14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020)
Virtual Congress: 11—15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

SIMULATING COMPRESSIBLE AND NEARLY-INCOMPRESSIBLE LINEAR ELASTICITY USING AN EFFICIENT PARALLEL SCALABLE MATRIX-FREE HIGH-ORDER FINITE ELEMENT METHOD

Arash Mehraban[1], Jeremy Thompson[2], Jed Brown[1], Richard Regueiro[3], Valeria Barra[1,4], Henry Tufo[1],

[1] University of Colorado Boulder, Department of Computer Science
arash.mehraban@colorado.edu
jed.brown@colorado.edu
henry.tufo@colorado.edu

[2] University of Colorado Boulder, Department of Applied Mathematics
jeremy.thompson@colorado.edu

[3] University of Colorado Boulder, Department of Civil, Environmental and Architectural Engineering
richard.regueiro@colorado.edu

[4] California Institute of Technology
valeria@caltech.edu

Key words: Compressible and Nearly-Incompressible 3D Linear Isotropic Elasticity, High-order FEM, Matrix-Free, P-multigrid Preconditioning, High-Performance and Parallel Computing

Abstract. We examine a residual and matrix-free Jacobian formulation of compressible and nearly incompressible ($\nu \to 0.5$) displacement-only linear isotropic elasticity with high-order hexahedral finite elements. A matrix-free p-multigrid method is combined with algebraic multigrid on the assembled sparse coarse grid matrix to provide an effective preconditioner. The software is verified with the method of manufactured solutions. We explore convergence to a predetermined L^2 error of 10^{-4}, 10^{-5} and 10^{-6} for the compressible case and 10^{-4}, 10^{-5} for the nearly-incompressible cases, as the Poisson's ratio approaches 0.5, based upon grid resolution and polynomial order. We compare our results against results obtained from C3D20H mixed/hybrid element available in the commercial finite element software ABAQUS that is quadratic in displacement and linear in pressure. We determine, for the same problem size, that our matrix-free approach for displacement-only implementation is faster and more efficient for quadratic elements compared to the C3D20H element from ABAQUS that is specially designed to handle nearly-incompressible and incompressible elasticity problems. However, as we approach the near incompressibility limit, the number of Conjugate Gradient iterations required to achieve the desired solution increases significantly.

Notation Boldface denotes vectors and tensors in symbolic notation. Unless otherwise indicated, all vector and tensor products in symbolic form are assumed to be inner products, such as $\boldsymbol{v}\boldsymbol{v} = v_i v_i$, $(\boldsymbol{a}\boldsymbol{b})_{ik} = a_{ij}b_{jk}$ and $\boldsymbol{a} : \boldsymbol{b} = a_{ij}b_{ij}$, where repeated indices denote a sum over those indices. Cartesian coordinates are assumed. The symbol tr(•) is the trace operator, such that $\mathrm{tr}(\boldsymbol{\sigma}) = \sigma_{ii}$. The symbol $\boldsymbol{I}$ is the unit tensor, i.e., $(\boldsymbol{I})_{ij} = \delta_{ij}$ where δ_{ij} is the Kronecker delta operator.

1 Introduction

In the solid mechanics framework, it is well-known that for nearly-incompressible isotropic elastic materials, linear finite elements perform poorly when the standard displacement finite element (FE) formulation is utilized due to volumetric locking of the strain [15, 31]. In order to overcome the locking phenomenon [31] numerous frameworks using mixed/hybrid formulations have been proposed [23, 9, 12]. Most mixed/hybrid formulations for linear elasticity at small strain lead to a system of equations where two variables of displacement and pressure are treated separately via an additive split of the volumetric and deviatoric parts of strain [16, 24]. Some displacement-only formulations, such as reduced and selective integration, have been developed in the literature. Their equivalence to mixed/hybrid methods has been established and they have proven to be successful in handling near-incompressiblity limits for linear isotropic elasticity problems [21]. Other alternatives to mixed/hybrid formulations have been adopted in the literature, e.g., the penalty method [15, 10, 31] has been applied to linearly elastic [28] and viscoelastic materials [6]. Moreover, high-order finite elements may be utilized to overcome element locking in the nearly-incompressible regime and still avoid a mixed/hybrid formulation that is more computationally expensive, compared to a displacement-only one.

However, high-order finite element methods are often avoided due to significant cost of assembling a global Jacobian matrix with high memory requirements that lead to expensive solve times [7]. Matrix-free formulations [7, 19] offer the benefits of high-order finite element methods with more efficient memory usage. Using tensor product basis evaluation can further improve the performance of high-order methods, as they have favorable storage and work estimates of $O(mp^d)$ and $O(mp^{d+1})$, respectively, for discretizations in R^d with m elements of order p [11]. Moreover, the tensor-product-based operator evaluation can be cast as matrix-matrix products. Iterative solvers, such as Krylov subspace-based methods, only require the result of matrix-vector products rather than an assembled matrix, which allows these methods to take advantage of the computational efficiencies offered by matrix-free formulations. Therefore, storage of large, sparse matrices may be avoided with iterative solvers.

Krylov methods require preconditioning [27] to efficiently and reliably solve large-scale elliptic problems. Geometric multigrid is a robust preconditioning technique that is an attractive choice for structured meshes and has been considered in numerous studies with finite difference methods and finite elements with tensor product bases on CPUs and GPUs [22, 30, 20]. *P*-multigrid, developed by Rønquist and Patera [26], is a version of geometric multigrid based on coarsening by decreasing the order of the bases in high-order or spectral finite elements rather than coarsening by aggregating elements. This technique is a natural fit for problems on unstructured meshes, and convergence only weakly depends on the polynomial basis degree, p, if properly implemented. Computational costs of the discretization scheme may be affected by polynomial order p and element size h, among other factors [29, 8] such as adaptive r-refinement of the mesh [25].

This paper presents an efficient matrix-free high-order finite element discretization of the linear 3D elasticity problem with p-multigrid preconditioning for which tri-quadratic (Q_2) discretization solve time is roughly the same as for a tri-linear discretization (Q_1), for the same mesh size. We aim to answer the following questions about the proposed method: 1) *Given the proposed implementation, if a target accuracy in the error is desired, what combination of mesh-refinement and polynomial order requires the least cost* *in the compressible and nearly-incompressible cases using the standard displacement-only formulation? 2) What combination of mesh-refinement and polynomial order achieves the desired error tolerance the least amount of time?* 3) *Is the high-order displacement-only formulation remain competitive as the Poisson's ratio approaches the near-incompressibility limit?*

Compressible linear isotropic elasticity (e.g., $\nu = 0.3$) is appropriate for modeling metals in their elastic regime, such as in serviceability studies to analyze the stress intensity factor at a crack tip to determine if the crack will propagate and thus estimate the life cycle of the metallic component. Nearly-incompressible linear isotropic elasticity ($\nu \to 0.5$) is appropriate for modeling small deformations (small strain and small rotation) of rubber-like materials such as solid rock propellant binders [13]. When metals or rubber-like materials are loaded beyond their small strain linear elastic limit, then large deformation hyperelasticity or hyper-elasto-plasticity constitutive models are needed, which are beyond the scope of this paper. Therefore, in addition to answering the questions raised above, we investigate convergence of the high-order FE in both compressible and near-incompressible cases. In the proposed implementation, residual and Jacobian evaluations are performed with tensor product basis evaluation using libCEED [3], and the parallel computational toolkit PETSc [5] is employed for the linear solver and utilizing AMG as preconditioner for the coarse-grid solve. In the nearly incompressible cases, the results and performance of the proposed algorithm are compared with those using mixed/hybrid finite elements in the ABAQUS software package that uses assembled sparse representation of the high-order Jacobian matrix [1].

The rest of this article is organized as follows: In section §2, we present the constitutive model for the compressible 3D linear isotropic elasticity problem its variational form. In section §3, the details of a preconditioning technique used to accelerate convergence are discussed. In section §4, we discuss our numerical results and a comparison of our results with results from ABAQUS. Finally, in section §5 we summarize our observations and conclusions for this paper.

2 Compressible Linear Elasticity Problem

For isotropic linear elastic materials, we consider the stored strain energy function as,

$$\Phi(\boldsymbol{\varepsilon}) = \frac{\lambda}{2}\,\mathrm{tr}(\boldsymbol{\varepsilon})^2 + \mu\boldsymbol{\varepsilon} : \boldsymbol{\varepsilon} \tag{1}$$

Therefore, its stress-strain relationship is given by its derivative with respect to strain,

$$\boldsymbol{\sigma} = \frac{\partial\Phi(\boldsymbol{\varepsilon})}{\partial\boldsymbol{\varepsilon}} = \lambda\,\mathrm{tr}(\boldsymbol{\varepsilon})\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}, \tag{2}$$

where in the constitutive relation (2), $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ are stress and strain tensors, respectively, λ and μ are the Lamé parameters, and $\boldsymbol{I}$ is a 3×3 identity matrix. In addition, the small strain tensor is,

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right). \tag{3}$$

The strong form of the static balance of linear momentum is given by the following,

$$\begin{cases} \nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{g} = \boldsymbol{0} \\ \boldsymbol{u} = \boldsymbol{u}_0 \\ \boldsymbol{\sigma}\cdot\boldsymbol{n} = \bar{\boldsymbol{t}} \end{cases} \tag{4}$$

where the functions $\boldsymbol{u}_0$ and $\bar{\boldsymbol{t}}$ are prescribed boundary displacement and traction, respectively, $\boldsymbol{g}$ is the body force per unit mass (such as the gravitational acceleration vector), and ρ is the mass density. Introducing weighting function $\boldsymbol{w}$ for the displacement field, the corresponding variational equation for (4) is given by,

$$\int_{\Omega} \nabla \boldsymbol{w} \colon \boldsymbol{\sigma}\, dv - \int_{\partial\Omega^{\bar{t}}} \boldsymbol{w} \cdot \bar{\boldsymbol{t}}\, da - \int_{\Omega} \boldsymbol{w} \cdot \rho \boldsymbol{g}\, dv = 0. \tag{5}$$

2.1 Residual Evaluation

Discretization of the variational equation (5) can be represented as described in [7] to facilitate matrix-free evaluation. The residual in the discrete form can be computed as,

$$\sum_e \mathcal{E}_e^T \left[(\boldsymbol{N}^e)^T \cdot \boldsymbol{W} \cdot \Lambda(f_0(\boldsymbol{u}^e, \nabla \boldsymbol{u}^e)) + \sum_{i=1}^{\text{dim}} (\boldsymbol{B}_i^e)^T \cdot \boldsymbol{W} \cdot \Lambda(f_1(\boldsymbol{u}^e, \nabla \boldsymbol{u}^e)) \right] = \boldsymbol{0} \tag{6}$$

where $\boldsymbol{N}^e$ and $\boldsymbol{B}_i^e$ are evaluations of the finite element shape functions and their derivatives at the quadrature points in x, y, and z directions, $\mathcal{E}_e$ is the element e restriction operator that separates Degrees of Freedom (DoF) based on the elements they belong to, and Λ represents pointwise function evaluation. Both $\boldsymbol{f}_0$ and $\boldsymbol{f}_1$ come from the constitutive law and its tangent where $\boldsymbol{u}^e = \boldsymbol{N}^e(\mathcal{E}_e \mathbf{d})$ and $\nabla \boldsymbol{u}^e = \sum_{I=1}^{\text{dim}} [\boldsymbol{B}_i^e(\mathcal{E}_e \mathbf{d})]$, where $\boldsymbol{d}$ is the total nodal displacement vector. The basis operators are represented as Kronecker products,

$$\begin{aligned} \mathbf{N}^e &= \hat{\boldsymbol{N}} \otimes \hat{\boldsymbol{N}} \otimes \hat{\boldsymbol{N}} \qquad & \mathbf{B}_1^e &= \hat{\boldsymbol{B}} \otimes \hat{\boldsymbol{N}} \otimes \hat{\boldsymbol{N}} \\ \mathbf{B}_2^e &= \hat{\boldsymbol{N}} \otimes \hat{\boldsymbol{B}} \otimes \hat{\boldsymbol{N}} \qquad & \mathbf{B}_3^e &= \hat{\boldsymbol{N}} \otimes \hat{\boldsymbol{N}} \otimes \hat{\boldsymbol{B}} \end{aligned} \tag{7}$$

where $\hat{\boldsymbol{N}}$ and $\hat{\boldsymbol{B}}$ are evaluations of the finite element shape functions and their derivatives at the quadrature points in 1D. Comparing the variational equation in (5) and (6) yields $\boldsymbol{f}_0 = \boldsymbol{g}$ and $\boldsymbol{f}_1 = \boldsymbol{\sigma}$, assuming $\bar{\boldsymbol{t}} = \boldsymbol{0}$.

2.2 Jacobian Evaluation

Similar to a residual evaluation, the action of the Jacobian can be computed using the notation proposed by [7] and [18]:

$$\frac{\partial \boldsymbol{F}(\boldsymbol{u})}{\partial \boldsymbol{u}} = \sum_e \mathcal{E}_e^T \left[\boldsymbol{N}^T \boldsymbol{B}^T \right] \boldsymbol{W} \begin{bmatrix} \boldsymbol{f}_{0,0} & \boldsymbol{f}_{0,1} \\ \boldsymbol{f}_{1,0} & \boldsymbol{f}_{1,1} \end{bmatrix} \begin{bmatrix} \boldsymbol{N} \\ \boldsymbol{B} \end{bmatrix} \mathcal{E}_e \tag{8}$$

where

$$\boldsymbol{f}_{i,j} = \begin{bmatrix} \dfrac{\partial \boldsymbol{f}_0}{\partial \boldsymbol{u}} & \dfrac{\partial \boldsymbol{f}_0}{\partial \nabla \boldsymbol{u}} \\[2ex] \dfrac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{u}} & \dfrac{\partial \boldsymbol{f}_1}{\partial \nabla \boldsymbol{u}} \end{bmatrix} (\boldsymbol{u}, \nabla \boldsymbol{u}) \tag{9}$$

In the small strain case, for equation (2), $\boldsymbol{f}_0$ is not a function of $\boldsymbol{u}$ or $\nabla \boldsymbol{u}$. Therefore, its derivative with respect to $\boldsymbol{u}$ or $\nabla \boldsymbol{u}$ is zero, (i.e., $\boldsymbol{f}_{0,0} = \boldsymbol{0}$ and $\boldsymbol{f}_{0,1} = \boldsymbol{0}$). On the other hand, $\boldsymbol{f}_1$ is a function of $\nabla \boldsymbol{u}$, but it is not a function of $\boldsymbol{u}$. Therefore, $\boldsymbol{f}_{1,0} = \boldsymbol{0}$ and $\boldsymbol{f}_{1,1} = \partial \boldsymbol{\sigma} / \partial \boldsymbol{\varepsilon}$; due to linearity, $\partial \boldsymbol{\sigma} / \partial \boldsymbol{\varepsilon} = \lambda \boldsymbol{I} \otimes \boldsymbol{I} + 2\mu \mathbb{I}$, or equivalently, $\boldsymbol{f}_{1,1}(\nabla \delta \boldsymbol{u}) = (\partial \boldsymbol{\sigma} / \partial \boldsymbol{\varepsilon}) : \nabla \delta \boldsymbol{u} = \lambda \operatorname{tr}[\boldsymbol{\varepsilon}(\delta \boldsymbol{u})] + 2\mu \boldsymbol{\varepsilon}(\delta \boldsymbol{u})$, which is equivalent to (2) applied to a variation $\delta \boldsymbol{u}$. While these minor simplifications are possible for linear problems in the present work, our implementation solves the problem as though it were nonlinear, and we'll continue using the corresponding terminology.

3 P-multigrid Preconditioning of Linear Elasticity Problem

Utilizing the notation of section §2 and the formulation in equation (8), we can compute the action of global Jacobian matrix on $\boldsymbol{u}$ with arbitrary user defined polynomial order. An iterative solver is required with matrix-free operators, which necessitates preconditioning, especially at high-order. With unstructured meshes, a natural hierarchy of grids does not exist, so h-multigrid can be difficult to implement. Algebraic multigrid (AMG) is suitable for low order meshes where the Jacobian matrix can be assembled. However, assembly of this matrix is prohibitively expensive for high-order meshes [14]. Therefore, we use geometric multigrid with p coarsening and utilizing AMG as the coarse grid solver. A Chebyshev polynomial smoother based upon the operator diagonal [4] is utilized in the multigrid cycle.

In p-multigrid, grid transfer operations increase or decrease the polynomial order of the element basis functions, and these operations can be implemented in a matrix-free fashion via $\boldsymbol{f}_{0,0}$ of (8) with suitable basis evaluators $\boldsymbol{N}$. The coarse-to-fine basis operation, $\boldsymbol{N}_{\text{ctof}}$, interpolates the DoFs on the nodes of a coarse grid element to the nodes of a fine grid element ($\boldsymbol{N}_{27\times 8}$ for Q_1 prolongation to Q_2, for example). The corresponding coarse and fine grid element restriction operators, $\mathcal{E}_{e,c}$ and $\mathcal{E}_{e,f}$, are used in the grid transfer operators. The operator $\boldsymbol{P} = \sum_e \mathcal{E}_{e,f}^T \boldsymbol{N}_{\text{ctof}} \mathcal{E}_{e,c}$ correctly computes the interior degrees of freedom but over-counts nodes on the interfaces between elements. We can count the multiplicity of each node on the fine grid by applying the transpose fine grid restriction to the unit vector, $\boldsymbol{\omega} = \sum_e \mathcal{E}_{e,f}^T \mathbf{1}$. Thus, the p-multigrid prolongation operator is given by,

$$\boldsymbol{P} = \sum_e \mathcal{E}_{e,f}^T \boldsymbol{\omega}^{-1} \boldsymbol{N}_{\text{ctof}} \mathcal{E}_{e,c}, \tag{10}$$

and the p-multigrid restriction operator is given by $\boldsymbol{R} = \boldsymbol{P}^T$.

4 Problem Statement and Computational Costs in Terms of Time

In this section, we provide numerical results based on using the 3D linear elastic constitutive model in equation (2) on structured box meshes using polynomials of order 1 through 4 for a range of box meshes. We consider a method of manufactured solution (MMS) on the problem's mesh where different Poisson's ratios are considered. For the MMS, we produce a contrived right hand side $\rho \boldsymbol{g}$ based on $\boldsymbol{u} = [u_1, u_2, u_3]^T$ with $u_1 = e^{2x} \sin(3y) \cos(4z)$, $u_2 = e^{3x} \sin(4y) \cos(2z)$ and $u_3 = e^{4x} \sin(2y) \cos(3z)$. In the compressible case, we consider Poisson's ratio of $\nu = 0.3$, and for the nearly-incompressible cases, ν ranges from 0.49 to 0.499999 with polynomials of order 1 through 4 on parallel platforms.

We employ PETSc to perform mesh management, domain decomposition, parallel assembly operations and handle communication over all MPI processes. In addition, a matrix-free operator is implemented using libCEED and PETSc for equation (8). Preconditioning with p-multigrid and AMG is accessed through PETSc's multigrid interface with grid-transfer and operator application at each level implemented in libCEED. Vectorized tensor-product operations for 8 element batches in the matrix-free operator evaluation on AVX-2 architecture for each local processor are performed through libCEED.

4.1 Achieving Target Error Tolerances using High-Order Polynomials on Box Meshes

The platform to run FE simulations with high-order polynomials and a range of box meshes is a two-socket AMD EPYC 7452 machine; each socket contains 32 CPU cores with base clock speed of 2.35GHz and 128MB of L3 cache. The NPS4 BIOS configuration was used and processes were bound to cores using MPICH-3.3.2 and `ch3:nemesis`. PETSc version 3.14 and libCEED version 0.7 are used to discretize and solve the formulations represented in equations 6 and 8 on a $[0,1]\times[0,1]\times[0,1]$ structured box mesh available from PETSc. We utilize the box mesh to create a family of meshes of $n\times n\times n$ elements by subdividing each side of the box. A non-homogeneous essential (Dirichlet) boundary condition is imposed on each side of the box as described in section 4.

For a given n, the resulting 3D mesh contains n^3 elements ($n\times n\times n$). Utilizing polynomials of order 1 through 4, we perform FE simulations for Poisson ratios of 0.3, 0.49, 0.49999 and 0.499999. Due to the locking phenomena, polynomial order 1 is avoided as we approach the incompressible limit. The desired achievable L^2 error tolerances in the solution of FE for $\nu = 0.3$ and $\nu = 0.49$ are 10^{-4}, 10^{-5}, and 10^{-6}, and 10^{-4} and 10^{-5} for $\nu = 0.49999$ and $\nu = 0.499999$. Therefore, for each polynomial order, the grid sizes are increased to achieve the desired error tolerances. For a given Poisson's ratio, polynomial order and grid size n, the FE simulations are repeated three times on 16, 32 and 64 CPU cores to reduce noise in the performance timing. Theoretically, it is expected that the total time to solution decreases for a scalable parallel implementation as the number of CPU cores utilized increases for a large enough problem. However, in the case of small problems, the cost of parallel communication may surpass the cost of computation as the number of CPU cores increases which translates to longer time to solution. Therefore, without any assumption about the scalability of the proposed implementation and the algorithm, we perform the simulations on a small number of CPU cores for comparison.

Hence, as we approach near-incompressibility limit, for any grid size $n < 25$ the simulations are also performed with 1 and 4 CPU cores leading to 729 simulations for the entire study. The "Total CPU time" and "L^2 Error" from each run are accumulated for analysis. The profiler available from PETSc provides the "Total Time" and "Solve Time" associated with each run but does not provide the time consumed by libCEED alone as an external package. An internal profiler is not currently available from libCEED, therefore, a bash script is used to time each simulation from start to end.

The results of the entire study in terms of L^2 error versus solve time and versus cost (in core seconds) are summarized in Figures 1 and 2 respectively. A point on these diagrams is called *Pareto optimal* if one cannot decrease error (moving down on the y axis) without increasing time (or cost) by moving to the right. The set of Pareto optimal configurations is known as the *Pareto front*, which evidently consists exclusively of higher degree finite element spaces ($p \in \{3,4\}$). The low order $p \in \{1,2\}$ cases are increasingly far from the Pareto front as ν increases.

In addition, Figure 2 represents that the minimal cost to achieve any order of accuracy occurs in the lower left pane of the figure indicating higher order p's are the most cost-effective polynomial orders to exploit in both the compressible and nearly incompressible cases. Larger problems with more DoFs to be amortized among CPU cores exhibit better strong scaling since computation time surpasses communication time. The strong scaling occurs when larger dots land on smaller dots.

We address the first question raised in the introduction section in regards to the minimal computational cost for a desired error tolerance in the solution. For each Poisson's ratio in this study, we compute the minimum amount of CPU work to reach the desired error tolerance. To do this, for each polynomial order, p, we accumulate the time spent for all n to reach the desired accuracy in a list. Then we determine the minimum of time in the list and multiply it with the number of processors ("np") that are exploited for the minimum time. Tables 1, 3, 5 and 7 summarize the minimum cost ("Min. Cost") to achieve the desired error tolerance in the FE solution for the Poisson's ratios considered in this study. In addition,

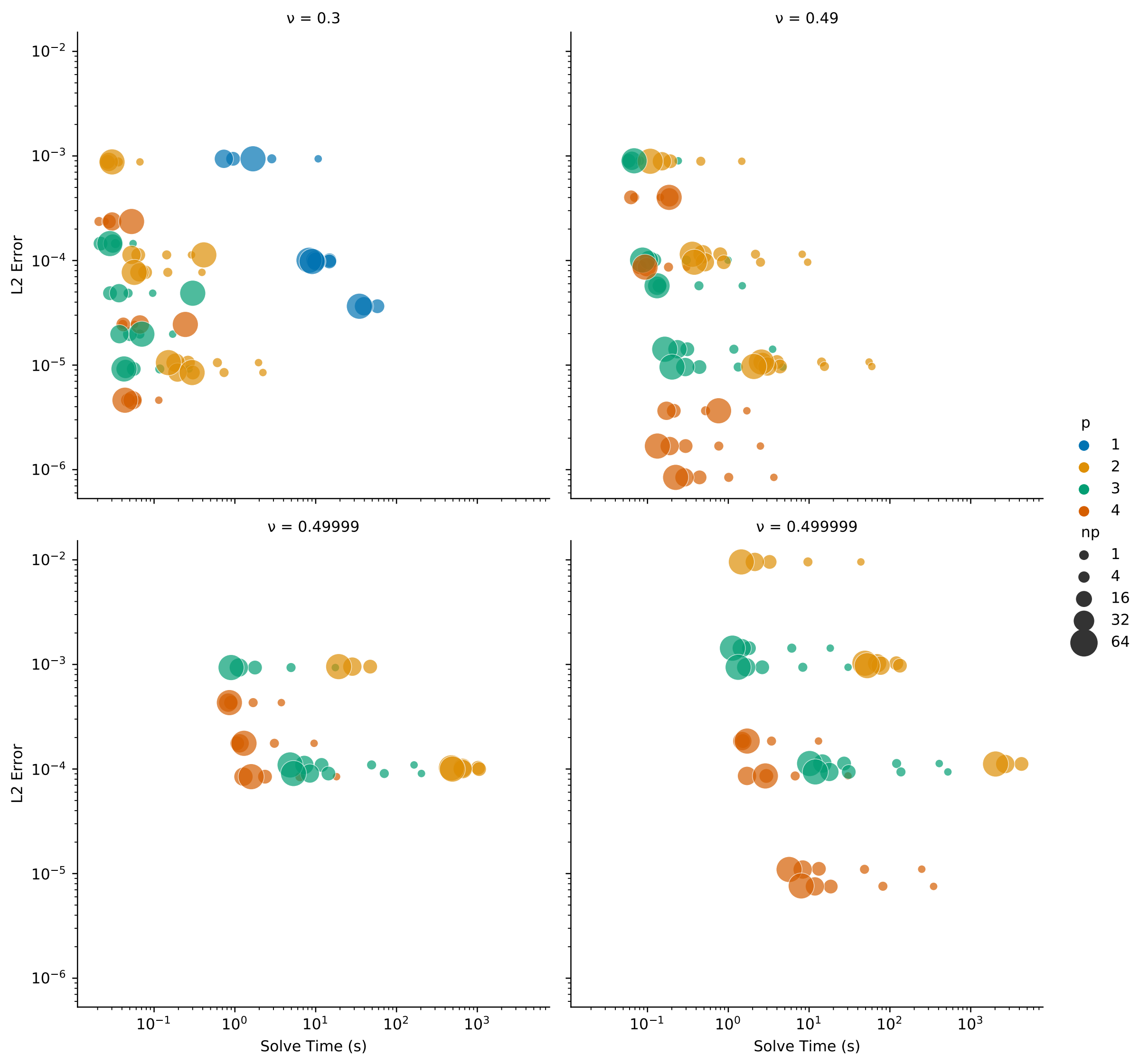

Figure 1: Error versus time for $\nu = 0.3$, $\nu = 0.49$, $\nu = 0.49999$ and $\nu = 0.499999$. The Pareto optimal configurations are toward the lower left of each pane, with $p = 3$ and $p = 4$ giving the fastest solutions for any error tolerance, while low order ($p = 1$ and $p = 2$) are increasingly further from the Pareto front for larger values of ν. Each horizontal series of the same color represents strong scaling of a given resolution h and p; cf. Figure 2 to quantify the cost of suboptimal strong scaling.

for each simulation, the smallest n that achieves the desired error tolerance is reported.

For $\nu = 0.3$ in table 1, the smallest n that achieves 10^{-4} and 10^{-5} error tolerance are 26 and 77 respectively. However, running the simulation with $n = 140$ does not improve the error tolerance. In addition, for error tolerance of 10^{-5} in the compressible case ($\nu = 0.3$), the smallest number of CPU cores utilized

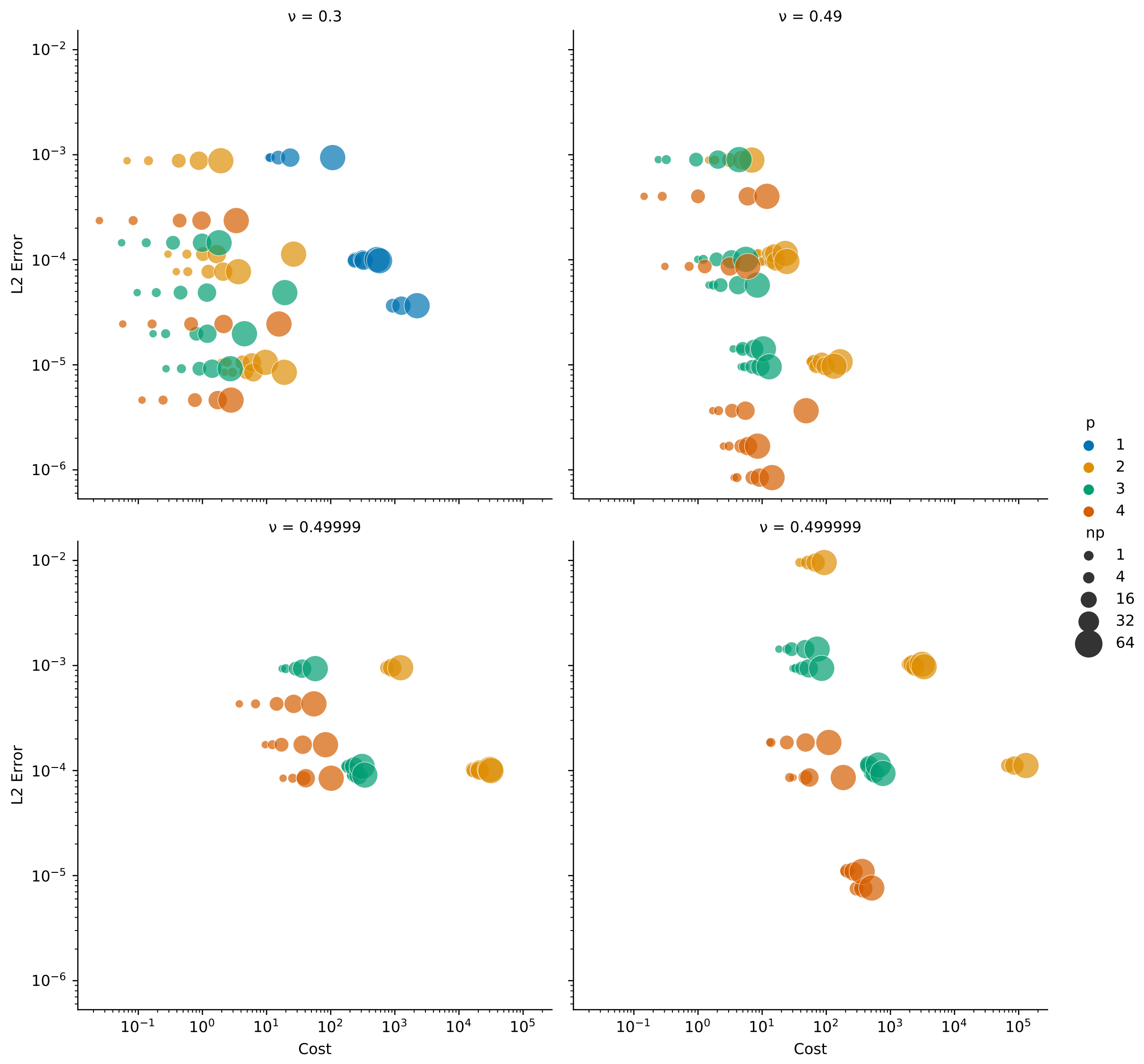

Figure 2: Error versus cost for $\nu = 0.3$, $\nu = 0.49$, $\nu = 0.49999$ and $\nu = 0.499999$. The Pareto optimal configurations are toward the lower left of each pane indicate higher order p is most cost-efficient. Each horizontal series of the same color represents a strong scaling study at fixed h and p, with perfect strong scaling manifesting when all the dots are collocated. The more expensive models tend to exhibit better strong scaling because they have more work over which to amortize the inherent communication costs, while small models are much more cost-efficient to run on a single core.

with polynomial order 1 is 16 due to the size of the problem. However, for small n, as table 1 indicates, the minimum cost to achieve the desired error tolerances are with 1 CPU core. For Poisson ratio of 0.49, $n < 45$ are also run with 1 and 4 CPU cores. As table 3 indicates, the minimum cost to achieve the desired accuracy in the L^2 error tolerance is with 1 CPU core. Tables 5 and 7 show that the minimum cost to achieve the desired error tolerances is with 1 CPU core except when polynomial order 2 is used which necessitates a much larger n (39 as apposed to 8 with polynomial order 3).

We now aim to answer the second question raised in the introduction section: what is the least amount of time to achieve a certain error tolerance in the solution given enough resources? To answer that question, we turn to tables 2, 4, 6 and 8 which summarize the best time and corresponding number of CPU cores required to achieve a certain error tolerance for different values of ν, n and p. From those tables, it can be concluded that increasing the number of CPU cores with higher order polynomials may reach a desired error tolerance faster as long as the size of n dictates a large enough problem for which the time spent with higher number of CPU cores is not spent in the communication of the parallel scheme. For example, in the compressible case from tables 2, to achieve the error tolerance of 10^{-4} with polynomial order 1, n must be increased to 26 which provides enough work for 16 CPU cores to reach the solution with the desired error tolerance in less time compared to using 1 or 4 CPU cores. However, utilizing more than 16 CPU cores for that problem size, due to communication costs, increases the amount of time to achieve the solution for the desired error tolerance. In addition, from table 8, in the incompressible case with $\nu = 0.499999$ and polynomial order 2 that necessitates an n of 39, utilizing all 64 CPU cores available in the platform reaches the solution with the desired error tolerance the fastest. However, for the same Poisson's ratio, if polynomials of order 3 or 4 are utilized, n can decrease to 8 and 5 respectively. For those small problem sizes, as table 8 indicates, 16 CPU cores can reach the desired accuracy in the solution the fastest.

Table 1: $\nu = 0.3$

	Error								
	10^{-4}			10^{-5}			10^{-6}		
p	n	Min. Cost	np	n	Min. Cost	np	n	Min. Cost	np
1	26	12.78	1	77	371.79	16*	-	-	-
2	6	0.62	1	11	1.07	1	19	3.47	1
3	4	0.71	1	5	0.74	1	7	0.96	1
4	3	1.15	1	3	1.18	1	4	1.24	1

Table 2: $\nu = 0.3$

	Error								
	10^{-4}			10^{-5}			10^{-6}		
p	n	Min. Time(s)	np	n	Min. Time(s)	np	n	Min. Time(s)	np
1	26	3.37	16	77	18.72	32	-	-	-
2	6	0.62	1	11	1.07	1	19	3.47	1
3	4	0.71	1	5	0.74	1	7	0.96	1
4	3	1.15	1	3	1.18	1	4	1.24	1

In addition, for the compressible case, we provide an h-refinement plot for polynomials of order 1 through 4 to determine the global rate of convergence of our implementation as the grid size is decreased. Figure 3 is a `log-log` plot of h versus L^2 error that represents the convergence of our implementation using grid

Table 3: $\nu = 0.49$

	Error								
	10^{-4}			10^{-5}			10^{-6}		
p	n	Min. Cost	np	n	Min. Cost	np	n	Min. Cost	np
2	13	2.26	1	23	11.43	1	41	67.40	1
3	5	0.91	1	9	2.26	1	13	5.70	1
4	3	1.27	1	7	1.43	1	9	2.90	1

Table 4: $\nu = 0.49$

	Error								
	10^{-4}			10^{-5}			10^{-6}		
p	n	Min. Time(s)	np	n	Min. Time(s)	np	n	Min. Time(s)	np
2	13	1.29	4	23	3.24	16	41	7.55	16
3	5	0.91	1	9	1.44	4	13	2.38	4
4	3	1.27	1	7	1.43	1	9	1.97	4

Table 5: $\nu = 0.49999$

	Error					
	10^{-4}			10^{-5}		
p	n	Min. Cost	np	n	Min. Cost	np
2	38	801.36	16*	97	17084.97	16*
3	8	18.25	1	17	204.86	1
4	4	4.90	1	6	19.28	1

Table 6: $\nu = 0.49999$

	Error					
	10^{-4}			10^{-5}		
p	n	Min. Time(s)	np	n	Min. Time(s)	np
2	38	32.88	32	97	511.43	64
3	8	4.08	16	17	12.67	32
4	4	3.74	16	6	5.20	16

sizes $h = 1/3$ to $h = 1/80$ for $\nu = 0.3$. All plots in this article are prepared using `matplotlib` [17] library from Python 3. We calculate the slope of the line that is the best fit in the least square sense in Figure 3. The slopes are 2.07, 4.00, 4.45, 4.75 for polynomials of order 1 through 4 respectively. PETSc's default tolerances are utilized across all runs. Therefore, in figure 3, for $p = 3$ and $p = 4$, we notice stagnation of the solver for $h = 1/40$ and $h = 1/80$ respectively. For polynomial of order 1, we notice accelerated convergence behavior for coarse meshes, therefore they are not considered in the computation of the slope.

Table 7: $\nu = 0.499999$

	Error					
	10^{-4}			10^{-5}		
p	n	Min. Cost	np	n	Min. Cost	np
2	39	2178.20	16*	-	-	-
3	8	31.11	1	17	523.15	1
4	5	14.23	1	6	31.43	1

Table 8: $\nu = 0.499999$

	Error					
	10^{-4}			10^{-5}		
p	n	Min. Time(s)	np	n	Min. Time(s)	np
2	39	63.40	64	-	-	-
3	8	5.31	16	17	21.75	32
4	5	4.24	16	6	5.82	16

Finally, we compare our results with the ABAQUS software package when $\nu = 0.499999$. ABAQUS offers a mixed/hybrid element known as C3D20H that is quadratic in displacement and linear in pressure especially developed to handle nearly incompressible problems. As Table 8 indicates, in order to achieve 10^{-4} error tolerance on the cube, we require $n = 39$. Therefore, a $[0,1] \times [0,1] \times [0,1]$ box mesh with $n = 39$ in each direction is generated in Trelis [2] software for ABAQUS. In addition, the same boundary conditions and physics as in section 4 are imposed on the imported mesh into ABAQUS. The mesh type is Hex20 Serendipity which contains a total of 398,641 mesh nodes with 64,000 Hex20 elements. ABAQUS is installed on a parallel platform with 28 cores @2.4 GHz and 128 GB RAM. For C3D20H element, ABAQUS only allows a direct solver. The total "CPU TIME"s reported by ABAQUS for this problem size utilizing C3D20H element on 28, 14 and 10 CPU cores are 4,372.3 seconds, 4,002.0 seconds and 3,858.1 seconds respectively when all outputs are turned off. Also, the "WALLCLOCK TIME" for 28, 14 and 10 CPU cores are 195 seconds, 322 seconds and 421 seconds.

Exploiting smaller than 10 CPU cores for this problem size causes ABAQUS to stop prematurely due to memory issues. Therefore, we consider the fastest time to reach a solution for the desired error tolerance with ABAQUS to be 3,858.1 seconds using 10 CPU cores. Comparing this result to those for error tolerance of 10^{-4} in table 8, we notice that our implementation can achieve the desired error tolerance for the same size problem in 2,178.20 seconds using 16 processors with polynomial order 2. However, we note that the platform chosen for the proposed implementation to run on contains faster CPU cores. In addition, according to table 8, it is possible to reach the same error tolerance using smaller h's with polynomials of order 3 and 4 on 1 CPU core which reduces the time to solution to 31.11 and 14.23 seconds, respectively.

5 Conclusions

This paper presents a matrix-free approach to finite element simulation of nearly-incompressible linear isotropic elasticity with p-multigrid preconditioning. We have investigated how varying the polynomial

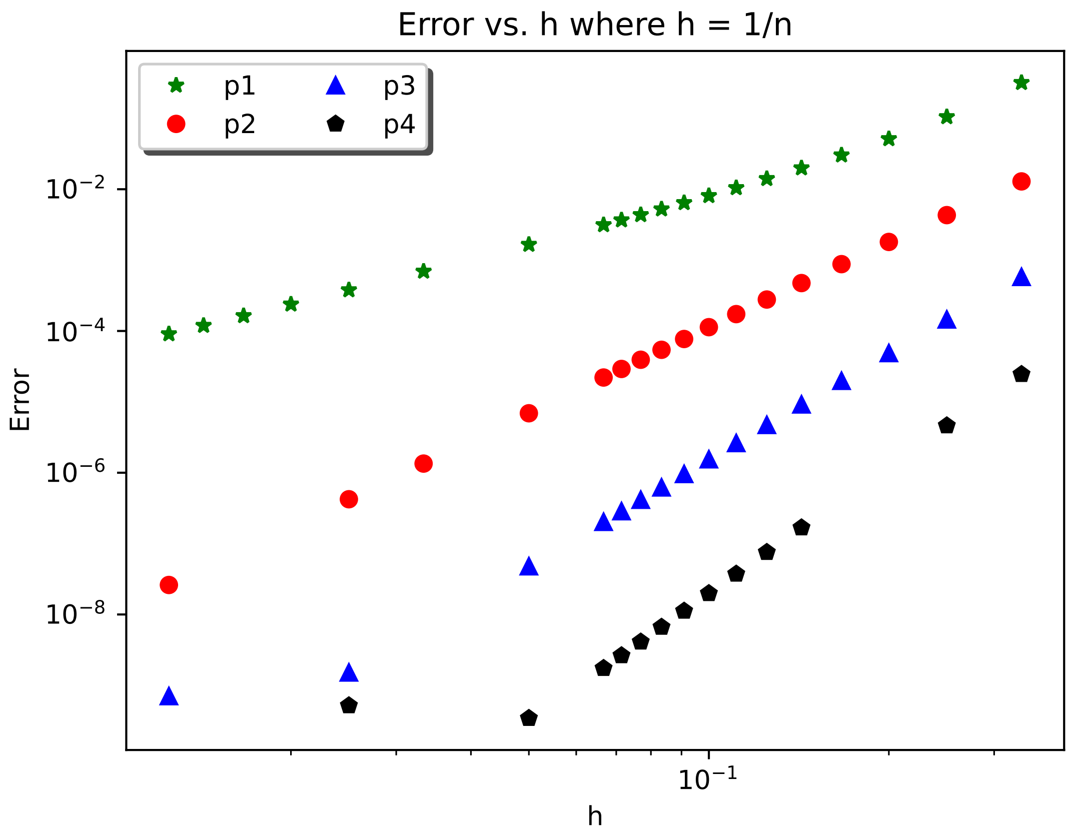

Figure 3: `log-log` plot of L^2 Error vs. h, where $h = 1/n$, for polynomials of order 1-4 in the compressible case $\nu = 0.3$. For each polynomial order, we calculate the slope of the line that is the best fit in the least square sense. The slopes are 2.07, 4.00, 4.45, 4.75 for polynomials of order 1 through 4 respectively. The stagnation for $p = 3$ and $p = 4$ are due to fixed algebraic tolerances across all runs.

order and grid size affects the time it takes to achieve a specific error tolerance using a manufactured solution over a range of Poisson's ratio from 0.3 to 0.499999. We observed the expected spectral convergence; fixing the grid size constant while increasing the polynomial order provides the fastest time to converge to the desired solution given a target error tolerance. The particular MMS chosen for this study contains only smooth functions and thus high order methods exhibit their design order of accuracy (cf. Figure 3). Real engineering problems almost always have singularities arising from reentrant corners or transitions between Dirichlet and Neumann boundaries, and thus all methods exhibit the same low order of accuracy. In many cases, high-order methods still have sufficiently better constants to justify their use in our setting, where the cost per degree of freedom decreases with increasing p (due to more structured computation and more efficient quadrature). A systematic evaluation of accuracy vs time/cost tradeoffs in the presence of natural singularities is left for future work.

For the test problems in this study, we showed the h and p convergence of our method for compressible and nearly-incompressible cases and compared it with ABAQUS's specially-designed mixed element to handle nearly-incompressible elasticity. Although our method is designed to be scalable for large number of CPU cores, it already outperforms ABAQUS's Q_2P_1 element in the nearly-incompressible cases on a small number of CPU cores.

In comparison to C3D20H element from ABAQUS, we demonstrated that displacement-only high-order FEM with matrix-free operator implementation may be utilized with nearly-incompressible linear elasticity problems as a substitute for the mixed/hybrid formulation when the mixed/hybrid implementation requires assembling matrices based on Q_2P_1 element and utilizes direct solvers. However, as we approach the incompressible limit for the linear elasticity problem, the number of CG operations increases significantly. Therefore, for future research, it is desirable to implement a two-field matrix-free approach where displacement and pressure fields are independent variables.

6 Acknowledgements

This work is supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, in support of the nation's exascale computing imperative.

References

[1] `https://www.3ds.com/products-services/simulia/products/abaqus/`.

[2] `http://csimsoft.com`.

[3] Ahmad Abdelfattah, Valeria Barra, Natalie Beams, Jed Brown, Jean-Sylvain Camier, Veselin Dobrev, Yohann Dudouit, Leila Ghaffari, Tzanio Kolev, David Medina, Thilina Rathnayake, Jeremy L Thompson, and Stanimire Tomov. libCEED User Manual, September 2020.

[4] Mark Adams, Marian Brezina, Jonathan Hu, and Ray Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *Journal of Computational Physics*, 188(2):593–610, 2003.

[5] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual, 2020.

[6] Valeria Barra, Shawn A. Chester, and Shahriar Afkhami. Numerical Simulations of Nearly Incompressible Viscoelastic Membranes. *Computers & Fluids*, 175:36–47 (2018).

[7] Jed Brown. Efficient nonlinear solvers for nodal high-order finite elements in 3D. *Journal of Scientific Computing*, 45(1-3):48–63, 2010.

[8] Chris D Cantwell, Spencer J Sherwin, Robert M Kirby, and Paul HJ Kelly. From h to p efficiently: selecting the optimal spectral/hp discretisation in three dimensions. *Mathematical Modelling of Natural Phenomena*, 6(3):84–96, 2011.

[9] Michele Chiumenti, Quino Valverde, C Agelet De Saracibar, and Miguel Cervera. A stabilized formulation for incompressible elasticity using linear displacement and pressure interpolations. *Computer methods in applied mechanics and engineering*, 191(46):5253–5264, 2002.

[10] J. Van der Zanden, G. D. C. Kuiken, A. Segal, W. J. Lindhout, and M. A. Hulsen. Numerical experiments and theoretical analysis of the flow of an elastic liquid of the upper-convected maxwell type in the presence of geometrical discontinuities. *Appl. Sci. Res.*, 42:303–318, 1985.

[11] Michel O Deville, Paul F Fischer, Paul F Fischer, EH Mund, et al. *High-order methods for incompressible fluid flow*. Number 9. Cambridge university press, 2002.

[12] Baudouin Fraeijs de Veubeke. Displacement and equilibrium models in the finite element method. *Stress analysis*, pages chapter–9, 1965.

[13] Leonard R Herrmann. Elasticity equations for incompressible and nearly incompressible materials by a variational theorem. *AIAA journal*, 3(10):1896–1900, 1965.

[14] JJ Heys, TA Manteuffel, Steve F McCormick, and LN Olson. Algebraic multigrid for higher-order finite elements. *Journal of computational Physics*, 204(2):520–532, 2005.

[15] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[16] Thomas JR Hughes, Leopoldo P Franca, and Marc Balestra. A new finite element formulation for computational fluid dynamics: V. circumventing the babuška-brezzi condition: A stable petrov-galerkin formulation of the stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, 59(1):85–99, 1986.

[17] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[18] Matthew Knepley, Jed Brown, Karl Rupp, and Barry Smith. Achieving high performance with unified residual evaluation. 09 2013.

[19] Dana A Knoll and David E Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[20] Martin Kronbichler and Karl Ljungkvist. Multigrid for matrix-free high-order finite element computations on graphics processors. 6(1), 2019.

[21] David S. Malkus and Thomas J.R. Hughes. Mixed finite element methods — reduced and selective integration techniques: A unification of concepts. *Computer Methods in Applied Mechanics and Engineering*, 15(1):63–81, 1978.

[22] D.A. May, J. Brown, and L. Le Pourhiet. A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 290(0):496–523, 2015.

[23] KB Nakshatrala, Arif Masud, and KD Hjelmstad. On finite element formulations for nearly incompressible linear elasticity. *Computational Mechanics*, 41(4):547–561, 2008.

[24] Theodore HH Pian and K Sumihara. Rational approach for assumed stress finite elements. *International Journal for Numerical Methods in Engineering*, 20(9):1685–1695, 1984.

[25] Jean-François Remacle, Joseph E Flaherty, and Mark S Shephard. An adaptive discontinuous galerkin technique with an orthogonal basis applied to compressible flow problems. *SIAM review*, 45(1):53–72, 2003.

[26] Einar M Rønquist and Anthony T Patera. Spectral element multigrid. I. formulation and numerical results. *Journal of Scientific Computing*, 2(4):389–406, 1987.

[27] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.

[28] Robert L. Taylor, Eugenio Oñate, and Pere-Andreu Ubach. *Finite Element Analysis of Membrane Structures*, pages 47–68. Springer Netherlands, Dordrecht, 2005.

[29] Peter EJ Vos, Spencer J Sherwin, and Robert M Kirby. From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low-and high-order discretisations. *Journal of Computational Physics*, 229(13):5161–5181, 2010.

[30] Yongning Zhu, Eftychios Sifakis, Joseph Teran, and Achi Brandt. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Transactions on Graphics (TOG)*, 29(2):16, 2010.

[31] Olek C Zienkiewicz, Robert L Taylor, and Jian Z Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 2005.