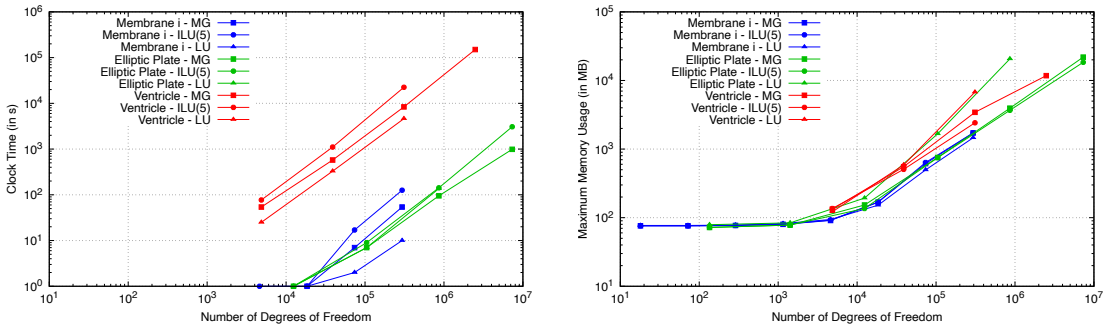
- **LU** - The *MUltifrontal Massively Parallel sparse direct Solver* (MUMPS) [43, 44] LU decomposition direct solver. A direct solver is expected to be the more robust but may suffer from excessive time and memory requirements for larger numbers of unknowns.
- **ILU(k)** - Incomplete LU decomposition with k fill-in. ILU(k) is expected to have lower memory requirements than the LU direct solver but at the expense of robustness. As the system of unknowns becomes larger, the number of ILU(k) iterations is expected to increase. As the fill-in factor k increases, ILU(k) approaches the robustness of a LU direct solver. In the current section, $k = 5$ for all cases examined.
- **Algebraic multigrid** - The Hypre Boomerang [42] parallelised multigrid preconditioner. Multigrid approaches have the potential to offer superior performance than other methods for larger problems, with near linear scaling of time and memory requirements.

An additional consideration when selecting a preconditioner is its ability to scale in parallel as the number of CPU cores increases. From this perspective, the iterative approaches (ILU(k) and multigrid) are expected to show better parallel scaling than direct methods (LU), a point that is briefly examined in Section 4.5.

The preconditioning approaches are compared in three cases: membrane i (2-D, linear elastic), elliptic plate (3-D, linear elastic), and idealised ventricle (3-D, hyperelastic). Figure 17 compares the clock times and maximum memory requirements for the three preconditioning approaches as a function of degrees of freedom. Examining the clock times, the LU preconditioner is seen to be faster for all meshes in the membrane i and idealised ventricle cases, while the multigrid approach is faster for all meshes in the elliptic plate cases. The LU approach was found to be up to 5.4× faster in the membrane i case than the multigrid approach, and 1.8× faster in the idealised ventricle case. In contrast, the multigrid was up to 1.4× faster than the LU approach in the elliptic plate case. The ILU(5) approach is seen to be slowest in all cases, in addition, the ILU(5) approach diverged

on the idealised ventricle finest mesh; lower values of fill-in k were found to cause divergence to occur earlier in the idealised ventricle case.

Regarding memory usage, below approximately 50 k degrees of freedom, the memory usage is the same for all three approaches (approximately 80 MB), corresponding to the solver's minimum memory overhead. For greater numbers of degrees of freedom, the LU approach uses the greatest amount of memory for both 3-D cases (idealised ventricle, elliptic plate), while the ILU(5) uses the least. In contrast, the LU approach uses the smallest amount of memory for the 2-D cases (membrane i) examined. The rate of memory increase is seen to be much steeper for the LU approach on the 3-D cases than the multigrid and ILU(5) approaches, with the LU results for the finest meshes not shown as they required more than the maximum available memory (64 GB).



(a) Clock time

(b) Maximum memory

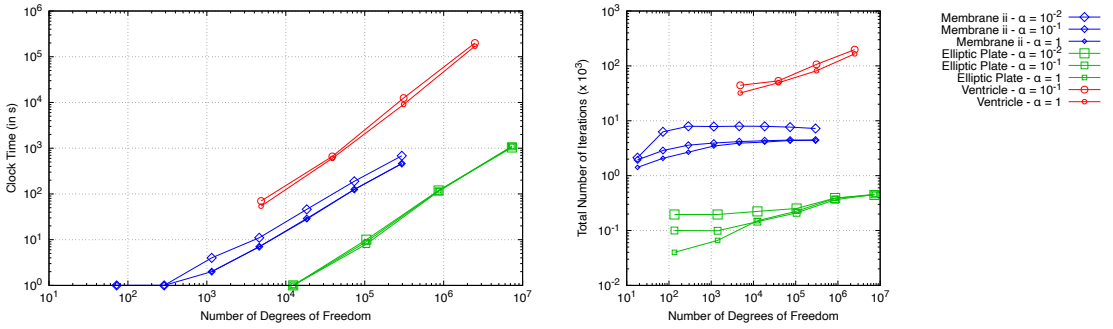
Fig. 17 The clock times and maximum memory usage for three preconditioning approaches (multigrid, ILU(5), and LU) as a function of degrees of freedom

4.4 Effect of Rhie-Chow Stabilisation

This section highlights the impact of the global scaling factor α in the Rhie-Chow stabilisation (Equation 27) on the performance of the proposed Jacobian-free Newton Krylov method. As described in Section 2.3, the stabilisation term is introduced to quell zero-energy modes (oscillations) in the discrete solution, such as checkerboarding. As the amount of stabilisation increases (increasing α), these numerical modes are quelled, and the solution stabilises; however, at some point, further increases in the amount of stabilisation reduce the accuracy of the discretisation due to over-smoothing. A less obvious consequence of changing the stabilisation magnitude is its effect on the convergence of the linear solver in the Jacobian-free Newton-Krylov solution procedure. This section examines this effect.

As in the previous section, three cases are used to highlight the effect: membrane i (2-D, linear elastic), elliptic plate (3-D, linear elastic), and idealised ventricle (3-D, hyperelastic). The LU preconditioning method is used for the 2-D membrane i case, while the multigrid approach is used for the two 3-D cases. Figure 18 presents the execution times and the number of accumulated linear solver iterations for three values of global stabilisation factor ($\alpha = [0.01, 0.1, 1]$) for several mesh densities. The memory usage is unaffected by the value of α and is hence not shown. From Figure 18(a), the clock time is seen to increase exponentially (linearly on a log-log plot) for all cases and

all values of α. For all meshes in all three cases, the lowest value of global stabilisation factor ($\alpha = 0.01$) – corresponding to the least amount of stabilisation – takes the greatest amount of time to converge, while the largest value of global stabilisation factor ($\alpha = 1$) is the fastest to converge. In the membrane i case, the largest value of (α) is approximage 1.5× faster than the lowest value of α. Nonetheless, the clock time is not a linear function of α, and it can be seen that $\alpha = 0.1$ requires approximately the same amount of time as $\alpha = 1$ in all cases. In terms of robustness, the idealised ventricle cases failed with $\alpha = 0.01$, suggesting increasing the stabilisation increases robustness. From Figure 18(b), the accumulated number of linear solver (GMRES) iterations are seen to follow the same trends as the clock times, where $\alpha = 0.01$ requires the greatest number of iterations while $\alpha = 0.1$ and $\alpha = 1$ require approximately the same number of iterations. In all cases, the number of iterations are seen to increase as the number of degrees of freedom increase.



(a) Clock time

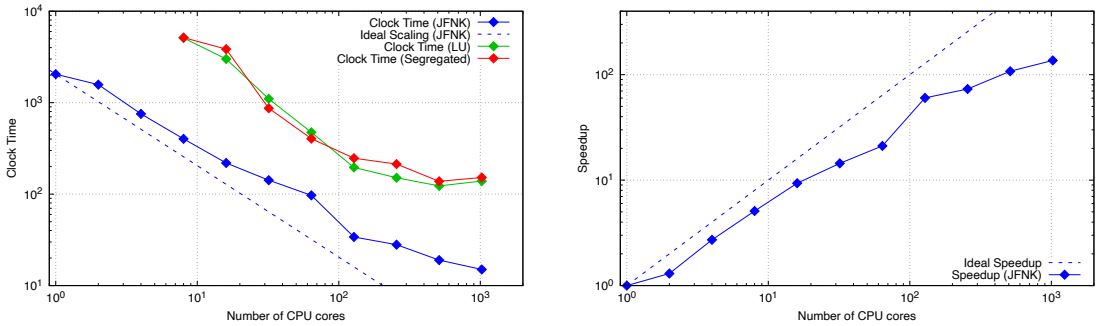
(b) Accumulated linear solver iterations

Fig. 18 The clock times and accumulated linear solver iterations for different values of Rhie-Chow global stabilisation factor α

4.5 Parallelisation

In this final analysis, the multi-CPU-core parallel scaling performance of the proposed Jacobian-free Newton Krylov method is compared with the segregated approach. A *strong* scaling study is performed, where the clock time to solve a fixed-size problem is measured as the number of CPU cores is increased. The elliptic plate (3-D, linear elastic) mesh with 2 438 242 cells is chosen for the scaling analysis. Parallelisation adopts the standard OpenFOAM domain decomposition approach, where the mesh is decomposed into one sub-domain for each CPU core. In the current work, the Scotch decomposition approach [62] is employed. The Jacobian-free Newton Krylov approach uses the GMRES linear solver where the performance of the multigrid and LU preconditioners are compared. The segregated approach uses the conjugate gradient solver with the incomplete (zero in-fill – $k = 0$) Cholesky preconditioner. The cases are run on the MeluXina high-performance computing system, where each standard computing node contains 2× AMD EPYC Rome 7H12 64c 2.6GHz CPUs with 512 GB of memory. In the current study, the number of CPU cores are varied from 1 to 1 048 ($[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1\,048]$). In the ideal case, the speedup should double when the number of cores is doubled; in reality, inter-CPU-core communication reduces the parallel scaling efficiency below the ideal.

The clock times from the strong scaling study are shown in Figure 19(a), where the *ideal* scaling is shown for comparison as a dashed line. Figure 19(b) shows the corresponding speedup S, defined as $S = {}^{t_1}/_{t_p}$, where t_1 is the clock time on one CPU core and t_p is the clock time on P CPU cores. The Jacobian-free Newton-Krylov approach with the multigrid preconditioner is found to be the fastest by almost an order of magnitude over the segregated approach and the Jacobian-free Newton-Krylov approach with LU preconditioner. The times for the segregated and Jacobian-free Newton-Krylov LU preconditioner approaches using 1, 2, and 4 cores were not recorded due to the excessive times. For all approaches the speedup is seen to increase at an approximately ideal rate up to 128 cores, after which the speedup continues to increase but at a lower rate. At 128 cores (corresponding to one full computing node), the number of cells per core is approximately 19 k; increasing the number of cores beyond 128 likely shows a less than ideal scaling for two reasons: (i) the number of cells per core is becoming small relative to the amount of inter-core communication, and (ii) inter-node communication is likely slower than intra-node communication. The segregated and Jacobian-free Newton-Krylov LU preconditioner approaches produce their fastest predictions at 512 cores, while the Jacobian-free Newton-Krylov multigrid preconditioner approach continues to speedup up to the maximum number of cores tested (1 048), albeit 1 048 cores is just 2× faster than 128 cores. At 1 048 cores, there is approximately 2.5 k cells per core and the inter-core communication is expected to be significant.



(a) Clock times vs number of CPU cores

(b) Speedup S vs number of CPU cores

Fig. 19 Strong parallel scaling study comparing the performance of the Jacobian-free Newton-Krylov approach using the multigrid and LU preconditioning strategies, and the segregated solution procedure

As a final observation, the effect of hardware can be highlighted by comparing the time required for 1 core when using the multigrid preconditioner: the case required 2 048 s on the MeluXina system (EPYC Rome CPU), while it required only 988 s on a Mac Studio (M1 Ultra CPU) system as presented in Section 4.2 – over twice as fast. The difference in performance can be primarily attributed to the unified memory architecture of the M1 Ultra, which provides significantly greater memory bandwidth compared to the DDR4 memory used in the MeluXina compute nodes.

5 Conclusions

A Jacobian-free Newton-Krylov solution algorithm has been proposed for solid mechanics problems discretised using the cell-centred finite volume method. A compact-stencil discretisation of the

diffusion term is proposed as the preconditioner matrix, allowing a straightforward extension of existing segregated solution frameworks. The key conclusions of the work are:

- **Efficiency of Jacobian-free Newton-Krylov approach**: The proposed Jacobian-free Newton-Krylov solution algorithm has been shown to be faster than a conventional segregated solution algorithm for all linear and nonlinear elastic test cases examined. In particular, speedups of one order of magnitude were seen in many cases, with a maximum speedup of over 200 in the 2-D linear elastic Cook's membrane case.
- **No additional memory overhead**: The Jacobian-free Newton-Krylov approach has been shown to have approximately the same memory requirements as a segregated solution algorithm, assuming a similar preconditioning approach is adopted, e.g. multigrid.
- **Applicability to existing segregated frameworks**: By employing a compact-stencil diffusion approximation of the Jacobian as the preconditioner, the proposed Jacobian-free Newton-Krylov approach can be integrated into existing segregated finite volume frameworks with minimal modifications to the existing code base, in particular if existing publicly available Jacobian-free Newton-Krylov solvers (e.g., PETSc) are used.
- **Choice of preconditioning approach**: It has been shown that the LU direct solver preconditioning approach is faster than multigrid and ILU(k) approaches for 2-D cases and moderately-sized 3-D problems; however, for larger 3-D problems the multigrid approach is the fastest while also requiring less memory than the direct LU approach. In addition, the multigrid approach is seen to show approximate ideal scaling in a parallel strong scaling study.
- **Rhie-Chow stabilisation**: The magnitude of the Rhie-Chow stabilisation term is shown to affect the speed and robustness of the Jacobian-free Newton-Krylov approach, where $\alpha = 0.1$ and 1 are seen to outperform $\alpha = 0.01$.
- **Open access and extensibility**: The implementation is made publicly available within the solids4foam toolbox for OpenFOAM to encourage implementation critique, community adoption, and comparative studies, contributing to advancing finite volume solid mechanics simulations.

The presented study highlights the potential of the Jacobian-free Newton-Krylov method in finite-volume solid mechanics simulations, yet several areas for future exploration remain:

1. Enhanced preconditioning for plasticity: While the proposed *elastic* compact-stencil preconditioning matrix has been shown to perform well in linear and nonlinear elastic scenarios, it fails in cases exhibiting plasticity, where a segregated approach succeeds. Future studies will explore modifications to the compact preconditioning matrix to extend the applicability of the proposed approach to small and large strain elastoplasticity.
2. Incorporation of non-orthogonality in the preconditioner: The current preconditioning matrix does not account for non-orthogonal contributions in distorted meshes. Including these effects in future work could further enhance convergence behaviour and accuracy, particularly for complex geometries.
3. Comparison with full Jacobian approaches: A comparison with a fully coupled Jacobian-based method would provide greater insights into the trade-offs between computational cost, memory usage, and convergence performance. Such a study could highlight the specific advantages of the Jacobian-free Newton-Krylov approach for large-scale, nonlinear problems.

4. Globalisation strategies: The robustness and efficiency of the proposed Jacobian-free Newton-Krylov approach could be improved for nonlinear problems by using a segregated approach as an initialisation phase for each loading step. In its current form, the segregated approach would not be generally suitable as it is less robust on several nonlinear hyperelastic problems; however, modifications such as introducing pseudo-time-stepping, may enhance its robustness and suitability for these scenarios.
5. Higher-order discretisations: Future work will explore the application of the proposed linear elastic compact-stencil Jacobian-free Newton-Krylov approach to higher-order finite volume discretisations. Using the same compact-stencil preconditioning matrix for these discretisations could yield efficient and accurate solutions for problems requiring increased fidelity.
6. Development of a monolithic fluid-solid interaction framework: Extending the Jacobian-free Newton-Krylov method to a fully monolithic fluid-solid interaction framework presents a promising avenue. In this approach, both fluid and solid domains would be solved simultaneously using a unified Jacobian-free Newton-Krylov method, potentially improving stability and performance in highly nonlinear coupled problems.

Acknowledgments. Technical reviews and insightful comments from Hiroaki Nishikawa of the National Institute of Aerospace, Hampton, VA, USA, are greatly appreciated. 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 RC2302_2, 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. [52] 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 Neo-Hookean J_2 Hyperelastoplasticity

For neo-Hookean J_2 Hyperelastoplasticity, the expression for the true (Cauchy) stress $\boldsymbol{\sigma}$ takes the same form as Equation A6, except $\boldsymbol{b}$ is replaced by its elastic component $\boldsymbol{b}_e$. Determination of $\boldsymbol{b}_e$ employs the definition of J_2 (Mises) plasticity in terms of a yield function, flow rule, Kuhn–Tucker loading/unloading conditions, and the consistency condition. The stress calculation procedure (radial return algorithm) is described by Simo and Hughes [63] (Box 9.1, page 319).

Appendix B Truncation Error Analysis of the Rhie-Chow Stabilisation Term

On a 1-D uniform mesh with spacing Δx and unity areas, the Rhie-Chow term stabilisation term (Equation 27) for an internal cell P (no boundary faces) becomes

$$\begin{aligned}\mathcal{D}_P^{\text{Rhie-Chow}} &= \sum_{f_i \in \mathcal{F}_p^{\text{int}}} \alpha \bar{K} \left[|\boldsymbol{\Delta}_{f_i}| \frac{\boldsymbol{u}_{N_{f_i}} - \boldsymbol{u}_P}{|\boldsymbol{d}_f|} - \boldsymbol{\Delta}_{f_i} \cdot (\boldsymbol{\nabla u})_f \right] |\boldsymbol{\Gamma}_{f_i}| \\ &= \alpha \bar{K} \left[\frac{\boldsymbol{u}_E - \boldsymbol{u}_P}{\Delta x} - (\boldsymbol{\nabla u})_e \right] + \alpha \bar{K} \left[\frac{\boldsymbol{u}_W - \boldsymbol{u}_P}{\Delta x} + (\boldsymbol{\nabla u})_w \right]\end{aligned} \tag{B13}$$

where E and W indicate the east and west neighbour cell centre values, EE and WW are the far east and west neighbour cell centre values, and e and w indicate east and west face values; α and $\bar{K}$ are assumed uniform, and $|\boldsymbol{\Gamma}_{f_i}|$ is assumed equal to unity. The face gradients are calculated as

$$\begin{aligned}(\boldsymbol{\nabla u})_e &= \frac{1}{2} \left[(\boldsymbol{\nabla u})_P + (\boldsymbol{\nabla u})_E \right] \\ (\boldsymbol{\nabla u})_w &= \frac{1}{2} \left[(\boldsymbol{\nabla u})_W + (\boldsymbol{\nabla u})_P \right]\end{aligned} \tag{B14}$$

where the cell centre gradients are calculated as

$$\begin{aligned}(\boldsymbol{\nabla u})_W &= \frac{\boldsymbol{u}_P - \boldsymbol{u}_{WW}}{2\Delta x} \\ (\boldsymbol{\nabla u})_P &= \frac{\boldsymbol{u}_E - \boldsymbol{u}_W}{2\Delta x} \\ (\boldsymbol{\nabla u})_E &= \frac{\boldsymbol{u}_{EE} - \boldsymbol{u}_P}{2\Delta x}\end{aligned} \tag{B15}$$

The final Rhie-Chow term becomes

$$\begin{aligned}\mathcal{D}_P^{\text{Rhie-Chow}} &= \alpha \bar{K} \left\{ \frac{\boldsymbol{u}_E - \boldsymbol{u}_P}{\Delta x} - \frac{1}{2} \left[(\boldsymbol{\nabla u})_P + (\boldsymbol{\nabla u})_E \right] + \frac{\boldsymbol{u}_W - \boldsymbol{u}_P}{\Delta x} + \frac{1}{2} \left[(\boldsymbol{\nabla u})_W + (\boldsymbol{\nabla u})_P \right] \right\} \\ &= \alpha \bar{K} \left\{ \frac{\boldsymbol{u}_E - \boldsymbol{u}_P}{\Delta x} - \frac{1}{2} (\boldsymbol{\nabla u})_E + \frac{\boldsymbol{u}_W - \boldsymbol{u}_P}{\Delta x} + \frac{1}{2} (\boldsymbol{\nabla u})_W \right\} \\ &= \alpha \bar{K} \left[\frac{\boldsymbol{u}_E - \boldsymbol{u}_P}{\Delta x} - \frac{\boldsymbol{u}_{EE} - \boldsymbol{u}_P}{4\Delta x} + \frac{\boldsymbol{u}_W - \boldsymbol{u}_P}{\Delta x} + \frac{\boldsymbol{u}_P - \boldsymbol{u}_{WW}}{4\Delta x} \right] \\ &= \frac{\alpha \bar{K}}{4\Delta x} \left[-\boldsymbol{u}_{WW} + 4\boldsymbol{u}_W - 6\boldsymbol{u}_P + 4\boldsymbol{u}_E - \boldsymbol{u}_{EE} \right]\end{aligned} \tag{B16}$$

A local truncation analysis can be performed using the following truncated Taylor series about the true solution ($\boldsymbol{U}$) and its gradients ($(\boldsymbol{\nabla U})_P$, $\left(\boldsymbol{\nabla}^2 \boldsymbol{U}\right)_P$, ...) at P:

$$\begin{aligned}\boldsymbol{u}_{WW} &= \boldsymbol{U}_P - 2\Delta x (\boldsymbol{\nabla U})_P + \frac{4\Delta x^2}{2!} \left(\boldsymbol{\nabla}^2 \boldsymbol{U}\right)_P - \frac{8\Delta x^3}{3!} \left(\boldsymbol{\nabla}^3 \boldsymbol{U}\right)_P + O(\Delta x^4) \\ \boldsymbol{u}_W &= \boldsymbol{U}_P - \Delta x (\boldsymbol{\nabla U})_P + \frac{\Delta x^2}{2!} \left(\boldsymbol{\nabla}^2 \boldsymbol{U}\right)_P - \frac{\Delta x^3}{3!} \left(\boldsymbol{\nabla}^3 \boldsymbol{U}\right)_P + O(\Delta x^4)\end{aligned}$$

$$
\begin{aligned}
\boldsymbol{u}_P &= \boldsymbol{U}_P \\
\boldsymbol{u}_E &= \boldsymbol{U}_P + \Delta x \left(\boldsymbol{\nabla U}\right)_P + \frac{\Delta x^2}{2!}\left(\boldsymbol{\nabla}^2\boldsymbol{U}\right)_P + \frac{\Delta x^3}{3!}\left(\boldsymbol{\nabla}^3\boldsymbol{U}\right)_P + O(\Delta x^4) \\
\boldsymbol{u}_{EE} &= \boldsymbol{U}_P + 2\Delta x \left(\boldsymbol{\nabla U}\right)_P + \frac{4\Delta x^2}{2!}\left(\boldsymbol{\nabla}^2\boldsymbol{U}\right)_P + \frac{8\Delta x^3}{3!}\left(\boldsymbol{\nabla}^3\boldsymbol{U}\right)_P + O(\Delta x^4)
\end{aligned} \tag{B17}
$$

where $O(\Delta x^4)$ are higher-order terms with a leading term proportional to Δx^4.

Substituting the expressions above (Equations B17) for the true solution into Equation B16 results in

$$
\begin{aligned}
\mathcal{D}_P^{\text{Rhie-Chow}} &= \frac{\alpha\bar{K}}{4\Delta x}\left[-\boldsymbol{u}_{WW} + 4\boldsymbol{u}_W - 6\boldsymbol{u}_P + 4\boldsymbol{u}_E - \boldsymbol{u}_{EE}\right] \\
&= \frac{\alpha\bar{K}}{4\Delta x}\Bigg[-\boldsymbol{U}_P + 2\Delta x\left(\boldsymbol{\nabla U}\right)_P - 4\frac{\Delta x^2}{2!}\left(\boldsymbol{\nabla}^2\boldsymbol{U}\right)_P + 8\frac{\Delta x^3}{3!}\left(\boldsymbol{\nabla}^3\boldsymbol{U}\right)_P \\
&\qquad +4\boldsymbol{U}_P - 4\Delta x\left(\boldsymbol{\nabla U}\right)_P + 4\frac{\Delta x^2}{2!}\left(\boldsymbol{\nabla}^2\boldsymbol{U}\right)_P - 4\frac{\Delta x^3}{3!}\left(\boldsymbol{\nabla}^3\boldsymbol{U}\right)_P \\
&\qquad -6\boldsymbol{U}_P \\
&\qquad +4\boldsymbol{U}_P + 4\Delta x\left(\boldsymbol{\nabla U}\right)_P + 4\frac{\Delta x^2}{2!}\left(\boldsymbol{\nabla}^2\boldsymbol{U}\right)_P + 4\frac{\Delta x^3}{3!}\left(\boldsymbol{\nabla}^3\boldsymbol{U}\right)_P \\
&\qquad -\boldsymbol{U}_P - 2\Delta x\left(\boldsymbol{\nabla U}\right)_P - 4\frac{\Delta x^2}{2!}\left(\boldsymbol{\nabla}^2\boldsymbol{U}\right)_P + 8\frac{\Delta x^3}{3!}\left(\boldsymbol{\nabla}^3\boldsymbol{U}\right)_P + O(\Delta x^4)\Bigg] \\
&= \frac{\alpha\bar{K}}{4\Delta x}\left[16\frac{\Delta x^3}{3!}\left(\boldsymbol{\nabla}^3\boldsymbol{U}\right)_P + O(\Delta x^4)\right] \\
&= \frac{2}{3}\alpha\bar{K}\Delta x^2\left(\boldsymbol{\nabla}^3\boldsymbol{U}\right)_P + O(\Delta x^3)
\end{aligned} \tag{B18}
$$

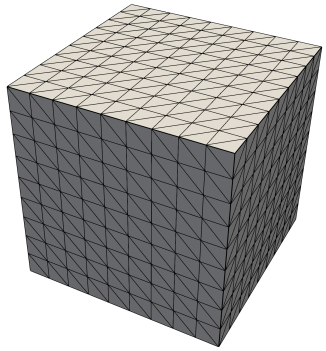
showing the leading truncation error term to reduce at a second order rate as Δx is reduced.

Appendix C Body Force for the Method of Manufactured Solutions Case

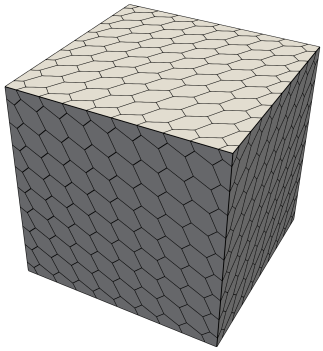
The body force for the manufactured solution case is [45]:

$$\boldsymbol{f}_b = \begin{pmatrix} \lambda\left[8a_y\pi^2\cos(4\pi x)\cos(2\pi y)\sin(\pi z)\right. \\ \quad +4a_z\pi^2\cos(4\pi x)\cos(\pi z)\sin(2\pi y) \\ \quad \left.-16a_x\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ +\mu\left[8a_y\pi^2\cos(4\pi x)\cos(2\pi y)\sin(\pi z)\right. \\ \quad +4a_z\pi^2\cos(4\pi x)\cos(\pi z)\sin(2\pi y) \\ \quad \left.-5a_x\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ -32a_x\mu_\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ \\ \lambda\left[8a_x\pi^2\cos(4\pi x)\cos(2\pi y)\sin(\pi z)\right. \\ \quad +2a_z\pi^2\cos(2\pi y)\cos(\pi z)\sin(4\pi x) \\ \quad \left.-4a_y\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ +\mu\left[8a_x\pi^2\cos(4\pi x)\cos(2\pi y)\sin(\pi z)\right. \\ \quad +2a_z\pi^2\cos(2\pi y)\cos(\pi z)\sin(4\pi x) \\ \quad \left.-17a_y\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ -8a_y\mu_\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z) \\ \\ \lambda\left[4a_x\pi^2\cos(4\pi x)\cos(\pi z)\sin(2\pi y)\right. \\ \quad +2a_y\pi^2\cos(2\pi y)\cos(\pi z)\sin(4\pi x) \\ \quad \left.-a_z\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ +\mu\left[4a_x\pi^2\cos(4\pi x)\cos(\pi z)\sin(2\pi y)\right. \\ \quad +2a_y\pi^2\cos(2\pi y)\cos(\pi z)\sin(4\pi x) \\ \quad \left.-20a_z\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z)\right] \\ -2a_z\mu_\pi^2\sin(4\pi x)\sin(2\pi y)\sin(\pi z) \end{pmatrix} \tag{C19}$$

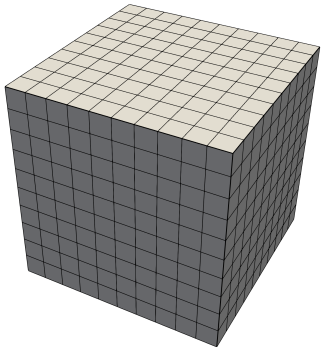
Appendix D Meshes Used with the Method of Manufactured Solutions



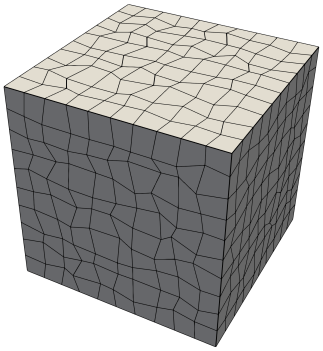
(a) Regular tetrahedral mesh with 4 374 cells



(b) Regular polyhedral mesh with 1 000 cells



(c) Regular hexahedral mesh with 1 000 cells



(d) Distorted hexahedral mesh with 1 000 cells

Fig. D1 Meshes used in the method of manufactured solutions case

References

[1] Trangenstein, J.A., Colella, P.: A higher-order Godunov method for modeling finite deformation in elastic-plastic solids. Communications on Pure and Applied Mathematics **44**, 41–100 (1991)

[2] Kluth, G., Després, B.: Discretization of hyperelasticity on unstructured mesh with a cell-centered Lagrangian scheme. Journal of Computational Physics **229**, 9092–9118 (2010)

[3] Lee, C.H., Gil, A.J., Bonet, J.: Development of a cell centred upwind finite volume algorithm for a new conservation law formulation in structural dynamics. Computers & Structures **118**, 13–38 (2013)

[4] Haider, J., Lee, C.H., Gil, A.J., Bonet, J.: A first order hyperbolic framework for large strain computational solid dynamics: An upwind cell centred total Lagrangian scheme. International Journal for Numerical Methods in Engineering **109**, 407–456 (2017)

[5] 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

[6] Fryer, Y.D., Bailey, C., Cross, M., Lai, C.-H.: A control volume procedure for solving the elastic stress-strain equations on an unstructured mesh. Applied Mathematical Modelling **15**, 639–645 (1991)

[7] Demirdžić, I., Muzaferija, S.: Numerical method for coupled fluid flow, heat transfer and stress analysis using unstructured moving meshes with cells of arbitrary topology. Computer Methods in Applied Mechanics and Engineering **125**(1), 235–255 (1995) https://doi.org/10.1016/0045-7825(95)00800-G

[8] Jasak, H., Weller, H.G.: Application of the finite volume method and unstructured meshes to linear elasticity. International Journal for Numerical Methods in Engineering **48**(2), 267–287 (2000) https://doi.org/10.1002/(SICI)1097-0207(20000520)48:2⟨267::AID-NME884⟩3.0.CO;2-Q https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291097-0207%2820000520%2948%3A2%3C267%3A%3AAID-NME884%3E3.0.CO%3B2-Q

[9] 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

[10] 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

[11] 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

[12] Batistić, I., Cardiff, P., Tuković, Ž.: A finite volume penalty based segment-to-segment method for frictional contact problems. Applied Mathematical Modelling **101**, 673–693 (2022) https://doi.org/10.1016/j.apm.2021.09.009

[13] Das, S., Mathur, S.R., Murthy, J.Y.: An Unstructured Finite-Volume Method for Structure–Electrostatics Interactions in MEMS. Numerical Heat Transfer, Part B: Fundamentals **60**(6), 425–451 (2011) https://doi.org/10.1080/10407790.2011.628252 https://doi.org/10.1080/10407790.2011.628252

[14] Cardiff, P., Tuković, Ž., Jasak, H., Ivanković, A.: A block-coupled finite volume methodology for linear elasticity and unstructured meshes. Computers & Structures **175**, 100–122 (2016) https://doi.org/10.1016/j.compstruc.2016.07.004

[15] 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

[16] 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

[17] 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

[18] 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

[19] 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

[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] Nejat, A., Ollivier-Gooch, C.: A high-order accurate unstructured finite volume Newton–Krylov algorithm for inviscid compressible flows. Journal of Computational Physics **227**(4), 2582–2609 (2008) https://doi.org/10.1016/j.jcp.2007.11.011

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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

[28] 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

[29] 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

[30] Badcock, K.J., Gaitonde, A.L.: An unfactored implicit moving mesh method for the two-dimensional unsteady N-S equations. International Journal for Numerical Methods in Fluids **23**(6), 607–631 (1996) https://doi.org/10.1002/(SICI)1097-0363(19960930)23:6⟨607::AID-FLD433⟩3.0.CO;2-E . First published: 30 September 1996

[31] 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

[32] Nishikawa, H., Nakashima, Y., Watanabe, N.: Effects of high-frequency damping on iterative convergence of implicit viscous solver. Journal of Computational Physics **348**, 66–81 (2017) https://doi.org/10.1016/j.jcp.2017.07.021

[33] 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)

[34] Jasak, H.: Error analysis and estimation for the finite volume method with applications to fluid flows. PhD thesis, Imperial College London (University of London) (1996)

[35] 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

[36] Jasak, H.: Finite Volume Discretisation with Polyhedral Cell Support: What is Discretisation? Presentation at NUMAP-FOAM Summer School, Zagreb, Croatia, 2-15 September 2009 (2011). https://www.slideshare.net/slideshow/57969246-finitevolume/16128767

[37] Syrakos, A., Oxtoby, O., Villiers, E., Varchanis, S., Dimakopoulos, Y., Tsamopoulos, J.: A unification of least-squares and green–gauss gradients under a common projection-based gradient reconstruction framework. Mathematics and Computers in Simulation **205**, 108–141 (2023) https://doi.org/10.1016/j.matcom.2022.09.008

[38] 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

[39] 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

[40] 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

[41] Jsvobs, D.A.H.: A Generalization of the Conjugate-Gradient Method to Solve Complex Systems. IMA Journal of Numerical Analysis **6**(4), 447–452 (1986) https://doi.org/10.1093/imanum/6.4.447 https://academic.oup.com/imajna/article-pdf/6/4/447/2181350/6-4-447.pdf

[42] Falgout, R.D., Yang, U.M.: hypre: A library of high performance preconditioners. In: Sloot, P.M.A., Hoekstra, A.G., Tan, C.J.K., Dongarra, J.J. (eds.) Computational Science — ICCS 2002, pp. 632–641. Springer, Berlin, Heidelberg (2002)

[43] Amestoy, P.R., Duff, I.S., Koster, J., L'Excellent, J.-Y.: A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM Journal on Matrix Analysis and Applications **23**(1), 15–41 (2001)

[44] Amestoy, P.R., Buttari, A., L'Excellent, J.-Y., Mary, T.: Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures. ACM Transactions on Mathematical Software **45**, 2–1226 (2019)

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

[46] 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

[47] Hitchings, D., Davies, G.A.O., Kamoulakos, A.: Linear Static Benchmarks. International Association for the Engineering Analysis Community & National Agency for Finite Element Methods & Standards (NAFEMS), Glasgow, UK (1987)

[48] Demirdžić, I., Muzaferija, S., Perić, M.: Advances in computation of heat transfer, fluid flow, and solid body deformation using finite volume approaches. In: W. J. Minkowycz, E.M.S. (ed.) Advances in Numerical Heat Transfer, Chapter 2, pp. 59–96. Taylor & Francis, London, United Kingdom (1997)

[49] Cardiff, P., Tuković, Jasak, H., Ivanković, A.: A block-coupled finite volume methodology for linear elasticity and unstructured meshes. Computers & Structures **175**, 100–122 (2016) https://doi.org/10.1016/j.compstruc.2016.07.004

[50] Rohatgi, A.: WebPlotDigitizer. https://automeris.io

[51] 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

[52] 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

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

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

[55] 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

[56] Simo, J.C., Armero, F.: Geometrically non-linear enhanced strain mixed methods and the method of incompatible modes. International Journal for Numerical Methods in Engineering **33**(7), 1413–1449 (1992) https://doi.org/10.1002/nme.1620330705 https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.1620330705

[57] Sá, J.M.A., Areias, P.M.A., Natal Jorge, R.M.: Quadrilateral elements for the solution of elasto-plastic finite strain problems. International Journal for Numerical Methods in Engineering **51**(8), 883–917 (2001) https://doi.org/10.1002/nme.183 https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.183

[58] Tuković, Ž., Jasak, H.: Updated Lagrangian finite volume solver for large deformation dynamic response of elastic body. Transactions of FAMENA **31**(1), 55–70 (2007)

[59] Turek, S., Hron, J.: Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In: Bungartz, H.-J., Schäfer, M. (eds.) Fluid-Structure Interaction: Modelling, Simulation, Optimisation. Lecture Notes in Computational Science and Engineering, vol. 53, pp. 371–385. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/3-540-34596-5_20

[60] Ferziger, J.H., Peric, M.: Computational Methods for Fluid Dynamics, 3^{rd} edn. Springer, Berlin, Germany (2002)

[61] Degroote, J., Bathe, K.-J., Vierendeels, J.: Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction. Computers & Structures **87**(11-12), 793–801 (2009) https://doi.org/10.1016/j.compstruc.2009.01.006

[62] Pellegrini, F.: Scotch and PT-Scotch graph partitioning software: An overview. In: Naumann, U., Schenk, O. (eds.) Combinatorial Scientific Computing, pp. 373–406. Chapman and Hall/CRC, London, United Kingdom (2012). Chap. 14. https://doi.org/10.1201/b11644-15 . https://hal.inria.fr/hal-00770422

[63] Simo, J.C., Hughes, T.J.R.: Computational Inelasticity vol. 7. Springer, New York (1998)