

1 **PRECONDITIONING MATRIX-FREE HIGH-ORDER FINITE**
2 **ELEMENT OPERATORS**

3 JEREMY L. THOMPSON*

4 **Abstract.** Global sparse matrices are no longer a good representation of a high-order finite
5 element operator. Matrix-free operators offer superior performance, both with respect to FLOPs
6 needed for evaluation and the memory transfer needed for a matrix-vector product. However, matrix-
7 free methods require iterative solvers, which are sensitive to the condition number of the operator.
8 Matrix-free operators with tensor product bases have the best efficiency at high-order, but the ill-
9 conditioning slows iterative solver convergence. Preconditioners can accelerate convergence of these
10 solvers with high-order operators. We discuss preconditioners that can be efficiently derived from
11 matrix-free operators or formulated as matrix-free operators themselves, to include diagonal or block
12 diagonal based techniques such as Jacobi and Chebyshev, multi-level techniques such a p-multigrid,
13 and domain decomposition techniques such as Additive Schwartz and BDDC.

14 **1. Introduction.** High Performance Computing (HPC) hardware has seen im-
15 provements in Floating Point Operations per Second (FLOPs) that outstrip improve-
16 ments in memory and network bandwidth, as highlighted in McCalpin's invited talk an
17 Supercomputing 2016 [24]. Under these hardware constraints, global sparse matrices
18 are no longer a good representation of a high-order finite element operator. Matrix-
19 free operators offer superior performance, both with respect to FLOPs needed for
20 evaluation and the memory transfer needed for a matrix-vector product. However,
21 matrix-free methods require iterative solvers, which are sensitive to the condition
22 number of the operator.

23 In Section 2, we discuss the specific hardware limitations that constrain the per-
24 formance of finite element codes in HPC. In Section 3, we provide notation to describe
25 arbitrary PDEs for matrix-free implementation and discuss the performance matrix-
26 free implementation compared to assembled matrices for high-order finite elements on
27 3D hexahedral meshes. In Section 4, we discuss preconditioning techniques for high-
28 order finite elements and highlight areas where future work can bring these techniques
29 to new applications or improve the performance of these preconditioners.

30 **2. Hardware Limitations.** Two key performance metrics for high performance
31 computing hardware are FLOPs and memory and network bandwidth. FLOPs is the
32 more well known of these two metrics; the Top 500 [25] list tracks the 500 super-
33 computers with the highest peak FLOPs, as measured by High-Performance Linpack
34 (HPL) [26]. HPL measures the performance when solving random dense linear sys-
35 tems in double precision via LU factorization.

36 Other benchmarks, such as High-Performance Geometric Multigrid (HPGMG) [1]
37 and High-Performance Conjugate Gradient (HPCG) [6], measure performance based
38 upon solving a more complex benchmark problem. These The disparity between the
39 FLOPs achieved in benchmarks such as HPGMG and HPCG and the peak FLOPs
40 measured by HPL is partially explained by the difference between FLOPs and memory
41 and network bandwidth.

42 Over the last thirty years, the peak FLOPs for new HPC hardware has been
43 increasing more rapidly than memory bandwidth and network bandwidth, for both
44 CPUs and GPUs. As discussed in McCalpin's Supercomputing 2016 invited talk
45 [24], peak FLOPs per socket have been increasing at a rate of 50-60% per year while

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

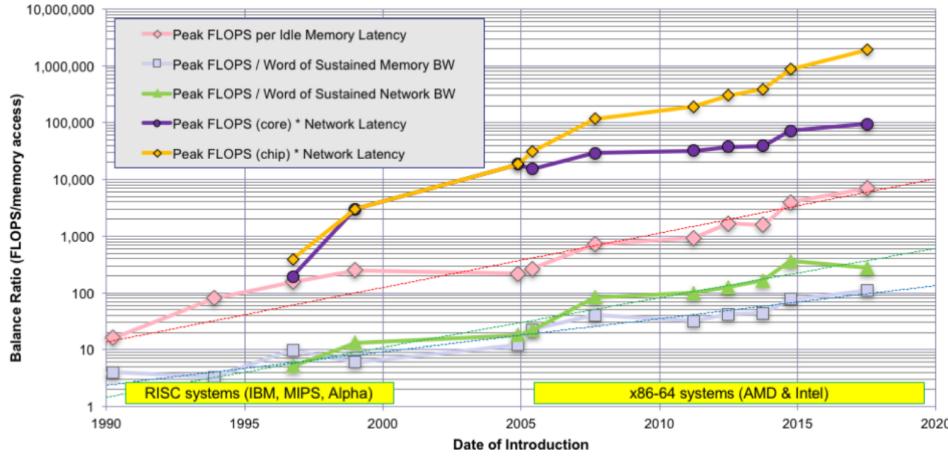


FIG. 1. 30 year system balance history

46 memory bandwidth has only been increasing at a rate of approximately 23% per year
 47 and network bandwidth has only been increasing at a rate of approximately 20% per
 48 year.

49 As seen in Figure 1, from 2013 to 2016, FLOPs have improved twice as much as
 50 memory and network bandwidth. This problem is exacerbated by network latency,
 51 which is decreasing at a rate of approximately 20% per year, and memory latency,
 52 which is increasing at a rate of approximately 20% per year.

53 Matrix-free representations of high-order finite element operators provide a way
 54 to reduce total FLOPs required to execute matrix-vector products and significantly
 55 reduce the memory transfer required.

56 **3. Matrix-Free Finite Elements.** High-order finite elements implemented in
 57 a matrix-free fashion are one way to address the common performance bottlenecks for
 58 modern HPC hardware. In this section we develop notation to describe high-order
 59 finite elements on unstructured meshes implemented in a matrix-free fashion. Fur-
 60 thermore, we compare the performance of matrix-free implementations and assembled
 61 matrices.

62 **3.1. Notation.** The development of this notation will largely follow [2].

63 Let $\{X_i\}_{i=1}^P$ denote the Legendre-Gauss-Lobatto (LGL) nodes of degree $P - 1$
 64 on the reference interval $[-1, 1]$ while $\{q_i\}_{i=1}^Q$ and $\{w_i\}_{i=1}^Q$ denote the quadrature
 65 points and quadrature weights corresponding to a Q point quadrature rule. If we
 66 consider Lagrange basis functions $\{\phi_i\}_{i=1}^P$, we can construct matrices $B_{ij} = \phi_j(q_i)$,
 67 $D_{ij} = \partial_x \phi_j(q_i)$, and $W_{ij} = w_i \delta_{ij}$, representing interpolation to the quadrature points,
 68 computation of derivatives at the quadrature points, and quadrature weights.

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

$$\mathbf{B} = B \otimes B \otimes B$$

$$\mathbf{D}_0 = D \otimes B \otimes B$$

70 (3.1) $\mathbf{D}_1 = B \otimes D \otimes B$

$$\mathbf{D}_2 = B \otimes B \otimes D$$

$$\mathbf{W} = W \otimes W \otimes W.$$

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

78 (3.2)
$$\mathbf{D}_i^e = \Lambda \left(\frac{\partial X_0}{\partial x_i} \right) \mathbf{D}_0 + \Lambda \left(\frac{\partial X_1}{\partial x_i} \right) \mathbf{D}_1 + \Lambda \left(\frac{\partial X_2}{\partial x_i} \right) \mathbf{D}_2$$

79 where $\Lambda(X)_{ij} = X_i \delta_{ij}$ expresses pointwise multiplication of J_{ij}^{-1} at quadrature points
 80 as a diagonal matrix. Element integration weights therefore become $\mathbf{W}^e = W \Lambda(|J^e(q)|)$. ■

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

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

90 Given the weak form of an arbitrary PDE

91 (3.3)
$$\begin{aligned} & \text{find } u \in V \text{ such that for all } v \in V \\ & \langle v, u \rangle = \int_{\Omega} v \cdot f_0(u, \nabla u) + \nabla v : f_1(u, \nabla u) = 0 \end{aligned}$$

92 the corresponding Galerkin system of equations is

93 (3.4)
$$\sum_e \mathcal{E} \left[(\mathbf{B})^T \mathbf{W}^e \Lambda(f_0(u^e, \nabla u^e)) + \sum_{i=0}^{d-1} (\mathbf{D}_i^e)^T \Lambda(f_1(u^e, \nabla u^e)) \right] = 0$$

94 where $u^e = \mathbf{B} \mathcal{E}^e u$ and $\nabla u^e = \{\mathbf{D}_i^e \mathcal{E}^e u\}_{i=0}^{d-1}$.

95 Dirichlet boundary conditions are represented in the element restriction opera-
 96 tion by enforcing the specified values on the constrained nodes. Neumann or Robin
 97 boundary conditions are represented by adding boundary integral terms with basis
 98 and element restriction operations. Boundary integrals internal to the domain Ω , such
 99 as face integrals in Discontinuous Galerkin methods, can also be represented using
 100 additional terms with appropriate bases and element restrictions.

101 **3.2. Linearization.** When the PDE 3.3 is linear, the pointwise functions $f_0()$
 102 and $f_1()$ are also linear and Krylov subspace methods can be used to solve the
 103 Galerkin system of equations 3.4. When the PDE is non-linear, the Jacobian of
 104 the residual evaluator given in 3.4 can be represented in a similar fashion, based upon
 105 the weak form

106 (3.5) $\langle v, J(u) w \rangle = \int_{\Omega} [v^T \nabla v^T] \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} w \nabla w \end{bmatrix}$

107 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 are not
 108 available analytically, they can be computed via algorithmic differentiation or finite
 109 differencing. With these pointwise functions, Jacobian-free Newton-Krylov methods
 110 can be used to solve non-linear PDEs. Jacobian-free Newton-Krylov methods were
 111 summarized, with preconditioning strategies, by Knoll and Keyes in [17].

112 **3.3. Performance.** To demonstrate the performance benefits of high-order fi-
 113 nite elements implemented in a matrix free format, we explore the specific case of the
 114 Helmholtz equations. The strong form of the Helmholtz equations is given by

115 (3.6) $\nabla^2 f = -k^2 f$

116 The corresponding weak form is given by

117 (3.7) $\int_{\Omega} \nabla v : \nabla u - k^2 v \cdot u = 0$

118 with Galerkin system of equations

119 (3.8) $\sum_e \mathcal{E} \left[(\mathbf{B})^T \mathbf{W}^e \Lambda ((-k^2) \mathbf{B} \mathcal{E}^e u) + \sum_{i=0}^{d-1} (\mathbf{D}_i^e)^T \Lambda (\{\mathbf{D}_i^e \mathcal{E}^e u\}_{i=0}^{d-1}) \right] = 0$

120 The total operations and matrix entries required to apply the matrix-vector product
 121 for the operator representing the Galerkin system of equations 3.8 depends upon a
 122 specific partitioning of the domain Ω into elements. For simplicity, we will compare
 123 the total floating point operations and matrix entries required to apply the operator
 124 representing 3.8 for a single high-order element with arbitrary hexahedral geometry.
 125 While fewer operations and matrix entries will be required on a domain decomposed
 126 into multiple elements due to the shared nodes on element boundaries, this comparison
 127 will adequately illustrate the relative merits of matrix-free implementations.

128 With a polynomial basis of degree $P - 1$, a 3D hexahedral element has P^3 nodes.
 129 Therefore, the assembled matrix representing the Galerkin system of equations 3.8
 130 has P^6 entries. Additionally, applying a matrix-vector product requires $\mathcal{O}(2P^6 - P^3)$
 131 floating point operations.

132 On the other hand, a matrix-free representation of the Galerkin system of equa-
 133 tions 3.8 can exploit tensor contractions to reduce total required operator entries
 134 and floating point operations. With a P point quadrature rule, the 1D basis oper-
 135 ators, \mathbf{B} and \mathbf{D} , have $2P^2$ entries, the inverses of the Jacobians of the coordinate
 136 mappings require $d^2 P^3$ entries, and the element quadrature weights require P^3 en-
 137 tries. Including the spatial frequency, $(d^2 + 1) P^3 + 2P^2 + 1$ entries are required.

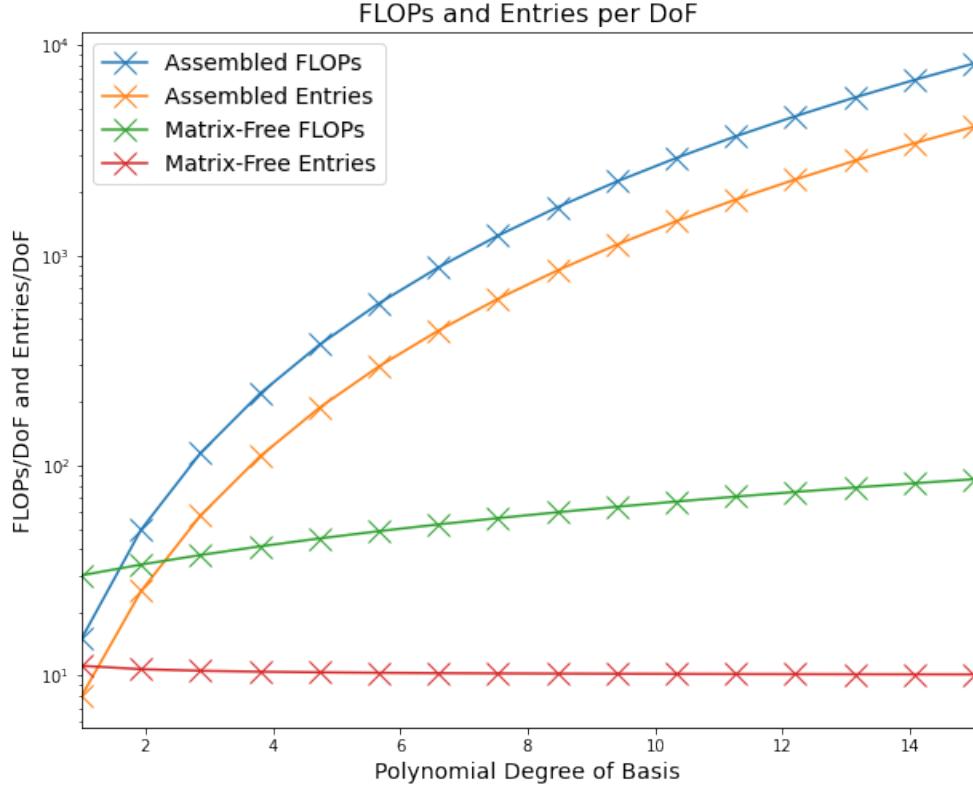


FIG. 2. Performance per DoF

138 Using tensor contractions to interpolate the nodal values to the quadrature space
 139 requires $\mathcal{O}(P^{d+1})$ operations, and computing derivatives at the quadrature points
 140 requires an additional $\mathcal{O}(P^{d+1} + (2d^2 - 1)P^3)$ operations. With appropriate pre-
 141 computation of the geometric factors from the coordinate mapping, the pointwise
 142 application of the weak form and transpose basis operators can be applied in an
 143 additional $\mathcal{O}(2P^{d+1} + (d + 1)P^3)$ operations. In total, the matrix-vector requires
 144 $\mathcal{O}(4P^{d+1} + (2d^2 + d + 1)P^3)$ floating point operations.

145 For simplicity, we considered an arbitrary P point quadrature rule. The analysis
 146 is similar for over-integration or under-integration. If the LGL points are also used for
 147 then quadrature, then $2P^3$ floating point operations are required to apply the basis
 148 operators, as $\mathbf{B} = I$.

149 As shown in 2, the number of values required per node to represent the Galerkin
 150 system of equations 3.8 in a matrix-free fashion is constant with respect to polynomial
 151 order, while the number of values required for the assembled matrix representing the
 152 operator grows exponentially. The number of operations required to compute the
 153 matrix-vector product in a matrix-free fashion per node grows only linearly while the
 154 number of operations grows cubically for the assembled matrix.

155 In comparison to generation of simplex meshes, generation of high quality hexa-
 156 hedral meshes is a time intensive process. However, as the formulation 3.4 can handle
 157 meshes comprised of different finite element geometries, it is possible to generate

158 meshes comprised predominately of high quality hexahedral elements with initial re-
 159 refinement of a simplex mesh without the costly process of fully converting a simplex
 160 mesh into hexahedral elements. Thus, the performance benefits of high-order matrix-
 161 free finite elements can be realized without substantial additional effort in generating
 162 a mesh exclusively composed of high quality hexahedral elements.

163 **4. Preconditioning.** As discussed in section 3, high-order matrix-free finite
 164 elements offer performance benefits in comparison to assembled sparse matrix rep-
 165 resentations. When using matrix-free formulations, iterative solvers are required to
 166 solve the Galerkin system of equations. We are primarily interested in Krylov sub-
 167 space methods such as Conjugate Gradients, first developed by Hestens and Stiefel
 168 [13], and related methods by Lanczos [18] and [19]. Krylov subspace methods are a
 169 natural fit for matrix-free finite elements, as these methods only require matrix-vector
 170 products to populate the Krylov subspace

$$171 \quad (4.1) \quad \mathcal{K}(r_0, A, k) = \{r_0, Ar_0, \dots, A^{k-1}r_0\}$$

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

174 The iteration count to reach convergence of Krylov subspace methods is based
 175 upon condition number of the operator [21] and high order finite element operators
 176 have notoriously poor condition numbers [15]. In this section we discuss precondi-
 177 tioners to control the condition number of high order finite elements implemented in
 178 a matrix-free fashion. With these preconditioners, we can reduce total iteration count
 179 and thus total time to solution for these operators.

180 Suppose we are solving the linear system given by

$$181 \quad (4.2) \quad Ax = b$$

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

$$186 \quad (4.3) \quad (M_L^{-1}AM_R^{-1})(M_Rx) = M_L^{-1}b$$

187 via a Krylov method instead.

188 We will investigate left preconditioning, where $M_R = I$ and $M_L^{-1} \approx A^{-1}$. There-
 189 fore, we will adopt the notation $M_L = M$.

190 **4.1. Jacobi and Chebyshev.** Jacobi iterations for an assembled linear opera-
 191 tors produce a new approximate solution x^k via

$$192 \quad (4.4) \quad x_i^k = \left(b_i - \sum_{j \neq i} a_{ij}x_j^{k-1} \right) / a_{ii}$$

193 Computation of the full assembled linear operator for preconditioning defeats the
 194 benefits of matrix-free methods, but the true operator diagonal can be efficiently
 195 computed.

196 For Jacobi preconditioning based upon the true diagonal of the operator A , we
 197 have $M = \text{diag}(A)$ and

198 (4.5)
$$x_i^k = b_i/a_{ii}$$

199 The more diagonally dominant the linear operator A is, the better M_L^{-1} approximates
 200 the true inverse A^{-1} . Point block diagonal assembly offers a middle ground between
 201 diagonal assembly and operator assembly for operators that are insufficiently diag-
 202 onally dominant for Jacobi preconditioning based upon the true diagonal to be an
 203 effective preconditioner.

204 The Chebyshev semi-iterative method, analyzed extensively by Golub and Varga
 205 [12], can be viewed as an improvement of the Jacobi, or related Gauss-Seidel, tech-
 206 niques. In the Chebyshev method, we generate iterates of the form

207 (4.6)
$$y^k = \omega_k (y^{k-1} - y^{k-2} + \gamma r^{k-1}) + y^{k-2}$$

208 where $\omega_k = 2 \frac{2-\beta-\alpha}{\beta-\alpha} \frac{c_{k-1}(\mu)}{c_k(\mu)}$, $\gamma = 2(2-\alpha-\beta)$, $\hat{M}r^{k-1} = b - Ay^{k-1}$, and c_i are the
 209 Chebyshev polynomials with $\mu = 1 + \frac{1-\beta}{\beta-\alpha}$. The eigenvalues of A are in the range $[\alpha, \beta]$.
 210 \hat{M} , $y^0 = x^0$ and $y^1 = x^1$ come from another iterative preconditioning technique such
 211 as Jacobi.

212 While Jacobi and Chebyshev techniques are well established, there is some work
 213 to be done in providing efficient operator diagonal or block diagonal assembly, espe-
 214 cially in the context of matrix-free Jacobians derived numerically or algorithmically.
 215 Efficient Jacobi and Chebyshev smoothers are important as smoothers for multigrid
 216 preconditioners.

217 **4.2. P-Multigrid.** Multigrid methods are popular multi-level techniques that
 218 provide resolution independent convergence rates. p -type multigrid, developed by
 219 Ronquist and Patera [28], is a natural choice for high-order finite elements on an un-
 220 structured mesh and can be used with operators represented as in 3.4. Multigrid can be
 221 used as an independent solver, but we investigate the use of multigrid as a precondi-
 222 tioner for an Krylov method. There has been work by Heys, Manteuffel, McCormick,
 223 and Olson on algebraic multigrid for high order finite elements [14]; however algebraic
 224 multigrid requires assembly of the finite element operator, which defeats the benefits
 225 of matrix-free implementation. There are examples of using h -multigrid, such as [4];
 226 however p -multigrid offers more flexibility with respect to meshes in comparison to
 227 h -multigrid.

228 For p -type multigrid with nodal bases, the prolongation operator interpolates to
 229 a higher order basis nodes and is defined by

230 (4.7)
$$P_{p-1}^p = m_p^{-1} \sum_e \mathcal{E}_p^T \mathbf{B}_{p-1}^p \mathcal{E}_{p-1}$$

231 where \mathbf{B}_{p-1}^p interpolates from a basis of degree $p-1$ to degree p and $m_p = \mathcal{E}_p^T \mathcal{E}_p 1$
 232 counts the multiplicity of shared nodes between elements. The restriction operator
 233 is defined as the transpose of the prolongation operator, $R_{p-1}^p = (P_{p-1}^p)^T$. These
 234 operators can be implemented matrix-free, in the same fashion as 3.4.

235 As described by May, Sanan, Rupp, Knepley, and Smith in [23], with these com-
 236 ponents we can implement p -type multigrid

Algorithm 4.1 p -type multigrid

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

237 With even modest order finite elements, such as $P = 4$, p -type multigrid can
 238 substantially reduce the size of the global solution vector, which makes assembly of
 239 the finite element operator on the coarse grid tractable so that direct solvers such as
 240 Algebraic Multigrid can be used to solve the coarse problem.

241 While the theoretical foundation for p -type multigrid implemented in a matrix-
 242 free fashion on unstructured meshes has existed for quite some time, there do not
 243 appear to be many examples of these techniques being combined to practical problems.
 244 We are collaborating on a paper implementing these techniques in the context of
 245 Neo-Hookean hyperelasticity at finite strain. This is a new contribution to the solid
 246 mechanics community, which typically uses low order finite elements and assembled
 247 sparse matrices.

248 **4.3. Domain Decomposition.** Domain decomposition methods are another
 249 popular class of preconditioners for high order finite elements. Fisher, with others,
 250 has used overlapping Schwarz for high order finite elements or spectral elements with
 251 fluid dynamics problems, such as in [8] and [9]. As with h -type multigrid, overlapping
 252 Schwarz techniques require additional information about the mesh in order to provide
 253 overlapping subdomains based upon element boundaries.

254 Balancing Domain Decomposition by Constraints (BDDC), first developed by
 255 Dohrmann [5], is technique for non-overlapping domain decomposition. BDDC is
 256 closely related to Finite Element Tearing and Interconnecting (FETI), developed by
 257 Farhat and Roux [7], and subclasses of these two methods can be shown to be the
 258 same method [11], [16], and [27].

259 For sufficiently high order elements, we can treat each element as a subdomain.
 260 We can partition the degrees of freedoms into two groups, those on the interface Γ
 261 and those in the interior I . We can therefore formulate the BDDC preconditioner, as
 262 seen in [3], as

263 (4.8)
$$M^{-1} = (R_1^T - \mathcal{H}J_D) \hat{A}^{-1} (R_1 J_D^T \mathcal{H})$$

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

267 (4.9)
$$(J_D^T v(x))^{(i)} = \sum_{j \in \mathcal{N}_x} \left(\delta_j(x) v^{(i)}(x) - \delta_i(x) v^{(j)}(x) \right) \forall x \in \Gamma_i$$

268 with $\delta_i(x) = 1/|\mathcal{N}_x|$ and \mathcal{N}_x is the set of indices of subdomains that have x on their
 269 boundary. The function $\delta_i(x)$ is used to create the scaled injection operator R_1 such
 270 that interior values have 1 and interface values have $|\mathcal{N}_x|$ entries each set to $\delta_i(x)$.

271 The operator \tilde{A}^{-1} represents a subdomain solver. This subdomain solver can
 272 be a direct method, or an inexact solver as discussed by Li and Widlun in [20].
 273 We are interested in investigating separable approximate inverses based on the Fast
 274 Diagonalization Method (FDM) that approximate inverses to non-separable problems.

275 Lynch, Rice, and Thomas introduced FDM in [22]. FDM directly solves separable
 276 linear equations based upon tensor products of lower dimension operators. For a single
 277 element, the Galerkin system of equations 3.8 for the 3D Helmholtz problem 3.6 can
 278 be rewritten as

$$279 \quad (4.10) \quad A = \sum_{i=0}^{d-1} \mathbf{K}_i - k^2 \mathbf{M}$$

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

282 These operators \mathbf{M} and \mathbf{K} can be written in terms of the 1D operators, $M =$
 283 $B^T W B$ and $K = D^T W D$, as $\mathbf{M} = M \otimes M \otimes M$ and $\mathbf{D}_0 = D \otimes M \otimes M$. Provided
 284 that K is symmetric and M is symmetric and positive definite, we can simultane-
 285 ously diagonalize K and M , yielding $\mathcal{X}^T M \mathcal{X} = I$ and $\mathcal{X}^T K \mathcal{X} = L$. With these
 286 pseudoeigenvalues and pseudoeigenvectors, we can rewrite A as

$$287 \quad (4.11) \quad A = \mathcal{X} \left(\sum_{i=0}^{d-1} \mathbf{L}_i - k^2 I_3 \right) \mathcal{X}^T$$

288 with inverse

$$289 \quad (4.12) \quad A^{-1} = \mathcal{X}^T \left(\sum_{i=0}^{d-1} \mathbf{L}_i - k^2 I_3 \right)^{-1} \mathcal{X}$$

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

292 With non-separable operators A , a suitable choice of $\left(\sum_{i=0}^{d-1} \mathbf{L}_i - k^2 I_3 \right)$ can pro-
 293 duce suitable approximate subdomain solvers. Fisher, Miller, and Tufo demonstrated
 294 in [10] that while FDM cannot be used for arbitrarily deformed subdomains, defining
 295 the diagonalization over a parallelepiped with average dimensions in each coordinate
 296 direction is adequate for preconditioning. For PDEs with arbitrarily deformed sub-
 297 domains and non-linear pointwise functions f_0 and f_1 , early experiments indicate
 298 that a suitable approximate inverse might be constructed by correctly selecting pe-
 299 sudeigenvalues to use with pseudoeigenvectors from the mass and stiffness matrices
 300 given above.

301 **4.4. Split Preconditioners.** These preconditioners might not be appropriate
 302 for more complex PDEs. In these cases, splitting these PDE by fields and applying
 303 different preconditioners to different fields can yield results. In this way, the above
 304 preconditioners can be composed to handle a wider range of problems. We are collab-
 305 orating in the development of solver for Neo-Hookean hyperelasticity at finite strain
 306 in the incompressible regime. This solver will split the displacement and discontinu-
 307 ous pressure fields, applying p -type multigrid as a preconditioner to the displacement
 308 fields and block Jacobi as a preconditioner to the discontinuous pressure field.

5. Conclusion. We discussed the performance benefits of high-order finite elements implemented in a matrix-free fashion. High order finite element operators solved with Krylov subspace methods require preconditioning to improve convergence and time to solution.

We highlighted several areas for future improvement in preconditioning for high-order finite element operators implemented in a matrix-free fashion. While Jacobi, block Jacobi, and Chebyshev semi-iterative preconditioning is not new, these techniques are important, on their own or as smoothers for other techniques, and there is work to be done on providing efficient operator diagonal or point block diagonal assembly. p -type multigrid with matrix-free prolongation and restriction operators is a natural fit for high-order matrix-free finite elements and has proven effective in solid mechanics problems. BDDC is another attractive technique that can be implemented in a matrix-free fashion, and FDM based matrix-free separable approximate inverses may provide suitable subdomain solvers for BDDC and other domain decomposition techniques.

REFERENCES

- [1] M. ADAMS, J. BROWN, J. SHALF, B. STRAALEN, E. STROHMAIER, AND S. WILLIAMS, *Hpmgm 1.0: A benchmark for ranking high performance computing systems*, LBNL Technical Report, (2014).

[2] J. BROWN, *Efficient nonlinear solvers for nodal high-order finite elements in 3d*, Journal of Scientific Computing, 45 (2010), pp. 48–63.

[3] 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.

[4] 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).

[5] C. R. DOHRMANN, *A preconditioner for substructuring based on constrained energy minimization*, SIAM Journal on Scientific Computing, 25 (2003), pp. 246–258.

[6] J. DONGARRA, M. A. HEROUX, AND P. LUSCZEK, *High-performance conjugate-gradient benchmark*, International Journal of High Performance Computing Applications, 30 (2016), pp. 3–10.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] G. H. GOLUB AND R. S. VARGA, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods*, (1961).

[13] 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).

[14] 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.

[15] 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.

- 365 [16] A. Klawonn and O. Widlund, *Feti and neumann-neumann iterative substructuring meth-*
 366 *ods: connections and new results*, Communications on Pure and Applied Mathematics: A
 367 Journal Issued by the Courant Institute of Mathematical Sciences, 54 (2001), pp. 57–90.
- 368 [17] D. A. Knoll and D. E. Keyes, *Jacobian-free newton-krylov methods: a survey of approaches*
 369 *and applications*, Journal of Computational Physics, 193 (2004), pp. 357–397.
- 370 [18] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential*
 371 *and integral operators*, United States Governm. Press Office Los Angeles, CA, 1950.
- 372 [19] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur.
 373 Standards, 49 (1952), pp. 33–53.
- 374 [20] J. Li and O. B. Widlund, *On the use of inexact subdomain solvers for bddc algorithms*,
 375 Computer Methods in Applied Mechanics and Engineering, 196 (2007), pp. 1415–1428.
- 376 [21] D. G. Luenberger, *Introduction to linear and nonlinear programming*, vol. 28, Addison-Wesley
 377 Reading, MA, 1973.
- 378 [22] R. E. Lynch, J. R. Rice, and D. H. Thomas, *Direct solution of partial difference equations*
 379 *by tensor product methods*, Numerische Mathematik, 6 (1964), pp. 185–199.
- 380 [23] D. A. May, P. Sanan, K. Rupp, M. G. Knepley, and B. F. Smith, *Extreme-scale multigrid*
 381 *components within petsc*, in Proceedings of the Platform for Advanced Scientific Computing
 382 Conference, 2016, pp. 1–12.
- 383 [24] J. D. McCalpin, *Memory bandwidth and system balance in hpc systems*, Invited talk, Super-
 384 computing, (2016).
- 385 [25] H. Meuer, E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer, *Top 500 list*, 2020.
- 386 [26] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, *Hpl-a portable implementation of*
 387 *the high-performance linpack benchmark for distributed-memory computers*, <http://www.netlib.org/benchmark/hpl/>, (2004).
- 388 [27] D. J. Rixen, C. Farhat, R. Tezaur, and J. Mandel, *Theoretical comparison of the feti and*
 389 *algebraically partitioned feti methods, and performance comparisons with a direct sparse*
 390 *solver*, International Journal for Numerical Methods in Engineering, 46 (1999), pp. 501–
 392 533.
- 393 [28] E. M. Rønquist and A. T. Patera, *Spectral element multigrid. i. formulation and numerical*
 394 *results*, Journal of Scientific Computing, 2 (1987), pp. 389–406.