PRECONDITIONING MATRIX-FREE HIGH-ORDER FINITE ELEMENT OPERATORS

JEREMY L. THOMPSON*

Abstract. Finite element methods with global sparse matrices are not adequate for modeling solid mechanics problems at large scale. We are particularly interested in efficient solvers for Neo-Hookean hyperelasticity at finite strain in the incompressible regime. High-order matrix-free operators offer superior performance in comparison to assembled sparse matrices, both with respect to FLOPs needed for evaluation and the memory transfer needed for a matrix-vector product. We describe a p-multigrid preconditioner that can be efficiently formulated with matrix-free operators. Specifically, we discuss traditional smoothers such as Jacobi and Chebyshev as well as a new domain decomposition based smoother using Balancing Domain Decomposition by Constraints with subdomain solvers using approximate separable inverses based on the Fast Diagonalization Method. With these techniques, we can solve substantially larger problems when compared to assembled global sparse matrices from low-order finite elements.

1. Introduction. Solid mechanics problems are commonly solved with finite element solvers using sparse matrix representations of low order finite elements. However, assembled sparse matrices are memory bound for sufficiently large problems. We explore using high-order finite elements implemented in a matrix-free fashion with appropriate preconditioners to solve solid mechanics problems at large scale.

High-order finite element methods offer advantages over low-order finite elements; *hp* finite elements offer high accuracy and exponential convergence [7], [27], [29]. However, high-order finite elements are less common because the operator or its Jacobian rapidly loses sparsity as the order is increased. Matrix-free implementation of high-order finite elements can provide the benefits of high-order methods without relying upon matrix sparsity for efficient implementation.

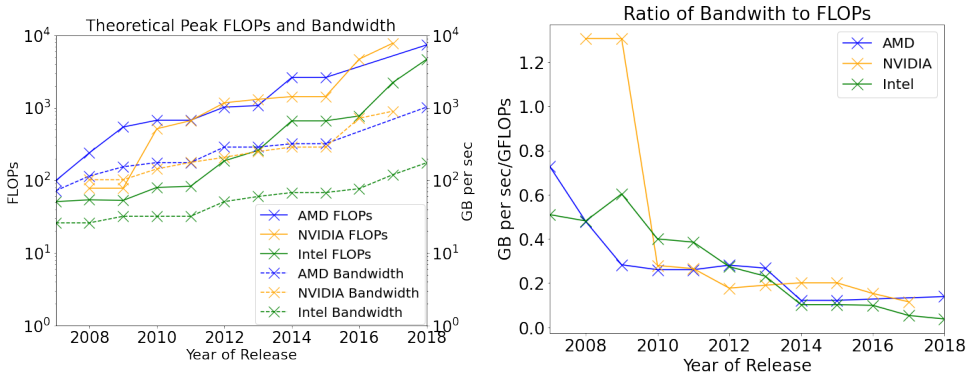
High Performance Computing (HPC) hardware improvements in memory and network bandwidth have not kept up with improvements in Floating Point Operations per second (FLOPs), as highlighted in McCalpin's invited talk an Supercomputing 2016 [25]. Under these hardware constraints, matrix-free operators offer superior performance, both with respect to the memory transfer and FLOPs needed for needed for evaluation of a matrix-vector product. However, iterative solvers are sensitive to the high condition numbers of high-order operator and require appropriate preconditioning to control total iterations and time to solution.

In Section 2, we discuss the specific hardware limitations that constrain the performance of HPC application codes. In Section 3, we provide notation to describe arbitrary PDEs and their Jacobians for matrix-free implementation. In Section 4, we provide an overview of the solid mechanics problems we are exploring. In Section 5, we discuss preconditioning techniques for these solid mechanics problems.

2. Hardware Limitations. Two key performance metrics for HPC hardware are FLOPs and memory and network bandwidth. FLOPs is the more widely popularized of these two metrics, but memory and network bandwidth is a common bottleneck in HPC application codes.

The Top 500 [26] list tracks the 500 supercomputers with the highest peak FLOPs, as measured by High-Performance Linpack (HPL) [28]. HPL measures the performance when solving random dense linear systems in double precision via LU fac-

*Department of Applied Mathematics, University of Colorado Boulder, Boulder, CO (jeremy.thompson@colorado.edu



(a) Peak FLOPs and Bandwidth

(b) Ratio Bandwidth to FLOPs

Fig. 1: System Balance

torization and measures maximum achievable FLOPs. Other benchmarks, such as High-Performance Geometric Multigrid (HPGMG) [2] and High-Performance Conjugate Gradient (HPCG) [9], measure performance based upon solving a more complex benchmark problem. The disparity between the FLOPs achieved in benchmarks such as HPGMG and HPCG and the peak FLOPs measured by HPL is partially explained by the growing gap between FLOPs and memory and network bandwidth.

Over the last thirty years, the peak FLOPs for new HPC hardware has been increasing more rapidly than memory bandwidth and network bandwidth, for both CPUs and GPUs. As discussed in McCalpin's Supercomputing 2016 invited talk [25], peak FLOPs per socket have been increasing at a rate of 50-60% per year while memory bandwidth has only been increasing at a rate of approximately 23% per year and network bandwidth has only been increasing at a rate of approximately 20% per year. FLOPs have improved twice as much as memory and network bandwidth. This problem is exacerbated by network latency, which is decreasing at a rate of approximately 20% per year, and memory latency, which is *increasing* at a rate of approximately 20% per year.

Using data from [32], we can see in Figure 1 that AMD, NVIDIA, and Intel top of the line hardware has a steady decrease in the ratio of maximum memory bandwidth to FLOPs over the last 13 years. HPC applications need to be careful to control the memory bandwidth required for their codes to better realize the FLOPs capabilities of HPC hardware.

3. Matrix-Free Finite Elements. High-order finite elements implemented in a matrix-free fashion are one way to address the memory bandwith bottlenecks for modern HPC hardware. In this section we develop notation to describe high-order matrix-free finite elements on unstructured meshes. This notation is applicable to both linear and non-linear PDEs.

3.1. Notation. The development of this notation will largely follow [4].

Let $\{X_i\}_{i=1}^{P}$ denote the Legendre-Gauss-Lobatto (LGL) nodes of degree $P-1$ on the reference interval $[-1,1]$ while $\{q_i\}_{i=1}^{Q}$ and $\{w_i\}_{i=1}^{Q}$ denote the quadrature points and quadrature weights corresponding to a Q point quadrature rule. If we

consider Lagrange basis functions $\{\phi_i\}_{i=1}^{P}$, we can construct matrices $B_{ij} = \phi_j(q_i)$, $D_{ij} = \partial_x \phi_j(q_i)$, and $W_{ij} = w_i \delta_{ij}$, representing interpolation to the quadrature points, computation of derivatives at the quadrature points, and quadrature weights, respectively.

We can define the corresponding matrices for 3D problems via tensor products

$$\begin{gathered} \mathbf{B} = B \otimes B \otimes B \quad \mathbf{D}_0 = D \otimes B \otimes B \\ \mathbf{D}_1 = B \otimes D \otimes B \quad \mathbf{D}_2 = B \otimes B \otimes D \\ \mathbf{W} = W \otimes W \otimes W. \end{gathered} \tag{3.1}$$

The basis operations 3.1 are defined on a reference element $\hat{K} = [-1, 1]^3$. In the finite element and spectral element methods, we partition the domain Ω into a set of E elements, denoted $\{K^e\}_{e=1}^{E}$ with coordinate mapping to the reference element given by $X : \hat{K} \to K^e$. The Jacobian of this mapping is given by $J_{ij} = \partial x_i / \partial X_j$, where X is the reference coordinates and x the physical coordinates. We can invert the Jacobian and compute the derivatives of the physical coordinates in the reference space at every quadrature point.

$$\mathbf{D}_i^e = \Lambda\left(\frac{\partial X_0}{\partial x_i}\right)\mathbf{D}_0 + \Lambda\left(\frac{\partial X_1}{\partial x_i}\right)\mathbf{D}_1 + \Lambda\left(\frac{\partial X_2}{\partial x_i}\right)\mathbf{D}_2 \tag{3.2}$$

where $\Lambda(X)_{ij} = X_i \delta_{ij}$ expresses pointwise multiplication of J_{ij}^{-1} at quadrature points as a diagonal matrix. With this coordinate mapping, element integration weights become $\mathbf{W}^e = W\Lambda(|J^e(q)|)$.

When using an assembled matrix to represent a finite element operator, a global assembly operator is defined as $\mathcal{E} = [\mathcal{E}^e]$, where $\mathcal{E}^e$ represents local restriction operators extracting degrees of freedom that correspond to element e from the global solution vector. Notice that these local restriction operators do not assume a structured mesh, a conforming mesh, or consistent polynomial order bases for each element.

With these definitions, we can represent the Galerkin system of equations corresponding to the weak form of arbitrary second order PDEs. The weak form of PDEs is linear in test functions and can be expressed as pointwise operations where functions of u and ∇u are contracted with v and ∇v.

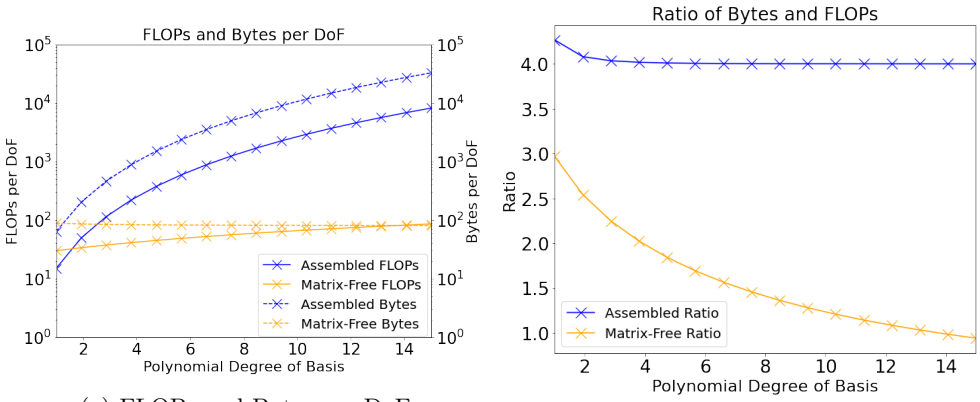
Consider the weak form of an arbitrary PDE

$$\begin{gathered} \text{find } u \in V \text{ such that for all } v \in V \\ \langle v, u \rangle = \int_\Omega v \cdot f_0(u, \nabla u) + \nabla v : f_1(u, \nabla u) = 0 \end{gathered} \tag{3.3}$$

where $\cdot$ represents contraction over fields and $:$ represents contraction over fields and spatial dimensions. The pointwise representation of the weak form given by f_0 and f_1 does not depend upon discretization choices such as geometry or polynomial degree of the bases. The corresponding Galerkin system of equations is

$$\sum_e \mathcal{E}^T \left[(\mathbf{B}^e)^T \mathbf{W}^e \Lambda\left(f_0\left(u^e, \nabla u^e\right)\right) + \sum_{i=0}^{d-1} (\mathbf{D}_i^e)^T \mathbf{W}^e \Lambda\left(f_1\left(u^e, \nabla u^e\right)\right) \right] = 0 \tag{3.4}$$

where $u^e = \mathbf{B}^e \mathcal{E}^e u$ and $\nabla u^e = \{\mathbf{D}_i^e \mathcal{E}^e u\}_{i=0}^{d-1}$. In this formulation, the element restriction operators and basis operators can represent different element geometries and different degree polynomial bases, providing a flexible description for arbitrary meshes. Furthermore, this notation can be extended to handle separate fields with different bases, such as with mixed finite element methods.



(a) FLOPs and Bytes per DoF

(b) Ratio of Bytes to FLOPs

Fig. 2: Performance per DoF

Dirichlet boundary conditions are represented in the element restriction operation by enforcing the specified values on the constrained nodes. Neumann or Robin boundary conditions are represented by adding boundary integral terms in the same form as 3.4 with appropriate basis and element restriction operators. Boundary integrals internal to the domain Ω, such as face integrals in Discontinuous Galerkin methods, can also be represented using additional terms with corresponding bases and element restrictions.

3.2. Linearization. When the PDE 3.3 is linear, the pointwise functions f_0 and f_1 are also linear and Krylov subspace methods can be used to solve the Galerkin 3.4. When the PDE is non-linear, the Galerkin system of equations 3.4 provides the residual evaluator for a non-linear solver and Jacobian can be represented in a similar fashion as 3.4, based upon the weak form

$$\langle v, J(u)\, w \rangle = \int_\Omega \begin{bmatrix} v^T & (\nabla v)^T \end{bmatrix} \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} w & \nabla w \end{bmatrix} \tag{3.5}$$

where $f_{i,0} = \frac{\partial f_i}{\partial u}(u, \nabla u)$ and $f_{i,1} = \frac{\partial f_i}{\partial \nabla u}(u, \nabla u)$. If these pointwise functions $f_{i,j}$ are not available analytically, they can be computed via algorithmic differentiation or finite differencing.

With these pointwise functions, we can describe Jacobian-free Newton-Krylov methods, which are used to solve non-linear PDEs. Jacobian-free Newton-Krylov methods were summarized, with preconditioning strategies, by Knoll and Keyes in [21].

3.3. Performance. To demonstrate the performance benefits of high-order finite elements implemented in a matrix-free fashion, we consider the specific case of the screened Poisson equation, $\nabla^2 u - \alpha^2 u = f$. In this case, application of the finite element operator for a single element requires $\mathcal{O}\left(P^6\right)$ matrix entries and $\mathcal{O}\left(P^6\right)$ floating point operations. In contrast, application of the matrix-free operator for a single element requires $\mathcal{O}\left(P^3\right)$ floating point values and $\mathcal{O}\left(P^{d+1}\right)$ floating point operations.

As seen in Figure 2, balance between bandwidth and FLOPs for matrix-free implementations more closely agrees with current HPC hardware capabilities.

In comparison to generation of simplex meshes, generation of high quality hexahedral meshes is a time intensive process. However, it is possible to generate meshes comprised predominately of high quality hexahedral elements with initial refinement of a simplex mesh without the costly process of fully converting a simplex mesh into hexahedral elements. Thus, the performance benefits of high-order matrix-free finite elements can be realized without substantial additional effort in generating a mesh exclusively composed of high quality hexahedral elements.

4. Hyperelasticity. Although our techniques are applicable to a wide range of problems, we are primarily interested in exploring Neo-Hookean hyperelasticity at finite strain in the incompressible regime.

In the total Lagrangian approach for the Neo-Hookean hyperelasticity problem, the discrete equations are formulated with respect to the reference configuration; we solve for displacement $u(X)$ in the reference frame X. Our notation is inspired by [18].

The strong form of the static balance of linear-momentum at finite strain is given by

$$-\nabla_X \cdot \boldsymbol{P} - \rho_0 \boldsymbol{g} = 0 \tag{4.1}$$

where ∇_X indicates that the gradient is calculated with respect to the reference configuration in the finite strain regime. P and $\boldsymbol{g}$ are the first Piola-Kirchhoff stress tensor and the prescribed forcing function, respectively and ρ_0 is the reference mass density. The first Piola-Kirchhoff stress tensor is given by

$$\boldsymbol{P} = \boldsymbol{F}\,\boldsymbol{S}, \tag{4.2}$$

where S is the second Piola-Kirchhoff stress tensor and $\boldsymbol{F} = \boldsymbol{I} + \nabla_X u$ is the deformation gradient. $\boldsymbol{S}$ is defined by the constitutive model.

Integrating by parts, we have the weak form

$$\int_\Omega \nabla_X \boldsymbol{v} \colon \boldsymbol{P} - \int_\Omega \boldsymbol{v} \cdot \rho_0 \boldsymbol{g} - \int_{\partial\Omega} \boldsymbol{v} \cdot (\boldsymbol{P} \cdot \hat{\boldsymbol{N}}) = 0 \tag{4.3}$$

where $\boldsymbol{P} \cdot \hat{\boldsymbol{N}}|_{\partial\Omega}$ is replaced by any prescribed force/traction boundary conditions written in terms of the reference configuration.

The Newton linearization of the volumentric term is given by

$$\int_\Omega \nabla_X \boldsymbol{v} : d\boldsymbol{P} = \int_\Omega d\boldsymbol{F}\boldsymbol{S} + \boldsymbol{F} d\boldsymbol{S} \tag{4.4}$$

were $d\boldsymbol{S}$, as above, depends upon the constitutive model chosen to determine the second Piola-Kirchhoff stress tensor.

Omitting the full derivation for the sake of brevity, we are currently interested in the Neo-Hookean constitutive model.

$$\boldsymbol{S} = \lambda \log(|\boldsymbol{F}|)\,\boldsymbol{C}^{-1} + 2\mu \boldsymbol{C}^{-1}\boldsymbol{E} \tag{4.5}$$

where $\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F}$ is the right Cauchy-Green tensor and $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I})$ is the Green-Lagrange strain tensor. The Lamé parameters are given by $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$.

We therefore have in our linearization

$$d\boldsymbol{S} = \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{E}} : d\boldsymbol{E} = \lambda(\boldsymbol{C}^{-1} : d\boldsymbol{E})\boldsymbol{C}^{-1} + 2(\mu - \lambda \log(|\boldsymbol{F}|))\boldsymbol{C}^{-1} d\boldsymbol{E} \boldsymbol{C}^{-1} \tag{4.6}$$

5. Preconditioning. As discussed in Section 4, high-order matrix-free finite elements offer performance benefits in comparison to assembled sparse matrix representations. As is typical with sparse matries, matrix-free formulations require iterative solvers for solve the Galerkin system of equations. We are primarily interested in Conjugate Gradient (CG), first developed by Hestens and Stiefel [16], which is Krylov subspace method. Krylov subspace methods are a natural fit for matrix-free finite elements, as these methods only require matrix-vector products to populate the Krylov subspace

$$\mathcal{K}\left(r_0, \mathbf{A}, k\right) = \{r_0, \mathbf{A}r_0, \ldots, \mathbf{A}^{k-1}r_0\} \tag{5.1}$$

which is used to construct increasingly accurate iterates x^k that approach the true solution x.

The iteration count to reach convergence of Krylov subspace methods is based upon condition number of the operator [23] and high-order finite element operators have notoriously poor condition numbers [19]. In this section we discuss preconditioners based upon p-type multigrid to control the condition number of high-order finite elements implemented in a matrix-free fashion. With these preconditioners, we can reduce total iteration count and thus total time to solution for these operators.

Suppose we are solving the linear system given by

$$\mathbf{A}x = b \tag{5.2}$$

via a Krylov subspace method. This linear system may come from the Galerkin system of equations of our PDE of interest or the Jacobian of our PDE of interest. We can improve the convergence of our Krylov method for this system by solving the preconditioned system

$$\left(\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\right)\left(\mathbf{M}_R x\right) = \mathbf{M}_L^{-1} b \tag{5.3}$$

via our Krylov method instead. We will investigate left preconditioning, where $\mathbf{M}_R = I$ and $\mathbf{M}_L^{-1} \approx \mathbf{A}^{-1}$. Therefore, we will adopt the notation $\mathbf{M}_L = \mathbf{M}$.

Note that preconditioning CG is more restrictive than other Krylov methods as the CG algorithm requires preconditioners to preserve the symmetric positive definite property of the operator representing the Galerkin system. Thus, preconditioning methods for CG can be applied to more general Krylov methods, such as generalized minimal residual method (GMRES).

5.1. P-Multigrid. Multigrid methods are popular multi-level techniques that provide resolution independent convergence rates. p-type multigrid, developed by Ronquist and Patera [31], is a natural choice for high-order finite elements on an unstructured mesh and can be implemented with operators represented as in 3.4. Ronquist and Patera declared p-multigrid *sensibly independent* of number of elements and polynomial degree of the element bases. Multigrid can be used as an independent solver, but we investigate the use of multigrid as a preconditioner for a Krylov method.

There has been work by Heys, Manteuffel, McCormick, and Olson demonstrating the feasibility of algebraic multigrid (AMG) for high-order finite elements [17]; however, algebraic multigrid requires assembly of the finite element operator, which defeats the benefits of matrix-free implementation. There are examples of using h-multigrid for matrix-free finite elements in solid mechanics, such as [6]; however p-multigrid offers more flexibility with respect to meshes in comparison to h-multigrid as it does not require aggregation of multiple elements into larger elements.

While the theoretical foundation for p-multigrid implemented in a matrix-free fashion on unstructured meshes has existed for quite some time, there do not appear to be many examples of these techniques being combined for practical problems. We are collaborating on a paper implementing these techniques in the context of Neo-Hookean hyperelasticity at finite strain. This is a new contribution to the solid mechanics community, which typically uses low order finite elements and assembled sparse matrices. Overall, the use of high-order finite elements with p-multigrid is under-explored for solid mechanics problems, especially with high-contrast or nearly incompressible materials.

Application of p-multigrid follows the typical multigrid algorithm 5.1, such as described by [3] where $\mathbf{P}$ is the grid prolongation operator, $\mathbf{R}$ the grid restriction

Algorithm 5.1 Multigrid Algorithm

1: Compute x^k
2: $x^k \leftarrow x^k + \hat{\mathbf{M}}^{-1}\left(b - \mathbf{A}x^k\right)$ ▷ pre-smooth m times
3: $r = \mathbf{R}\left(b - \mathbf{A}x^k\right)$ ▷ restrict the residual
4: $\mathbf{A}_c e = r$ ▷ Solve on coarse grid (may involve additional levels)
5: $x^k \leftarrow x^k + \mathbf{P}e$ ▷ prolongate error
6: $x^k \leftarrow x^k + \hat{\mathbf{M}}^{-1}\left(b - \mathbf{A}x^k\right)$ ▷ post-smooth m times

operator, $\hat{\mathbf{M}}$ a separate preconditioner used for smoothing, and $\mathbf{A}_c$ represents solving the error correction problem on the coarse grid, which may involve recursively applying p-multigrid.

For p-multigrid with nodal bases, the prolongation operator interpolates using the lower order basis functions to the higher order basis nodes and is defined by

$$\mathbf{P}_{p-1}^{p} = \Lambda\left(m_p^{-1}\right)\sum_e \mathcal{E}_p^T \mathbf{B}_{p-1}^{p} \mathcal{E}_{p-1} \tag{5.4}$$

where $\mathbf{B}_{p-1}^{p}$ interpolates from a basis of degree $p-1$ to degree p and $m_p = \mathcal{E}_p^T \mathcal{E}_p 1$ counts the multiplicity of shared nodes between elements. To preserve symmetry and prevent aliasing, the restriction operator is defined as the transpose of the prolongation operator, $\mathbf{R}_{p-1}^{p} = \left(\mathbf{P}_{p-1}^{p}\right)^T$. These operators can be implemented matrix-free, in the same fashion as 3.4.

Even modest order finite elements, such as degree 4, p-multigrid can substantially reduce the size of the global solution vector by a factor of approximately $P^3/8$. Thus, assembly of the finite element operator on the coarse grid, $\mathbf{A}_c$, becomes tractable and direct solvers can be used to solve the coarse problem. We use AMG for a coarse grid solver, which allows us to leverage the benefits discussed by Heys, Manteuffel, McCormick, and Olson [17].

5.2. Smoothers. In multigrid preconditioning, the smoother is an important component that helps resolve the error correction on a given multigrid level. The smoother targets the high-energy components, which corresponds to the larger eigenvalues in the spectrum of the operator. Below we discuss the more traditional Jacobi and Chebyshev smoothers as well as a proposed domain decomposition based technique. These smoothers can also be used as preconditioners outside the context of multigrid.

5.2.1. Jacobi and Chebyshev. Efficient Jacobi and Chebyshev implementations are important as smoothers for multigrid preconditioners. As discussed in [1],

the Chebyshev semi-iterative method is a particularly good choice as a smoother for parallel multigrid implementations. It is easier to provide adequate estimates of the maximal eigenvalue and Chebyshev is not too sensitive to estimation of this parameter.

Currently, we use the Chekyshev semi-iterative method with our current Neo-Hookean hyperelasticity solver. While Jacobi and Chebyshev techniques are well established, there is some work to be done in providing efficient operator diagonal or block diagonal assembly, especially in the context of matrix-free Jacobians computed via numerical or algorithmic differentiation.

5.2.2. Domain Decomposition. Domain decomposition methods are another popular class of preconditioners that can provide smoothers for multigrid methods. Fisher, among others, has used overlapping Schwarz for high-order finite elements or spectral elements with fluid dynamics problems, such as in [12] and [13]. However, overlapping Schwarz requires additional memory movement and computation due to the overlap.

Balancing Domain Decomposition by Constraints (BDDC), first developed by Dohrmann [8], is technique for non-overlapping domain decomposition. In BDDC, the coarse problem is solved on a reduced set of shared nodes between subdomains, using element corners and edges in 3D. BDDC is closely related to Finite Element Tearing and Interconnecting (FETI), developed by Farhat and Roux [11], and subclasses of these two methods can be shown to be the same method [15], [20], and [30].

For sufficiently high-order elements, we can treat each element as a subdomain. We partition the degrees of freedoms into two groups, those on the interface Γ and those in the interior I. We can therefore formulate the BDDC preconditioner, as seen in [5], as

$$\hat{\mathbf{M}}^{-1} = \left(\mathbf{R}_1^T - \mathcal{H}\mathbf{J}_D\right) \hat{\mathbf{A}}^{-1} \left(\mathbf{R}_1 \mathbf{J}_D^T \mathcal{H}\right) \tag{5.5}$$

where $\mathcal{H}$ is the direct sum of local operators $\mathcal{H}^{(i)} = -\left(\mathbf{A}_{II}^{(i)}\right)^{-1} \left(A_{\Gamma I}^{(i)}\right)^T$ that map the jump over subdomain interfaces J_D to subdomain interiors by solving a local Dirichlet problem and giving zero for other values so

$$\left(\mathbf{J}_D^T v\left(x\right)\right)^{(i)} = \sum_{j \in \mathcal{N}_x} \left(\delta_j\left(x\right) v^{(i)}\left(x\right) - \delta_i\left(x\right) v^{(j)}\left(x\right)\right) \forall x \in \Gamma_i \tag{5.6}$$

with $\delta_i\left(x\right) = 1/|\mathcal{N}_x|$ and $\mathcal{N}_x$ is the set of indices of subdomains that have x on their boundary. The function $\delta_i\left(x\right)$ is used to create the scaled injection operator $\mathbf{R}_1$ such that interior values have 1 and interface values have $|\mathcal{N}_x|$ entries each set to $\delta_i\left(x\right)$.

The operator $\widetilde{\mathbf{A}}^{-1}$ represents a subdomain solver. This subdomain solver is often a direct method; however, Li and Widlun demonstrated the feasibility of inexact solvers in [22].

5.2.3. Subdomain Solvers. We are investigating separable approximate inverses based on the Fast Diagonalization Method (FDM) for non-separable problems as inexact subdomain solvers.

Lynch, Rice, and Thomas introduced FDM in [24]. FDM directly solves separable linear equations based upon tensor products of lower dimension operators. The screened Poisson operator, $\left(\alpha^2 - \nabla^2\right) u$, has the Galerkin system of equations

$$\sum_e \mathcal{E}^T \left[\left(\mathbf{B}^e\right)^T \mathbf{W}^e \Lambda\left(\left(\alpha^2\right) \mathbf{B}^e \mathcal{E}^e u\right) + \sum_{i=0}^{d-1} \left(\mathbf{D}_i^e\right)^T \mathbf{W}^e \Lambda\left(\left\{\mathbf{D}_i^e \mathcal{E}^e u\right\}_{i=0}^{d-1}\right) \right] \tag{5.7}$$

For a single element, the Galerkin system 5.7 for the 3D screened Poisson problem can be rewritten as

$$A = \alpha^2 \mathbf{M} + \sum_{i=0}^{d-1} \mathbf{K}_i \tag{5.8}$$

where $\mathbf{M} = \mathbf{B}^T \mathbf{W} \mathbf{B}$, $\mathbf{K}_i = \mathbf{D}_i^T \mathbf{W} \mathbf{D}_i$, and so on. In this example, we neglect the terms arising from the coordinate mapping as they can result in a non-separable operator.

These operators $\mathbf{M}$ and $\mathbf{K}$ can be written in terms of the 1D operators, $M = B^T W B$ and $K = D^T W D$, as $\mathbf{M} = M \otimes M \otimes M$, $\mathbf{D}_0 = D \otimes M \otimes M$, and so on. As K is symmetric and M is symmetric positive definite, we can simultaneously diagonalize K and M, yielding $\mathcal{X}^T M \mathcal{X} = I$ and $\mathcal{X}^T K \mathcal{X} = L$. With these pseudoeigenvalues and pseudoeigenvectors, we can rewrite A as

$$A = \boldsymbol{\mathcal{X}} \left(\alpha^2 \mathbf{I} + \sum_{i=0}^{d-1} \mathbf{L}_i \right) \boldsymbol{\mathcal{X}}^T \tag{5.9}$$

with inverse

$$A^{-1} = \boldsymbol{\mathcal{X}}^T \left(\alpha^2 \mathbf{I} + \sum_{i=0}^{d-1} \mathbf{L}_i \right)^{-1} \boldsymbol{\mathcal{X}} \tag{5.10}$$

where $\boldsymbol{\mathcal{X}} = \mathcal{X} \otimes \mathcal{X} \otimes \mathcal{X}$, $\mathbf{L}_0 = L \otimes I \otimes I$, and so on. Notice that if we treat $\boldsymbol{\mathcal{X}}$ as a basis operation, these inverses can be described and implemented matrix-free, as in 3.4.

With non-separable operators A, a suitable choice of $\left(\alpha^2 \mathbf{I} + \sum_{i=0}^{d-1} \mathbf{L}_i \right)$ can produce suitable approximate subdomain solvers. Fisher, Miller, and Tufo demonstrated in [14] that while FDM cannot be used for arbitrarily deformed subdomains, defining the diagonalization over a parallelepiped with average dimensions in each coordinate direction is adequate for preconditioning. For PDEs with arbitrarily deformed subdomains and non-linear pointwise functions f_0 and f_1, early experiments indicate that a suitable approximate inverse might be constructed by correctly selecting pseudoeigenvalues to use with pseudoeigenvectors from the mass and stiffness matrices given above in 5.8. These approximate inverse techniques may be used with other domain decomposition techniques; however, overlapping techniques may require significant additional computation at high-order due to the additional nodes in each subdomain.

5.3. Split Preconditioners. The preconditioners given above are not appropriate for mixed finite element methods. In these cases, splitting these PDE by fields and applying different preconditioners to different fields can yield an effective preconditioner. In this way, the above preconditioners can be composed to handle a wider range of problems.

If the operator A has the decomposition

$$A = \begin{pmatrix} F & B \\ B^T & C \end{pmatrix} \tag{5.11}$$

then it can be decomposed via LDU as

$$A = \begin{pmatrix} I & 0 \\ B^T F^{-1} & I \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & -S \end{pmatrix} \begin{pmatrix} I & F^{-1} B \\ 0 & I \end{pmatrix} \tag{5.12}$$

where $S = C + B^T F^{-1} B$ is the Schur complement of F in A. The development of block multi-level preconditioners has been extensively explored [10].

We are collaborating in the development of a solver for Neo-Hookean hyperelasticity at finite strain in the incompressible regime. This solver will split the displacement and discontinuous pressure fields, applying p-multigrid to the displacement fields and block Jacobi to the discontinuous pressure field. This approach will facilitate solving incompressible hyperelastic problems at large scale.

6. Conclusion. We discussed the benefits of high-order matrix-free finite elements for solid mechanics problems. We highlighted several areas for future improvement in preconditioning for high-order finite element operators implemented in a matrix-free fashion. p-multigrid with matrix-free prolongation, restriction, and smoothing operators is a natural fit for high-order matrix-free finite elements and has proven effective in solid mechanics problems. While Jacobi and Chebyshev semi-iterative preconditioning is not new, these techniques are important as smoothers for multigrid methods, and there is work to be done on providing efficient operator diagonal or point block diagonal assembly, especially for Jacobians computed via numerical or alogrithmic differentiation. BDDC is another attractive technique for smoothing multigrid methods, and FDM based matrix-free separable approximate inverses may provide suitable subdomain solvers for BDDC and other domain decomposition techniques with non-linear and non-separable problems. Neo-Hookean hyperelasticity at finite strain in the incompressible regime will require mixed finite elements with split preconditioning.

Acknowledgments. 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] M. Adams, M. Brezina, J. Hu, and R. Tuminaro, *Parallel multigrid smoothing: polynomial versus Gauss-Seidel*, Journal of Computational Physics, 188 (2003), pp. 593–610.
[2] M. Adams, J. Brown, J. Shalf, B. Straalen, E. Strohmaier, and S. Williams, *HPGMG 1.0: A benchmark for ranking high performance computing systems*, LBNL Technical Report, (2014).
[3] A. Brandt, *Guide to multigrid development*, in Multigrid methods, Springer, 1982, pp. 220–312.
[4] J. Brown, *Efficient nonlinear solvers for nodal high-order finite elements in 3D*, Journal of Scientific Computing, 45 (2010), pp. 48–63.
[5] J. Brown, Y. He, and S. MacLachlan, *Local fourier analysis of balancing domain decomposition by constraints algorithms*, SIAM Journal on Scientific Computing, 41 (2019), pp. S346–S369.
[6] D. Davydov, J.-P. Pelteret, D. Arndt, and P. Steinmann, *A matrix-free approach for finite-strain hyperelastic problems using geometric multigrid*, arXiv preprint arXiv:1904.13131, (2019).
[7] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy, *Toward a universal hp adaptive finite element strategy, part 1. constrained approximation and data structure*, Computer Methods in Applied Mechanics and Engineering, 77 (1989), pp. 79–112.
[8] C. R. Dohrmann, *A preconditioner for substructuring based on constrained energy minimization*, SIAM Journal on Scientific Computing, 25 (2003), pp. 246–258.
[9] J. Dongarra, M. A. Heroux, and P. Luszczek, *High-performance conjugate-gradient benchmark*, International Journal of High Performance Computing Applications, 30 (2016),

pp. 3–10.
[10] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations*, Journal of Computational Physics, 227 (2008), pp. 1790–1808.
[11] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, International Journal for Numerical Methods in Engineering, 32 (1991), pp. 1205–1227.
[12] P. F. Fischer, *An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations*, Journal of Computational Physics, 133 (1997), pp. 84–101.
[13] P. F. Fischer and J. W. Lottes, *Hybrid Schwarz-multigrid methods for the spectral element method: Extensions to Navier-Stokes*, in Domain Decomposition Methods in Science and Engineering, Springer, 2005, pp. 35–49.
[14] P. F. Fischer, H. M. Tufo, and N. Miller, *An overlapping Schwarz method for spectral element simulation of three-dimensional incompressible flows*, in Parallel Solution of Partial Differential Equations, Springer, 2000, pp. 159–180.
[15] Y. Fragakis and M. Papadrakakis, *The mosaic of high performance domain decomposition methods for structural mechanics: Formulation, interrelation and numerical efficiency of primal and dual methods*, Computer methods in applied mechanics and engineering, 192 (2003), pp. 3799–3830.
[16] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, 49 (1952).
[17] J. Heys, T. Manteuffel, S. F. McCormick, and L. Olson, *Algebraic multigrid for higher-order finite elements*, Journal of computational Physics, 204 (2005), pp. 520–532.
[18] G. Holzapfel, *Nonlinear solid mechanics: a continuum approach for engineering*, Wiley, Chichester New York, 2000.
[19] N. Hu, X.-Z. Guo, and I. Katz, *Bounds for eigenvalues and condition numbers in the p-version of the finite element method*, Mathematics of computation, 67 (1998), pp. 1423–1450.
[20] A. Klawonn and O. Widlund, *FETI and Neumann-Neumann iterative substructuring methods: connections and new results*, Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences, 54 (2001), pp. 57–90.
[21] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton–Krylov methods: a survey of approaches and applications*, Journal of Computational Physics, 193 (2004), pp. 357–397.
[22] J. Li and O. B. Widlund, *On the use of inexact subdomain solvers for BDDC algorithms*, Computer Methods in Applied Mechanics and Engineering, 196 (2007), pp. 1415–1428.
[23] D. G. Luenberger, *Introduction to linear and nonlinear programming*, vol. 28, Addison-Wesley Reading, MA, 1973.
[24] R. E. Lynch, J. R. Rice, and D. H. Thomas, *Direct solution of partial difference equations by tensor product methods*, Numerische Mathematik, 6 (1964), pp. 185–199.
[25] J. D. McCalpin, *Memory bandwidth and system balance in hpc systems*, Invited talk, Supercomputing, (2016).
[26] H. Meuer, E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer, *Top 500 list*, 2020, http://www.top500.org/.
[27] J. T. Oden, L. Demkowicz, W. Rachowicz, and T. Westermann, *Toward a universal hp adaptive finite element strategy, part 2. a posteriori error estimation*, Computer methods in applied mechanics and engineering, 77 (1989), pp. 113–180.
[28] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, *HPL-a portable implementation of the high-performance linpack benchmark for distributed-memory computers*, (2004), http://www.netlib.org/benchmark/hpl/.
[29] W. Rachowicz, J. T. Oden, and L. Demkowicz, *Toward a universal hp adaptive finite element strategy part 3. design of hp meshes*, Computer Methods in Applied Mechanics and Engineering, 77 (1989), pp. 181–212.
[30] D. J. Rixen, C. Farhat, R. Tezaur, and J. Mandel, *Theoretical comparison of the FETI and algebraically partitioned FETI methods, and performance comparisons with a direct sparse solver*, International Journal for Numerical Methods in Engineering, 46 (1999), pp. 501–533.
[31] E. M. Rønquist and A. T. Patera, *Spectral element multigrid. I. formulation and numerical results*, Journal of Scientific Computing, 2 (1987), pp. 389–406.
[32] K. Rupp, *CPU-GPU-MIC comparision charts*, 2020, https://github.com/karlrupp/cpu-gpu-mic-comparison.