Tuning Spectral Element Preconditioners for Parallel Scalability on GPUs

Malachi Phillips* Stefan Kerkemeier† Paul Fischer*†‡

Abstract

The Poisson pressure solve resulting from the spectral element discretization of the incompressible Navier-Stokes equation requires fast, robust, and scalable preconditioning. In the current work, a parallel scaling study of Chebyshev-accelerated Schwarz and Jacobi preconditioning schemes is presented, with special focus on GPU architectures, such as OLCF's Summit. Convergence properties of the Chebyshev-accelerated schemes are compared with alternative methods, such as low-order preconditioners combined with algebraic multigrid. Performance and scalability results are presented for a variety of preconditioner and solver settings. The authors demonstrate that Chebyshev-accelerated-Schwarz methods provide a robust and effective smoothing strategy when using p-multigrid as a preconditioner in a Krylov-subspace projector. The variety of cases to be addressed, on a wide range of processor counts, suggests that performance can be enhanced by automated run-time selection of the preconditioner and associated parameters.

1 Introduction

In fluid flow simulations, the incompressibility constraint is often used to bypass fast acoustic waves and thereby allow the solution to evolve on a convective time-scale that is most relevant for many engineering problems. This model leads to a Poisson problem for the pressure that is invariably the stiffest substep in time-advancement of the Navier-Stokes equations. For large 3D problems, which mandate the use of iterative methods, the pressure solve thus typically encompasses the majority of the solution time.

For the spectral element (SE) discretization, the Poisson system matrix contains $\mathcal{O}(Ep^6)$ nonzeros for E elements with a polynomial degree of p (i.e., approximately $n \approx Ep^3$ unknowns.) Through the use of tensor-product-sum factorization, however, the SE matrix-vector product can be effected in only $\approx 7E(p+1)^3$ reads and $12E(p+1)^4$ operations, even in the case of complex geometries [8, 25]. Consequently, the key to fast SE-based flow simulations is to find effective preconditioners tailored to this discretization. Many methods, including geometric p-multigrid approaches with pointwise Jacobi and Chebyshev-accelerated Jacobi smoothers [1, 17, 28], geometric p-multigrid with overlapping Schwarz smoothers [19, 20, 27], and preconditioning via low-order discretizations [3, 5, 24, 25], have been considered.

In this work, we explore the parallel performance of these methods and extend the Schwarz-smoothing based p-multigrid of [20] to support restrictive-additive Schwarz ideas of Cai and Sarkis [7] as well as Chebyshev-accelerated smoothing. In addition, we extend the low-order preconditioning strategy of [3] to run on GPU architectures through the use of AmgX [23] as an algebraic multigrid (AMG) solver for the sparse system. Numerical results are shown for the SE-based pressure Poisson problem as well as for the Navier-Stokes equation. All methods considered are implemented by the authors in the scalable open-source CFD code, nekRS [9]. nekRS started as a fork of libParanumal [6] and uses highly optimized kernels based on the Open Concurrent Compute Abstraction (OCCA) [21]. Special focus is given to performance and scalability on large-scale GPU-based platforms such as OLCF's Summit.

The structure of this paper is as follows. Section 2 outlines the spectral element formulation for the Poisson problem in $\mathbb{R}^3$, as well as describe the various preconditioners considered. A brief description of several model probems of interest are presented in section 3. Numerical results are highlighted in section 4. Finally, a brief summary of the survey of solver techniques considered is provided in section 6.

2 Background and Implementation

We introduce here basic aspects of the SE Poisson discretization and associated preconditioners.

2.1 SE Poisson Discretization Consider the Poisson equation in $\mathbb{R}^3$,

$$-\nabla^2 u = f \text{ for } u, f \in \Omega \subset \mathbb{R}^3 \mapsto \mathbb{R}. \tag{2.1}$$

*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana IL 61801 (malachi2@illinois.edu).

†Mathematics and Computer Science, Argonne National Laboratory, Lemont, IL 60439 (kerkemeier@anl.gov).

‡Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana IL 61801 (fischerp@illinois.edu).

Boundary conditions for the pressure Poisson equation are either periodic, Neumann, or Dirichlet, with the latter typically applicable only on a small subset of the domain boundary, $\partial\Omega_D$, correspondong to an outflow condition. Consequently, Neumann conditions apply over the majority (or all) of the domain boundary, which makes the pressure problem more challenging than the standard all-Dirichlet case.

The SE discretization of (2.1) is based on the weak form: *Find* $u \in X_0^p$ *such that,*

$$(\nabla v, \nabla u)_p = (v, f)_p \quad \forall v \in X_0^p, \tag{2.2}$$

where X_0^p is a finite-dimensional approximation comprising the basis functions used in the SE discretization, $\phi_j(\mathbf{x})$, $j = 1, \ldots, n$, that vanish on $\partial\Omega_D$ and $(\cdot,\cdot)_p$ represents the discrete L^2 inner product based on Gauss-Lobatto-Legendre quadrature in the reference element, $\hat{\Omega} := [-1, 1]^3$. The basis functions allow us to represent the solution, u, as $u(\mathbf{x}) = \sum u_j \phi_j(\mathbf{x})$, leading to a linear system of unknown basis coefficients,

$$A\underline{u} = B\underline{f}, \tag{2.3}$$

with respective mass- and stiffness-matrix entries, $B_{ij} := (\phi_i, \phi_j)_p$, and $A_{ij} := (\nabla\phi_i, \nabla\phi_j)_p$.

Ω is tesellated into nonoverlapping hexahedral elements, Ω^e, for $e = 1, \ldots, E$, with isoparametric mappings from $\hat{\Omega}$ to Ω^e provided by $\mathbf{x}^e(r, s, t) = \sum_{i,j,k} \mathbf{x}^e_{ijk} h_i(r) h_j(s) h_k(t)$, for $i, j, k \in [0, p]$. Each $h_*(\xi)$ is a pth-order Lagrange cardinal polynomial on the Gauss-Lobatto-Legendre (GLL) qudrature points, $\xi_j \in [-1, 1]$. Similarly, the test and trial functions u, v are written in local form as $u^e(r, s, t) = \sum_{i,j,k} u^e_{ijk} h_i(r) h_j(s) h_k(t)$. Continuity is ensured across the interface between adjacent elements by enforcing $u^e_{ijk} = u^e_{\hat{i}\hat{j}\hat{k}}$ when $\mathbf{x}^e_{ijk} = \mathbf{x}^e_{\hat{i}\hat{j}\hat{k}}$. From this, a global-to-local degree-of-freedom mapping, $\underline{u} := \{u_l\} \longrightarrow \underline{u}_L := \{u^e_{ijk}\}$, can be represented by a Boolean matrix Q, such that $\underline{u}_L = Q\underline{u}$. The assembled stiffness matrix is then $A = Q^T A_L Q$, where $A_L = \text{block-diag}(A^e)$ comprises the local stiffness matrices, A^e. Similarly, $B = Q^T B_L Q$. The SE formulation uses coincident GLL quadrature and nodal points, such that B^e is diagonal. Moreover, A^e is never formed, as it would contain $O(p^6)$ nonzeros in the general case. Rather, the tensor-product-sum factorization [25] allows for $A\underline{u}$ to be evaluated in $O(Ep^4)$ time with $O(Ep^3)$ storage, as described in detail in [8].

2.2 Preconditioners In the current study, all preconditioners are applied in the context of restarted GMRES. Although A is symmetric positive definite (SPD), many of the preconditioners are asymmetric. Further, a recent study [11] has shown the benefits of projection-based GMRES over flexible conjugate gradients (FCG) because the effectiveness of the overall pressure solution strategy (which includes projection onto prior solutions [12]) generally ensures a low enough iteration count, k, such that the $O(k^2)$ costs in GMRES are not overly onerous. We note, however, that FCG may yield very low iteration counts in certain cases (e.g., 1–2 iterations per step, as in [9]), in which case we use FCG rather than GMRES if it yields faster runtimes.

2.2.1 SEMFEM In [25], Orszag suggested that constructing a sparse preconditioner based on the low-order discretizations with nodes coinciding with those of the high-order discretization would yield bounded condition numbers and, under certain constraints, can yield $\kappa(M^{-1}A) \sim \pi^2/4$ for second-order Dirichlet problems. This observation has led to the development of preconditioning techniques based on solving the resulting low-order system [3, 5, 24].

In the current work, we employ the same low-order discretization considered in [3]. Each of the vertices of the hexahedral element is used to form one low-order, tetrahedral element, resulting in a total of eight low-order elements for each GLL sub-volume in each of the high-order hexahedral elements. This low-order discretization is then used to form the sparse operator, A_F. The so-called weak preconditioner, A_F^{-1}, is used to precondition the system. Algebraic multigrid (AMG), implemented in CUDA in AmgX [23], is used with the following setup to solve the low order system:

- PMIS coarsening
- 0.25 strength threshold
- Extended + i interpolation ($p_{max} = 4$)
- Damped Jacobi relaxation (0.9)
- One V-cycle for preconditioning
- Smoothing on the coarsest level

We denote this preconditioning strategy as SEMFEM.

2.2.2 p-multigrid, Schwarz Smoothers Another preconditioning strategy for the SE-based Poisson problem is to use geometric p-multigrid (pMG). The classical single pass V-cycle is summarized in Algorithm 1. In our application, we limit pMG preconditioning to a single V-cycle pass. Alternative strategies, such as F- or W-cycle multigrid are not considered.

The SE-based additive Schwarz method (ASM) presented in [19, 20] solves local Poisson problems on subdomains that are extensions of the spectral elements. The formal definition of the ASM preconditioner (or pMG smoother) is

$$\underline{s} = \sum_{e=1}^{E} W_e R_e^T \bar{A}_e^{-1} R_e \underline{r}, \tag{2.4}$$

Algorithm 1 Single pass multigrid V-cycle

$\mathbf{x} = \mathbf{x} + \mathbf{M}(\mathbf{b} - \mathbf{A}\mathbf{x})$ // smooth
$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ // re-evaluate residual
$\mathbf{r}_C = \mathbf{P}^T\mathbf{r}$ // coarsen
$\mathbf{e}_C = \mathbf{A}_C^{-1}\mathbf{r}_C$ // solve/re-apply V-cycle
$\mathbf{e} = \mathbf{P}\mathbf{e}_C$ // prolongate
$\mathbf{x} = \mathbf{x} + \mathbf{e}$ // update solution
$\mathbf{x} = \mathbf{x} + \mathbf{M}(\mathbf{b} - A\mathbf{x})$ // post smoothing

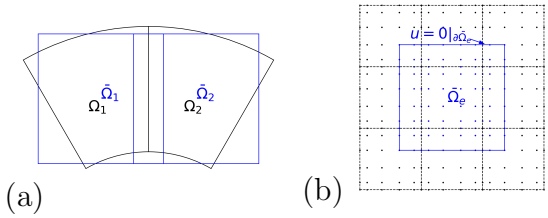

Figure 1: (a) Box-like approximation (b) overlapping domain

where R_e is the restriction matrix that extracts nodal values of the residual vector that correspond to each overlapping domain, as indicated in Fig. 1b. To improve the smoothing properties of the ASM, we introduce the diagonal weight matrix, W_e, which scales each nodal value by the inverse of the number of subdomains that share that node. Although it compromises symmetry, post-multiplication by W_e was found to yield superior results to pre- and post-multiplication by $W_e^{\frac{1}{2}}$ [20, 27].

In a standard Galerkin ASM formulation, one would use $\bar{A}_e = R_e A R_e^T$, but such an approach would compromise the $O(p^3)$ storage complexity of the SE method. To construct fast inverses for $\bar{A}_e$, we approximate each deformed element as a simple box-like geometry, as demonstrated in Fig. 1a. These boxes are then extended by a single degree-of-freedom in each spatial dimension to form overlapping subdomains with $\bar{p}^3 = (p+3)^3$ interior degrees-of-freedom in each domain. (See Fig. 1). The approximate box domain enables the use of the fast diagonalization method (FDM) to solve for each of the overlapping subdomains, which can be applied in $O(Ep^4)$ time in $\mathbb{R}^3$ [20]. The extended-box Poisson operator is separable, with the 3D form

$$\bar{A} = B_z \otimes B_y \otimes A_x + B_z \otimes A_y \otimes B_x + A_z \otimes B_y \otimes B_x, \tag{2.5}$$

where each B_*, A_* represents the extended 1D mass-stiffness matrix pairs along the given dimension [20]. The FDM begins with a preprocessing step of solving a series of small, $\bar{p} \times \bar{p}$, generalized eigenvalue problems,

$$A_* \underline{s}_i = \lambda_i B_* \underline{s}_i \tag{2.6}$$

and defining $S_* = (\underline{s}_1 \dots \underline{s}_{\bar{p}})$ and $\Lambda_* = \mathrm{diag}(\lambda_i)$, to yield the similarity transforms

$$S_*^T A_* S_* = \Lambda_*, \quad S_*^T B_* S_* = I. \tag{2.7}$$

From these, the inverse of the local Schwarz operator is

$$\bar{A}^{-1} = (S_z \otimes S_y \otimes S_x) D^{-1} (S_z^T \otimes S_y^T \otimes S_x^T), \tag{2.8}$$

where

$$D = I \otimes I \otimes \Lambda_x + I \otimes \Lambda_y \otimes I + \Lambda_z \otimes I \otimes I \tag{2.9}$$

is a diagonal matrix. This process is repeated for each element, at each multigrid level save for the coarsest one. Note that the per-element storage is only $3\bar{p}^2$ for the S_* matrices and $\bar{p}^3$ for D. At each multigrid level, the local subdomain solves are used as a smoother. On the coarsest level ($p = 1$), however, BoomerAMG [13] is used to solve the system with the same parameters as the AmgX solver in section 2.2.1, except using Chebyshev smoothing. Unless otherwise noted, all pMG preconditioners use a single BoomerAMG V-cycle iteration in the coarse-grid solve.

Presently, we also consider a restrictive additive Schwarz (RAS) version of (2.4), wherein overlapping values are not added after the action of the local FDM solve, following [7]. RAS has the added benefit of reducing the amount of communication required in the smoother. (Formally, RAS can be implemented by simply changing W_e.) Note that, in the case of ASM and RAS smoothers without Chebyshev-acceleration, an additive V-cycle with no post-smoothing is used to avoid residual re-evaluation in Algorithm 1 [10].

2.2.3 Chebyshev Acceleration A notable improvement over standard Jacobi-smoothed multigrid is to use Chebyshev-acceleration [1,28], as described in Algorithm 2 for a given surrogate smoother, S. While S is typically based on Jacobi smoothing (e.g., [17]), it is also possible to consider the use of overlapping-Schwarz (2.4) as the smoother, which is a new approach that we explore here.

Algorithm 2 requires approximate spectral bounds, $(\lambda_{min}, \lambda_{max})$, of the smoothed operator, SA. Using 10 rounds of Arnoldi iteration, we generate $\tilde{\lambda}$ as an initial proxy for λ_{max}. The bounds employed in Algorithm 2 are then determined through sensitivity analysis similar to that performed by Adams and coworkers [1]. The analysis is conducted using a challenging model problem, namely the Kershaw case of subsection 3.1, with $\varepsilon = 0.3$, $E = 24^3, p = 7$, and relative (2-norm) residual tolerance of 10^{-8}. A second-order Chebyshev-accelerated ASM smoother is used in the $p = 7, 3, 1$

Algorithm 2 Chebyshev smoother

$\theta = \frac{1}{2}(\lambda_{max}+\lambda_{min})$, $\delta = \frac{1}{2}(\lambda_{max}-\lambda_{min})$, $\sigma = \frac{\theta}{\delta}$, $\rho_1 = \frac{1}{\sigma}$

$r = S(b - Ax)$, $d_1 = \frac{1}{\theta}r$, $x_1 = 0$

for $k = 1, \ldots,$ `chebyshevOrder` **do**

$x_{k+1} = x_k + d_k$

$r_{k+1} = r_k - SAd_k$

$\rho_{k+1} = \frac{1}{2\sigma - \rho_k}$

$d_{k+1} = \rho_{k+1}\rho_k d_k + \frac{2\rho_{k+1}}{\delta}r_{k+1}$

end for

$x_{k+1} = x_k + d_k$

return x_{k+1}

$\lambda_{min}\backslash\lambda_{max}$	$0.9\tilde{\lambda}$	$0.95\tilde{\lambda}$	$1.0\tilde{\lambda}$	$1.1\tilde{\lambda}$	$1.2\tilde{\lambda}$	$1.3\tilde{\lambda}$
$0.0\tilde{\lambda}$	-	-	-	-	-	-
$0.025\tilde{\lambda}$	-	-	-	110	64	48
$0.05\tilde{\lambda}$	-	-	-	50	40	38
$0.1\tilde{\lambda}$	-	-	124	40	38	38
$0.2\tilde{\lambda}$	159	45	43	42	43	44
$0.25\tilde{\lambda}$	47	45	44	44	45	46

Table 1: Iteration counts for the $(\lambda_{min}, \lambda_{max})$ sensitivity study. Omitted entries failed to converge in 1000 iterations.

pMG V-cycle preconditioner, which we denote as the current default preconditioner in nekRS.

The results of the sensitivity study are shown in Table 1. The required iteration counts are seen to be sensitive to underestimation of λ_{max}, as also found by Adams and coworkers [1]. On the other hand, results are less sensitive to the minimum eigenvalue estimate, provided $\lambda_{min} > 0$, as this delimits between the low frequencies that are handled by the coarse grid correction and the high frequencies that are eliminated by the action of the smoother. The choice of the bounds (relative to $\tilde{\lambda}$) is not universal: (1/30,1.1) [1], (0.3,1) [2], (0.25,1) [28], and (1/6,1) [32] have all been considered. Based on the results of Table 1, we set $(\lambda_{min}, \lambda_{max}) = (0.1, 1.1)\tilde{\lambda}$ as a conservative choice.

3 Test Cases

We describe four model problems that are used to test the SE preconditioners. The first is a stand-alone Poisson solve, using variations of the Kershaw mesh. The others are modest-scale Navier-Stokes problems, where the pressure Poisson problem is solved over multiple timesteps. The problem sizes are listed in Table 2 and range from relatively small (n=21M points) to moderately large (n=645M).[1]

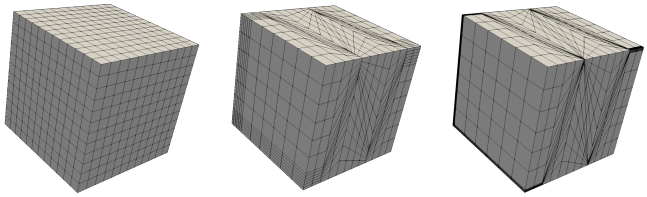

Figure 2: Kershaw, $E = 12^3, p = 1$. $\varepsilon = 1.0, 0.3, 0.05$.

Case Name	E	p	n
146 pebble (Fig. 3a)	62K	7	21M
1568 pebble (Fig. 3b)	524K	7	180M
67 pebble (Fig. 3c)	122K	7	42M
Speed bump (Fig. 3d)	885K	9	645M

Table 2: Problem discretization parameters.

3.1 Poisson The Kershaw family of meshes [15, 16]. has been proposed as the basis for a high-order Poisson-solver benchmark by Center for Efficient Exascale Discretization (CEED) within the DOE Exascale Computing Project (ECP). This family is parametrized by an anisotropy measure, $\varepsilon = \varepsilon_y = \varepsilon_z \in (0, 1]$, that determines the degree of deformation in the y and z directions. As ε decreases, the mesh deformation and aspect ratio increase along with it. The Kershaw mesh is shown in Fig. 2 for several values of ε. The domain $\Omega = [-1/2, 1/2]^3$ with Dirichlet boundary conditions on $\partial\Omega$. The right hand side for (2.1) is set to

$$f(x, y, z) = 3\pi^2 \sin(\pi x) \sin(\pi y) \sin(\pi z) + g, \tag{3.10}$$

where $g(x, y, z)$ is a random, continous vector vanishing on $\partial\Omega$. The linear solver terminates after reaching a relative residual reduction of 10^{-8}. Since this test case solves the Poisson equation, there is no timestepper needed for the model problem.

3.2 Navier-Stokes For the pressure-Poisson tests, four flow cases are considered, as depicted in Fig. 3. The first three cases corresponds to turbulent flow through a cylindrical packed-bed with 146, 1568, and 67 spherical pebbles. The 146 and 1568 pebble cases are from Lan and coworkers [18]. The 67 pebble case is constructed using a tet-to-hex meshing strategy by Yuan and coworkers [31]. The first two bed flows are at Reynolds number $Re_D = 5000$, based on sphere diameter, D, while the 67 pebble case is at Reynolds

[1] Larger cases for recent full-scale runs on Summit with n=51B are reported in [11].

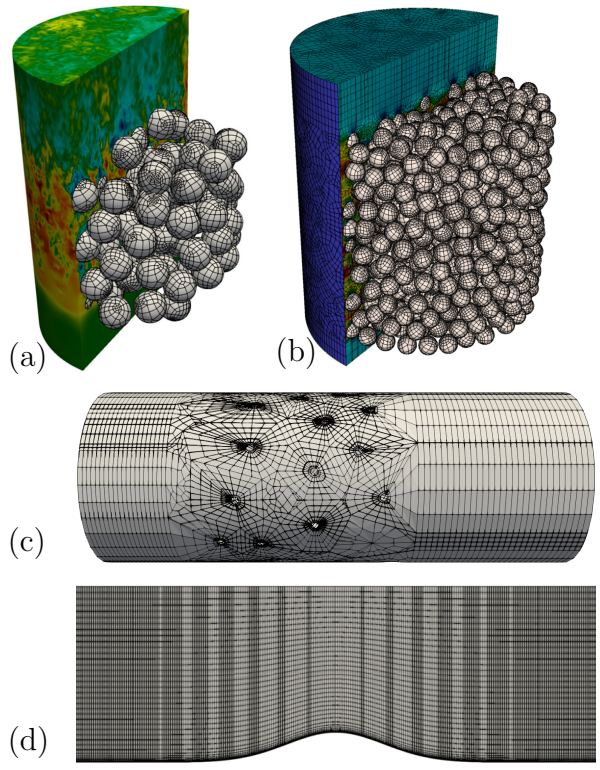

Figure 3: Navier-Stokes cases: pebble-beds with (a) 146, (b) 1568, and (c) 67 spheres; (d) Boeing speed bump.

number $Re_D = 1460$. Time advancement is based on a two-stage 2nd-order characteristics timestepper with CFL=4 ($\Delta t = 2 \times 10^{-3}$ $\Delta t = 5 \times 10^{-4}$, and $\Delta t = 5 \times 10^{-5}$ for the 146, 1568, and 67 pebble cases). An absolute pressure solver tolerance of 10^{-4} is used. A restart at $t = 10$, $t = 20$, and $t = 10.6$ convective time units is used for the 146, 1568, and 67 pebble cases, respectively, to provide an initially turbulent flow.

The fourth case, shown in Fig. 3d, is a direct numerical simulation (DNS) of seperated turbulent flow over a speed bump at $Re = 10^6$. This test case was designed by Boeing to provide a flow that exhibits separation. A DNS of the full 3D geometry, however, remains difficult [26]. Therefore, this smaller example proves a useful application for benchmarking solver performance. This case uses a 2nd-order timestepper with CFL=0.8 ($\Delta t = 4.5 \times 10^{-6}$) and an absolute pressure-solve tolerance of 10^{-5}. A restart at $t = 5.6$ convective time units is used for the initial condition.

In all cases, solver results are collected over 2000 timesteps. At each step, the solution is projected onto a space of up to 10 prior solution vectors to generate a high-quality initial guess, $\bar{\underline{u}}$. Projection is standard practice in nekRS as it can reduce the initial residual by orders of magnitude at the cost of just one or two matrix-vector products in A per step [12].

The perturbation solution, $\delta\underline{u} := \underline{u} - \bar{\underline{u}}$, is typically devoid of slowly evolving low wave-number content. Moreover, the initial residual is oftentimes sufficiently small that the solution converges in $k < 5$ iterations, such that the $O(k^2)$ overhead of GMRES is small. Testing the preconditioners under these conditions ensures that the conclusions drawn are relevant to the application space.

4 Results

Here we consider the solver performance results for the test cases of Section 3. We assign a single MPI rank to each GPU and denote the number of ranks as P. All runs are on Summit. Each node on Summit consists of 42 IBM Power9 CPUs and 6 NVIDIA V100 GPUs. We use 6 GPUs per node unless $P < 6$. In the following, we denote a pMG preconditioner using η-order Chebyshev-accelerated ξ smoother with a multigrid schedule of Π as Cheby$\xi(\eta)$,Π. A wide range of preconditioning strategies is considered.

4.1 Kershaw Mesh The Kershaw study comprises six tests. For each of two studies, we consider the regular box case ($\varepsilon = 1.0$), a moderately skewed case ($\varepsilon = 0.3$), and a highly skewed case ($\varepsilon = 0.05$). The first study is a standard weak-scale test, where P and E are increased, while the polynomial order is fixed at $p = 7$ and the number of gridpoints per GPU is set to $n/P = 2.67M$. The range of processors is P=6 to 384. The second study is a test of the influence of polynomial order on conditioning, with $P = 24$ and $n/P = 2.88M$ fixed, while p ranges from 3 to 10. Both cases use GMRES(20).

The results of the weak-scaling study are shown in Fig. 4. For all values of ε, the iteration count exhibits a dependence on problem size, as seen in Fig. 4a,d,g, especially in the highly skewed case ($\varepsilon = 0.05$). The time-per-solve also increases with n, in part due to the increase in iteration count, but also due to increased communication overhead as P increases. This trend is not necessarily monotonic, as shown in Fig. 4c,f. In the case of $\varepsilon = 0.3$ Cheby-RAS(2),(7,3,1), a minor fluctuation in the iteration count from the $P = 6$ to $P = 12$ case causes the greater than unity parallel efficiency. For $\varepsilon = 1.0$ Cheby-Jac(2),(7,5,3,1), the effects of system noise on the $P = 6$ run causes the greater than unity parallel efficiency for the $P = 12$ run.

Table 3 indicates the maximum number of of neighboring processors for the assembly (QQ^T) graph of A, which increases with P. In addition, the number of grid points per GPU ($n/P = 2.67M$) is relatively low.

Nodes	1	2	4	8	16	32	64
Max Neighbors	5	11	16	24	20	24	29

Table 3: Max neighbors, Kershaw.

These two factors cause an increased sensitivity of the problem to the additional communication overhead as the number of GPUs is initially increased. The number of neighbors, however, will saturate at larger (i.e., production-level) processor counts.

Lastly, the relative preconditioner performance depends on ε. Fig. 4b, e, h show that, for the easy $\varepsilon = 1.0$ case, a pMG scheme with a smoother that is cheap to apply is best, such as Cheby-RAS(1),(7,3,1), Cheby-Jac(2),(7,5,3,1), and ASM,(7,3,1). However, as ε decreases, more robust pMG smoothers such as Cheby-RAS(2),(7,3,1) and SEMFEM result in lower time to solution. Once $\varepsilon = 0.05$ (Fig. 4h), the problem is sufficiently challenging that SEMFEM overtakes the pMG based preconditioning schemes. This indicates that, in the highly skewed case in which the maximum element aspect ratio increases, the pMG preconditioner is not as effective as the SEMFEM preconditioner.

The influence of polynomial order is illustrated in Fig. 5. For $\varepsilon = 1.0$, iteration counts are essentially p-independent, as seen in Fig. 5a. For $\varepsilon = 0.3$, however, a slight upward trend in the iteration count is observed for SEMFEM and for the pMG preconditioners with ASM, RAS, and Chebyshev-Jacobi smoothing (Fig. 5b). Similarly, $\varepsilon = 0.05$ exhibits a dependence between the iteration count and the polynomial order for all preconditioners (Fig. 5c). Figures 5d, e and f demonstrate that the *time* per pressure solve is strongly dependent polynomial order. For SEMFEM, there is an increase in time as a result of increased overhead for the underlying AMG solver. For the pMG-based methods, higher orders are generally faster. This performance gain can be attributed to surface-to-volume effects in the evaluation of the operators and smoothers. Application of A^e and (smoother) S^e is highly vectorizable, whereas application of QQ^T (assembly) involves a significant amount of indirect addressing. The discrete surface to volume ratio for a spectral element of order $p = 7$ is $296/512 \approx 60\%$. For lower p this value is larger. This situation is exacerbated in the case of Schwarz-based smoothers, where the overlap contributes substantially to the work and communication for the small-element (i.e., low-p) cases. In addition, nekRS [9] uses a single element per thread block, which limits the amount of work available for a streaming multiprocessor for relatively small polynomial orders.

4.2 Navier-Stokes Results We consider scalability of nekRS for the cases of Fig. 3. All simulations except one use GMRES(15) with an initial guess generated by A-conjugate projection onto 10 prior solutions [12]. Due to memory constraints, the 1568-pebble case with $P = 24$ uses GMRES(10) with only 5 solution-projection vectors. For each case, two pMG schedules are considered: $(7, 5, 3, 1)$ and $(7, 3, 1)$ for $p = 7$; and $(9, 7, 5, 1)$ and $(9, 5, 1)$ for $p = 9$. Other parameters, such as the Chebyshev order and the number of coarse grid BoomerAMG V-cycles are also varied. Results are shown in Figs. 6,7. The plots relate the effective work rate per node, measured as the gridpoints n (as shown in Table 2) solved per second per node, to the time-to-solution. The y-axis notes the drop in the relative work rate, which corresponds to a lower parallel efficiency, as the strong scale limit is reached, while the x-axis denotes the time-to-solution. Each node consists of 6 GPUs, hence $P = 6 \times$ nodes.

In all the performance tests conducted, the pMG preconditioner with Chebyshev-Jacobi smoothing is outperformed by the other preconditioners, whether using one or two V-cycle iterations in the AMG coarse-grid solve. For each case, the fastest preconditioner scheme varies. In the 146 pebble case (Fig. 7a), using Cheby-RAS(2),(7,5,3,1) yields the smallest time per pressure solve. However, in the 1568 pebble case (Fig. 7b), SEMFEM is a moderate improvement over the second best preconditioner, Cheby-ASM(2),(7,5,3,1). pMG with a $(9, 5, 1)$ schedule and Chebyshev-RAS (of any order) yield the best scalability and lowest time per pressure solve for the Boeing speed bump case (Fig. 6). The Chebyshev-accelerated Schwarz schemes are not always the fastest, however. For the 67 pebble case (Fig. 7c), Cheby-Jac(2),(7,5,3,1) is comparable to Cheby-RAS(2),(7,5,3,1) and are the two fastest pMG based preconditioners. However, SEMFEM is significantly faster than the other preconditioners for this case.

Also considered is a hyrbid two-level pMG/SEMFEM approach wherein SEMFEM is used as the solver for the coarse level. For $p = 7$, a $(7, 6)$ schedule with 2nd order Chebyshev-accelerated ASM smoothing on the $p = 7$ level and SEMFEM solver on the $p = 6$ level, denoted as Cheby-ASM(2),(7,6) + SEMFEM, is used. Similarly, Cheb-ASM(2),(7,5) + SEMFEM and Cheb-ASM(2),(7,3) + SEMFEM are considered. For $p = 9$, a $(9, 8)$, $(9, 7)$, $(9, 5)$, and $(9, 3)$ hybrid pMG/SEMFEM approach is considered, denoted as Cheb-ASM(2),(9,8) + SEMFEM, Cheb-ASM(2),(9,7) + SEMFEM, Cheb-ASM(2),(9,5) + SEMFEM, and Cheb-ASM(2),(9,3) + SEMFEM, respectively. In the pebble cases shown in Fig. 7a,c, this hybrid approach performs somewhere between the SEMFEM and Cheby-ASM(2),(7,3,1) pre-

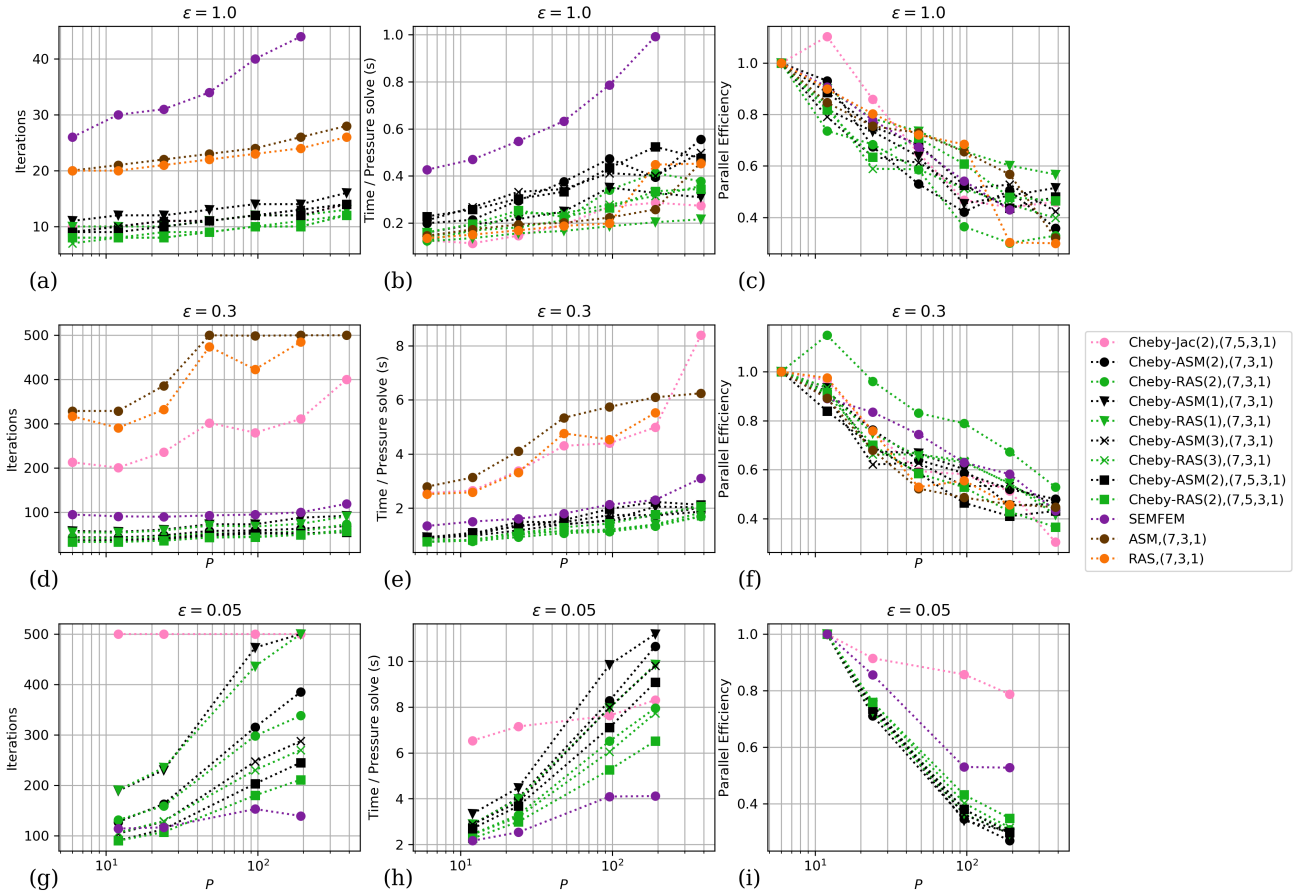

Figure 4: Kershaw weak scaling, GMRES(20). $n/P = 2.67M$.

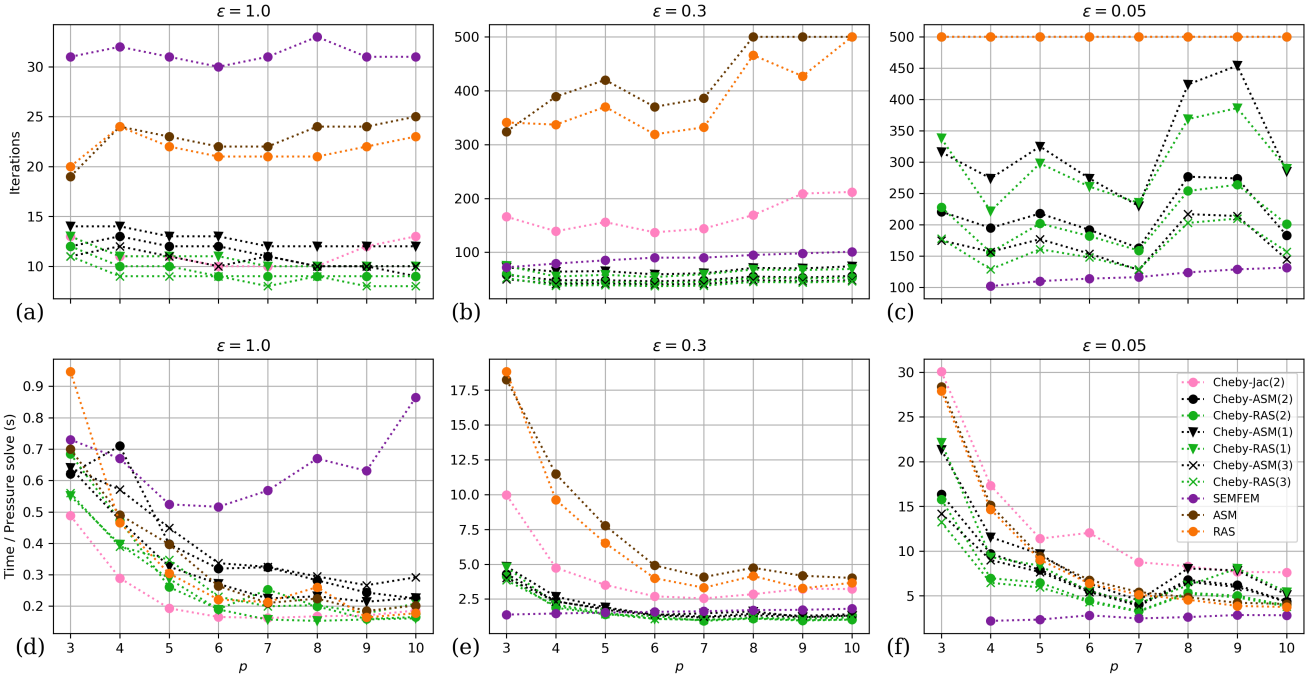

Figure 5: Kershaw order dependence, GMRES(20). $n/P = 2.88M$.

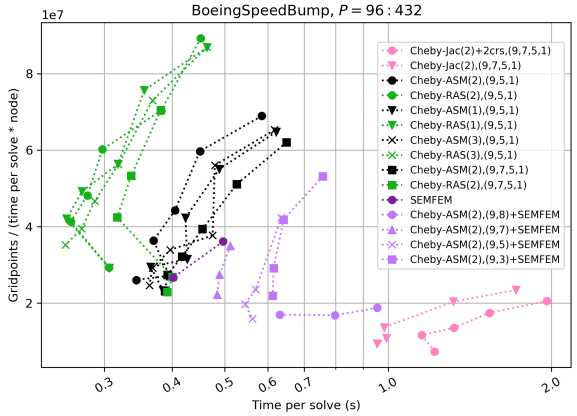

Figure 6: Strong scaling results on Summit for the Navier-Stokes case of Fig. 3d.

conditioners. In the Boeing speed bump case, however, Fig. 6 demonstrates that this approach is not as performant as either the SEMFEM or Cheby-ASM(2),(9,5,1) preconditioners.

Solver performance degrades whenever n/P is sufficiently small, regardless of the solver considered. For the various preconditioners considered, at 2 nodes ($n/P = 1.75M$), the strong-scale limit of 80% efficiency is far surpassed in the 146 pebble case. This leaves the effective strong scale limit at 1-2 nodes ($n/P = 3.5$ to $1.75M$). For the 1568 pebble case, 12 nodes ($n/P = 2.5M$) yields a parallel efficiency around 60-70%, depending on the specific solver. The 67 pebble cases reaches 70% effieciency on 3 nodes ($n/P = 2.3M$). The parallel efficiency for the Boeing speed bump case for the fastest time-to-solution preconditioners drops below 70% when using more than 48 nodes ($n/P = 2.24M$).

While the effect of mesh quality metrics, such as the max aspect ratio and scaled Jacobian, on solver convergence has been studied by Mittal *et al.* [22], predicting the optimal preconditioner settings from mesh quality metrics is not obvious. While the maximum aspect ratio is an important metric for mesh quality, it alone cannot explain the apparent poor performance of the pMG preconditioners in the 67 pebble case, Fig. 7c. Consider, for example, that the Boeing speed bump case has a larger maximum aspect ratio (255) than the 67 pebble case (204), but does not exhibit this poor performance in the pMG preconditioners. The minimum scaled Jacobian, however, is at least an order of magnitude smaller in the 67 pebble case (5.97×10^{-3}) as compared to the other cases (e.g., .996 for the Boeing speed bump case, 4.31×10^{-2} and 2.59×10^{-2} for the 146, 1568 pebble cases, respectively). This may, in turn, explain why the pMG preconditioners in the 67 pebble case were signficiantly suboptimal compared to the SEMFEM preconditioner. This demonstrates, however, that a user cannot simply rely on, e.g., the maximum aspect ratio when deciding whether or not to use SEMFEM as a preconditioner. This inability to correctly identify preconditioner settings based on mesh quality metrics alone motivates the introduction of an auto-tuner to choose preconditioner parameters during runtime.

5 Auto-tuning for Production Simulations

The results of the preceding section provides a small window into the varieties of performance behavior encountered in actual production cases, which span a large range of problem sizes, domain topologies, mesh qualities. Moreover, production simulations are solved across a range of architectures having varying on-node and network performance, interconnect topologies, and processor counts. One frequently encounters situations where certain communication patterns might be slower under a particular MPI version on one platform versus another. Unless a developer has access to that platform, it's difficult to measure and quantify the communication overhead. Processor count alone can be a major factor in preconditioner selection: large processor counts have relatively high coarse-grid solve costs that can be mitigated by doing more smoothing at the fine and intermediate levels. How much more is the open question. The enormity of parameter space, particularly "in the field" (i.e., users working on unknown platforms) limits the effectiveness of standard complexity analysis in selecting the optimal preconditioner for a given user's application.

From the user's perspective, there is only one application (at a time, typically), and one processor count of interest. Being able to provide optimized performance—tuned to the application at hand, which includes the processor count—is thus of paramount importance. Auto-tuning provides an effective way to deliver this performance. Auto-tuning of preconditioners has been considered in early work by Imakura *at al.* [14] and more recently by Yamada *at al.* [30] and by Brown *et al.* [4]. The latter work couples their auto-tuning with local Fourier analysis to guide the tuning process.

In large-scale fluid mechanics applications, auto-tuning overhead is typically amortized over 10^4–10^5 timesteps (i.e., pressure solves) *per run* (and more, over an entire simulation campaign). Moreover, auto-tuning is of particular importance for problems at large processor counts because these cases often have long queue times, which preclude making multiple job submissions in order to tweak parameter settings. Failure to optimize, however, can result in significant opportunity costs. For example, in [11] the authors realized a factor of 2.8 speedup in time-per-step for a 352,000-pebble-

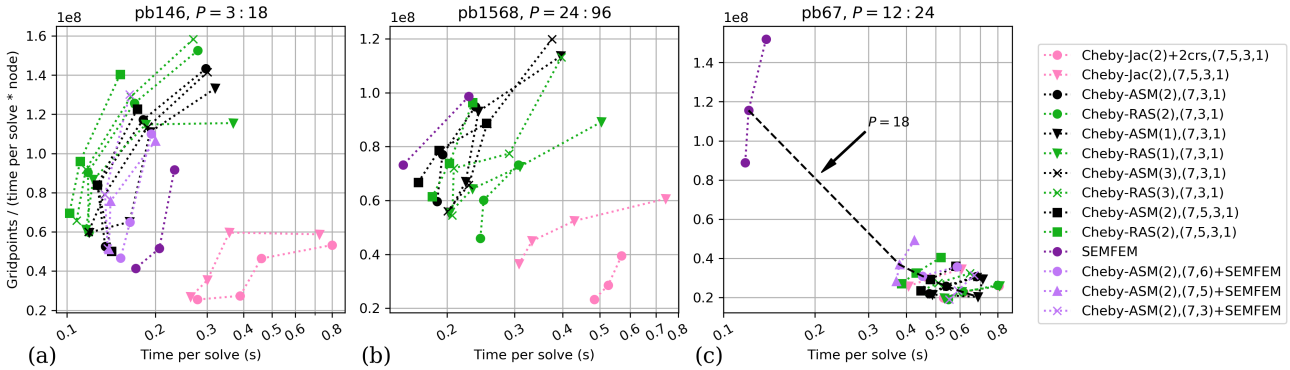

Figure 7: Strong scaling results on Summit for the Navier-Stokes cases of Fig. 3a,b,c. Iso-processor count line illustrated in (c). A user running on a specified number of processors should use the lowest time-to-solution preconditioner along this line.

Case Name	GLL Spacing (min/max)	Scaled Jacobian (min/max/avg)	Aspect Ratio (min/max/avg)
146 pebble	1.01×10^{-3} / .32	4.31×10^{-2} / .977 / .419	1.07 / 56.9 / 7.14
1568 pebble	2.21×10^{-4} / .3	2.59×10^{-2} / .99 / .371	1.12 / 108 / 12.6
67 pebble	4.02×10^{-5} / .145	5.97×10^{-3} / .970 / .38	1.17 / 204 / 13.2
Boeing Speed Bump	8.34×10^{-7} / 2.99×10^{-3}	.996 / 1 / 0.999	6.25 / 255 / 28.1

Table 4: Mesh quality metrics for cases from Fig. 3.

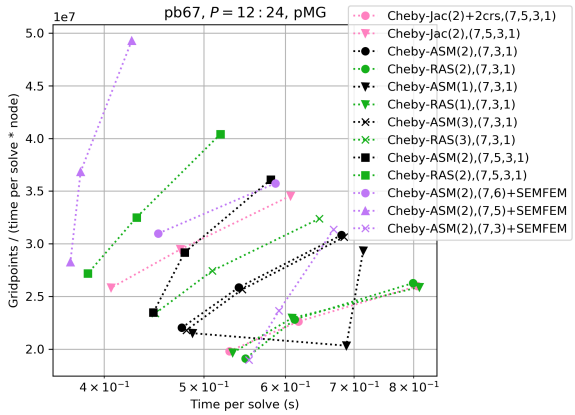

Figure 8: Same as Fig. 7c, pMG preconditioners only.

bed simulation (n=51B gridpoints) through a sequence of tuning steps. Even if there were 100 configurations in the preconditioner parameter space, an auto-tuner could visit each of these in succession 5 times each within the first 500 steps and have expended only a modest increment in overhead when compared to the cost of submitting many jobs (for tuning) or to the cost of 10,000 steps for a production run.

As a preliminary step, the authors consider constructing a small subset of the true search space (which could include, e.g., other cycles, schedules, or smoothers) for use in a nascent auto-tuner. Across all cases, with exception to the 67 pebble case, pMG preconditioning with Chebyshev-accelerated ASM or RAS smoothing is the fastest solver or is comparable to SEMFEM. The choice of Chebyshev order and multigrid schedule, moreover, contributes only a modest $\approx$ 10-20% improvement to the overall time-to-solution in most cases, all else being equal. This makes the default Cheby-ASM(2),(7,3,1) or Cheby-ASM(2),(9,5,1) preconditioner reasonably performant. However, in order to avoid the situation encountered in the 67 pebble case, SEMFEM is added to the considered search space. The parameters in Table 5 are chosen as they include optimal or near-optimal preconditioner settings for the results in Figs. 6, 7, while still restricting the search space to something that is amenable to exhaustive search. The authors note, however, that this parameter space may not reflect the various factors affecting the performance of the preconditioners at especially large P or on different machine architectures.

During the first timestep, our simple auto-tuner performs an exhaustive search over the small parameter space identified in Table 5. While this space is limited compared to the true search space, the resultant preconditioners selected are effective. In the 146 pebble case (Fig. 7a), Cheby-RAS(2),(7,3,1) is identified as the preconditioner on each of the processor counts, which was comparable in peformance to the best preconditioner. The auto-tuner chose the optimal SEMFEM for the 1568 pebble case (Fig. 7b). SEMFEM was identified at the preconditioner for the 67 pebble case on all

	Parameters
Solver	preconditioned GMRES
Preconditioner	pMG, SEMFEM
pMG	$p = 7$: Cheby-ASM(2),(7,3,1), Cheby-RAS(2), (7,3,1), Cheby-Jac(2), (7,5,3,1) $p = 9$: Cheby-ASM(2),(9,5,1), Cheby-RAS(2), (9,5,1), Cheby-Jac(2), (9,7,5,1)
Coarse grid	single boomerAMG V-cycle
SEMFEM	single AmgX V-cycle

Table 5: Solver parameter space considered in the auto-tuner. A pMG preconditioner using an η-order Chebyshev-accelerated ξ smoother with a multigrid schedule of Π is denoted as Cheby-$\xi(\eta)$,Π.

processor counts, which Fig. 7c confirms. In the Boeing speed bump case (Fig. 6), the auto-tuner chose Cheby-RAS(2),(9,5,1) across all processor counts, which was either the optimal or near-optimal preconditioner for the problem.

Note that the auto-tuner selects the fastest method at a *fixed-processor count* (i.e., whatever the user has selected). Consider, for example, the 67 pebble case on $P = 18$ GPUs (dashed black line, Fig. 7c). The goal of the auto-tuner is to select the preconditioner with lowest time-to-solution along the user-specified iso-processor count line. However, in the results discussed above, there was no change with respect to a change in the processor count. We note that our principal objective is not to squeeze out a few percent over a raft of good choices, but rather to ensure that a case does not run with an unfortunate set of parameters for which the performance is significantly substandard. This primitive auto-tuning technique proves effective at preventing the selection of highly suboptimal preconditioners. We have implemented this for production use and will continue to update and refine the strategy moving forward.

6 Conclusions and Future Work

In this work, we introduce Chebyshev-accelerated ASM and RAS smoothers for use in p-multigrid preconditioners for the spectral element Poisson problem. Further, we compare the performance of Schwarz-based smoothers, Chebyshev-accelerated Jacobi smoothing, and SEMFEM-based preconditioning for a suite of challenging test problems. We conclude that the Chebyshev-accelerated Schwarz smoothers with p-multigrid, as well as the low-order SEMFEM preconditioner solved with AmgX, are performant on GPU machines such as OLCF's Summit. The authors propose a runtime auto-tuner preconditioner strategy that, while primitive, can choose reasonable solver parameters.

The authors plan on conducting similar parallel scalability studies on other machines, such as OLCF's Spock, an AMD MI100 machine with similar hardware and software as the upcoming Frontier system. Additional preconditioner options, such as varying the AMG solver settings for the coarse grid solve as well as the solver location (CPU/GPU), varying the Chebyshev order and number of sweeps on each level of the multigrid heirarchy, and varying the Chebyshev eigenvalue bounds, are avenues for future performance optimization. Local Fourier analysis similar to that done by Thompson *et al.* in [29] is further needed to understand the smoothing properties of the Chebyshev-accelerated Schwarz smoothers, allowing for robust optimization of parameters, as in the work by Brown *et al.* [4].

7 Acknowledgements

This research 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. This research also used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract DE-AC05-00OR22725.

The authors thank YuHsiang Lan, Ramesh Balakrishnan, David Alan Reger, and Haomin Yuan for providing visualizations and mesh files. The authors thank the reviewers of this work for their insightful comments and suggestions.

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, https://doi.org/10.1016/S0021-9991(03)00194-3, https://linkinghub.elsevier.com/retrieve/pii/S0021999103001943 (accessed 2021-09-09).

[2] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang, *Multigrid Smoothers for Ultraparallel Computing*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2864–2887, https://doi.org/10.1137/100798806, http://epubs.siam.org/doi/10.1137/100798806 (accessed 2020-09-05).

[3] P. D. Bello-Maldonado and P. F. Fischer, *Scalable Low-Order Finite Element Preconditioners for High-Order Spectral Element Poisson Solvers*, SIAM Journal on Scientific Computing, 41 (2019), pp. S2–S18, https://doi.org/10.1137/18M1194997, https://epubs.siam.org/doi/10.1137/18M1194997 (accessed 2021-07-20).

[4] J. Brown, Y. He, S. MacLachlan, M. Menickelly, and S. M. Wild, *Tuning Multigrid Methods with Robust Optimization and Local Fourier Analysis*, SIAM Journal on Scientific Computing, 43 (2021), pp. A109–A138, https://doi.org/10.1137/19M1308669, https://epubs.siam.org/doi/10.1137/19M1308669 (accessed 2021-10-21).

[5] C. Canuto, P. Gervasio, and A. Quarteroni, *Finite-Element Preconditioning of G-NI Spectral Methods*, SIAM Journal on Scientific Computing, 31 (2010), pp. 4422–4451, https://doi.org/10.1137/090746367, https://epubs.siam.org/doi/abs/10.1137/090746367 (accessed 2021-09-24). Publisher: Society for Industrial and Applied Mathematics.

[6] N. Chalmers, A. Karakus, A. P. Austin, K. Swirydowicz, and T. Warburton, *libParanumal: a performance portable high-order finite element library*, 2020, https://doi.org/10.5281/zenodo.4004744, https://github.com/paranumal/libparanumal. Release 0.4.0.

[7] X. chuan Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput, 21 (1999), pp. 792–797.

[8] M. O. Deville, P. F. Fischer, P. F. Fischer, and E. Mund, *High-order methods for incompressible fluid flow*, vol. 9, Cambridge university press, 2002.

[9] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, and T. Warburton, *NekRS, a GPU-Accelerated Spectral Element Navier-Stokes Solver*, arXiv:2104.05829 [cs], (2021), http://arxiv.org/abs/2104.05829 (accessed 2021-09-23). arXiv: 2104.05829.

[10] P. Fischer and J. Lottes, *Hybrid Schwarz-multigrid methods for the spectral element method: Extensions to Navier-Stokes*, in Domain Decomposition Methods in Science and Engineering Series, R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu, eds., Springer, Berlin, 2004.

[11] P. Fischer, E. Merzari, M. Min, S. Kerkemeier, Y.-H. Lan, M. Phillips, T. Rathnayake, A. Novak, D. Gaston, N. Chalmers, and T. Warburton, *Highly Optimized Full-Core Reactor Simulations on Summit*, arXiv:2110.01716 [physics], (2021), http://arxiv.org/abs/2110.01716 (accessed 2021-10-07). arXiv: 2110.01716.

[12] P. F. Fischer, *Projection techniques for iterative solution of Ax= b with successive right-hand sides*, Computer methods in applied mechanics and engineering, 163 (1998), pp. 193–204. Publisher: Elsevier.

[13] V. E. Henson and U. M. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Applied Numerical Mathematics, 41 (2002), pp. 155–177, https://doi.org/10.1016/S0168-9274(01)00115-5, https://www.sciencedirect.com/science/article/pii/S0168927401001155 (accessed 2021-09-23).

[14] A. Imakura, T. Sakurai, K. Sumiyoshi, and H. Matsufuru, *An Auto-Tuning Technique of the Weighted Jacobi-Type Iteration Used for Preconditioners of Krylov Subspace Methods*, in 2012 IEEE 6th International Symposium on Embedded Multicore SoCs, Sept. 2012, pp. 183–190, https://doi.org/10.1109/MCSoC.2012.29.

[15] D. S. Kershaw, *Differencing of the diffusion equation in Lagrangian hydrodynamic codes*, Journal of Computational Physics, 39 (1981), pp. 375–395. Publisher: Elsevier.

[16] T. Kolev, P. Fischer, A. P. Austin, A. T. Barker, N. Beams, J. Brown, J.-S. Camier, N. Chalmers, V. Dobrev, Y. Dudouit, L. Ghaffari, S. Kerkemeier, Y.-H. Lan, E. Merzari, M. Min, W. Pazner, T. Ratnayaka, M. S. Shephard, M. H. Siboni, C. W. Smith, J. L. Thompson, S. Tomov, and T. Warburton, *CEED ECP Milestone Report: High-order algorithmic developments and optimizations for large-scale GPU-accelerated simulations*, tech. report, Zenodo, Mar. 2021, https://doi.org/10.5281/zenodo.4672664, https://zenodo.org/record/4672664 (accessed 2021-09-27).

[17] M. Kronbichler and K. Ljungkvist, *Multigrid for matrix-free high-order finite element computations on graphics processors*, ACM Transactions on Parallel Computing, 6 (2019), pp. 1–32, https://doi.org/10.1145/3322813.

[18] Y.-H. Lan, P. Fischer, E. Merzari, and M. Min, *All-Hex Meshing Strategies For Densely Packed Spheres*, Proceedings of the 29th International Meshing Roundtable, (2021), pp. 293–305, https://doi.org/10.5281/zenodo.5551173.

[19] S. Loisel, R. Nabben, D. B. Szyld, J. Lottes, and P. Fischer, *On Hybrid Multigrid-Schwarz Algorithms*, J Sci Comput, (2008), p. 11.

[20] J. W. Lottes and P. F. Fischer, *Hybrid*

Multigrid/Schwarz Algorithms for the Spectral Element Method, Journal of Scientific Computing, 24 (2005), pp. 45–78, https://doi.org/10.1007/s10915-004-4787-3, http://link.springer.com/10.1007/s10915-004-4787-3 (accessed 2021-07-20).

[21] D. S. Medina, A. St-Cyr, and T. Warburton, *Occa: A unified approach to multi-threading languages*, arXiv preprint arXiv:1403.0968, (2014).

[22] K. Mittal and P. Fischer, *Mesh smoothing for the spectral element method*, Journal of Scientific Computing, 78 (2019), pp. 1152–1173.

[23] M. Naumov, M. Arsaev, P. Castonguay, J. Cohen, J. Demouth, J. Eaton, S. Layton, N. Markovskiy, I. Reguly, N. Sakharnykh, V. Sellappan, and R. Strzodka, *AmgX: A Library for GPU Accelerated Algebraic Multigrid and Preconditioned Iterative Methods*, SIAM Journal on Scientific Computing, 37 (2015), pp. S602–S626, https://doi.org/10.1137/140980260, https://epubs.siam.org/doi/abs/10.1137/140980260 (accessed 2021-09-23). Publisher: Society for Industrial and Applied Mathematics.

[24] L. Olson, *Algebraic Multigrid Preconditioning of High-Order Spectral Elements for Elliptic Problems on a Simplicial Mesh*, SIAM Journal on Scientific Computing, 29 (2007), pp. 2189–2209, https://doi.org/10.1137/060663465, https://epubs.siam.org/doi/abs/10.1137/060663465 (accessed 2021-09-24). Publisher: Society for Industrial and Applied Mathematics.

[25] S. A. Orszag, *Spectral Methods for Problems in Complex Geometrics*, in Numerical Methods for Partial Differential Equations, S. V. PARTER, ed., Academic Press, 1979, pp. 273–305, https://doi.org/https://doi.org/10.1016/B978-0-12-546050-7.50014-9, https://www.sciencedirect.com/science/article/pii/B9780125460507500149.

[26] M. L. Shur, P. R. Spalart, M. K. Strelets, and A. K. Travin, *Direct numerical simulation of the two-dimensional speed bump flow at increasing Reynolds numbers*, International Journal of Heat and Fluid Flow, 90 (2021), p. 108840. Publisher: Elsevier.

[27] J. Stiller, *Nonuniformly Weighted Schwarz Smoothers for Spectral Element Multigrid*, Journal of Scientific Computing, 72 (2017), pp. 81–96, https://doi.org/10.1007/s10915-016-0345-z, http://link.springer.com/10.1007/s10915-016-0345-z (accessed 2020-05-07).

[28] H. Sundar, G. Stadler, and G. Biros, *Comparison of multigrid algorithms for high-order continuous finite element discretizations*, Numerical Linear Algebra with Applications, 22 (2015), pp. 664–680, https://doi.org/10.1002/nla.1979, http://onlinelibrary.wiley.com/doi/abs/10.1002/nla.1979 (accessed 2020-09-05). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nla.1979.

[29] J. L. Thompson, J. Brown, and Y. He, *Local fourier analysis of p-multigrid for high-order finite element operators*, arXiv preprint arXiv:2108.01751, (2021).

[30] K. Yamada, T. Katagiri, H. Takizawa, K. Minami, M. Yokokawa, T. Nagai, and M. Ogino, *Preconditioner auto-tuning using deep learning for sparse iterative algorithms*, in 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW), IEEE, 2018, pp. 257–262.

[31] H. Yuan, M. A. Yildiz, E. Merzari, Y. Yu, A. Obabko, G. Botha, G. Busco, Y. A. Hassan, and D. T. Nguyen, *Spectral element applications in complex nuclear reactor geometries: Tet-to-hex meshing*, Nuclear Engineering and design, 357 (2020), p. 110422.

[32] V. T. Zhukov, N. D. Novikova, and O. B. Feodoritova, *Multigrid method for anisotropic diffusion equations based on adaptive Chebyshev smoothers*, Mathematical Models and Computer Simulations, 7 (2015), pp. 117–127, https://doi.org/10.1134/S2070048215020118, http://link.springer.com/10.1134/S2070048215020118 (accessed 2021-10-08).