OPTIMAL CHEBYSHEV SMOOTHERS AND ONE-SIDED V-CYCLES

MALACHI PHILLIPS† AND PAUL FISCHER‡

Abstract. The solution to the Poisson equation arising from the spectral element discretization of the incompressible Navier-Stokes equation requires robust preconditioning strategies. One such strategy is multigrid. To realize the potential of multigrid methods, effective smoothing strategies are needed. Chebyshev polynomial smoothing proves to be an effective smoother. However, there are several improvements to be made, especially at the cost of symmetry. For the same cost per iteration, a symmetric V-cycle with order Chebyshev polynomial smoothing may be substituted with a one-sided V-cycle with order \(2k\) Chebyshev polynomial smoothing, wherein the smoother is omitted on the up-leg of the V-cycle. The choice of omitting the post-smoother in favor of higher order Chebyshev pre-smoothing is shown to be advantageous in cases where the multigrid approximation property constant, \(C\), is large. Results utilizing Lottes’s fourth-kind Chebyshev polynomial smoother are shown. These methods demonstrate substantial improvement over the standard Chebyshev polynomial smoother. The authors demonstrate the effectiveness of this scheme in \(p\)-geometric multigrid, as well as a 2D model problem with finite differences.

Key words. multigrid, smoothers, parallel computing

MSC codes. 76D05, 65N55, 65F08

1. Introduction. Chebyshev smoothing was introduced in the context of parallel multigrid (MG) methods in [1], where it was established that Chebyshev smoothing was competitive with Gauss-Seidel smoothing even in serial computing applications. Here, we explore several variations on Chebyshev smoothing for the Poisson problem in general domains. Our primary aim is to develop fast highly-scalable solvers for the pressure sub-step in time advancement of the Navier-Stokes (NS) equations, particularly for discretizations based on the spectral element method (SEM). Many of the finding, however, would apply in more general settings.

Our target problem is to solve a sequence of Poisson problems,

\[
-\nabla^2 \tilde{u} = \tilde{f} \quad \text{for} \quad \tilde{u}, \tilde{f} \in \Omega \subset \mathbb{R}^d \rightarrow \mathbb{R}.
\]

The weak formulation is written as: find \(u^m(x) \in X^N_0 \subset \mathcal{H}^1_0\) such that

\[
\int_\Omega \nabla v \cdot \nabla u \, dV = \int_\Omega v \, f^m \, dV \quad \forall \, v \in X^N_0,
\]

where \(f^m(x)\) is the data and \(u^m(x)\) is the corresponding solution field at some time instant \(t^m, m = 1, 2, \ldots\). Here, \(\Omega \subset \mathbb{R}^d\) is the computational domain in \(d\) (=1, 2, or 3) space dimensions; \(\mathcal{H}^1_0(\Omega)\) is the standard Sobolev space comprising functions that vanish on a subset of the boundary, \(\partial \Omega_D \subset \partial \Omega\), are square-integrable on \(\Omega\), and whose gradient is also square-integrable; and \(X^N_0 = \text{span}\{\phi_j(x)\}\) is the finite-dimensional trial/test space associated with a Galerkin formulation of the Poisson problem. The discrete problem statement is expressed as \(A u^m = b^m\), where \(u^m\) is the vector of basis coefficients at \(t^m\) and \(A\) is the symmetric-positive definite (SPD) matrix with

\[
a_{ij} = \int_\Omega \nabla \phi_i \cdot \nabla \phi_j \, dV.
\]

---

*This work was funded by the Exascale Computing Project under contract no. 17-SC-20-SC
†Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana IL 61801 (malachi2@illinois.edu).
‡Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign, Urbana IL 61801 (fischerp@illinois.edu).
We point out that the need to solve a sequence of problems differs from solving a single problem in several significant ways. First, solver set-up costs are typically amortized over thousands of right-hand sides and are therefore largely irrelevant to our cost concerns. Second, the solution is typically devoid of significant low wave-number content because we solve only for a perturbed solution, \( \delta u^m := u^m - \bar{u} \), where \( \bar{u} \) is an initial guess. If we take \( \bar{u} = u^{m-1} \) then the initial residual \( r_0 = b - Au^{m-1} = O(\Delta t) \). This result is improved to \( O(\Delta t) \) by projecting \( u^m \) onto the space of prior solutions, \( \{u^{m-1}, \ldots, u^{m-l}\} \) \([8, 16]\). Finally, with an initially small residual, GMRES is likely to converge in just a few iterations, which obviates the need for restarts and mitigates the \( O(k^2) \) complexity terms in a \( k \)-iteration GMRES solve. This latter observation puts less pressure on requiring a symmetric preconditioner since one can retain the full benefits of using Krylov subspace projection (KSP) without resorting to conjugate gradient iteration. With these circumstances in mind, we will drop the superscript \( m \) in the sequel.

We note that Chebyshev smoothers have gained a lot of attention recently. Kronbichler and co-workers \([20, 12, 11]\) have employed Chebyshev smoothing for discontinuous Galerkin discretizations of the NS equations. Rudi and coworkers employ algebraic multigrid (AMG) with Chebyshev smoothing \([39]\). Similarly, Chebyshev smoothing is considered by Sundar and coworkers as a multigrid smoother for high-order continuous finite element discretizations \([43]\).

A major difference here is that we consider Chebyshev in conjunction with additive Schwarz methods (ASM) \([45, 13, 24, 35]\) and restrictive additive Schwarz (RAS) \([5]\) in place of point-Jacobi smoothing. The principal idea is to use ASM or RAS to eliminate high wave number content. In the case of the spectral element method, local Schwarz solves can be effected at a cost that is comparable to forward operator evaluation through the use of fast diagonalization \([25, 15, 24]\). Another critical aspect of the current context is that many of our applications are targeting exascale platforms and beyond, where compute is performed on tens of thousands of GPUs for which the relative cost of global communication and hence, coarse-grid solves, is high \([31]\). In such cases, it often pays to have high-quality and broad bandwidth smoothing, such as provided by Chebyshev, in order to reduce the number of visits to the bottom of the V-cycle where the expensive coarse-grid solve is invoked.

Here, we explore a seemingly simple question: Given \( 2k \) smoothing iterations, what is the optimal choice of \( m \) pre-smoothing and \( n \) post-smoothing applications, where \( m + n = 2k \)? More specifically, in the Chebyshev context, the question is what order \( m \) pre-smoothing and order \( n \) post-smoothing should be used at the same cost per iteration. An additional and important point to this question is, What kind of Chebyshev smoothing should be used? One could use standard 1st-kind Chebyshev polynomials with tuned parameters. (Recall, we can afford significant tuning overhead.) Or, one could use standard or optimized 4th-kind Chebyshev polynomials that were proposed in recent work by Lottes. (See \([23]\) and references therein.) We explore these questions under several different conditions: using finite differences and spectral elements discretizations and using Jacobi, ASM, or RAS as the basic smoother.

The structure of this paper is as follows. Section 2 outlines the multigrid V-cycle and Chebyshev smoothers. 2D finite difference Poisson results on varying aspect ratio grids, along with a comparison between theoretical and observed multigrid error contraction rates, are presented in section 3. Spectral element (SE)-based pressure Poisson preconditioning schemes implemented in the scalable open-source CFD code, nekRS \([14]\), are presented in section 4. nekRS started as a fork of libParanumal \([7]\) and uses highly optimized kernels based on the Open Concurrent Compute Abstrac-
Following [40, 1], we wish to solve:

\[
A_{j+1} = \left( P^j_{j+1} \right)^T A_j P^j_{j+1}, j = 0, \ldots, \ell - 1.
\]

The multiplicative error propagator for a single V-cycle is given recursively by

\[
E_j = I - M_j^{-1} A_j
\]

\[
= G_j' \left( I - P^j_{j+1} M_{j+1}^{-1} \left( P^j_{j+1} \right)^T A_j \right) G_j, j = 0, \ldots, \ell - 1,
\]

with \( M_j^{-1} := A_{j+1}^{-1} \). \( G_j \) and \( G_j' \) are the smoother iteration matrices for the pre- and post-smoothing iteration matrices, respectively. For example, \( G_j = (I - \omega S_j A_j)^k \) corresponds to \( k \) steps of the simple smoothing iteration

\[
(\tilde{x}_{j+1})_j = (\tilde{x}_j)_j + \omega S_j (b_j - A_j (\tilde{x}_j)_j),
\]

where \( S_j \) is the smoother for the \( j \)th level, such as Jacobi with \( S_j = \text{diag}(A_j)^{-1} A_j \). \( G_j' \) and \( G_j \) are not necessarily the same – in fact, we will consider the choice of \( G_j' = G_j \) (symmetric post-smoothing) as well as \( G_j' = I \) (omitting post-smoothing), among others.

For later use, we will need to consider the one-sided V-cycle, which is sufficient for the analysis of general V-cycles [28, 23]. Similar as before, the “fine-to-coarse”

\[
(E_{\downarrow})_j = I - (M_{\downarrow})_j^{-1} A_j
\]

\[
= \left( I - P^j_{j+1} (M_{\downarrow})^{-1}_{j+1} \left( P^j_{j+1} \right)^T A_j \right) G_j, j = 0, \ldots, \ell - 1,
\]

and “coarse-to-fine”

\[
(E_{\uparrow})_j = I - (M_{\uparrow})_j^{-1} A_j
\]

\[
= G_j' \left( I - P^j_{j+1} (M_{\uparrow})^{-1}_{j+1} \left( P^j_{j+1} \right)^T A_j \right), j = 0, \ldots, \ell - 1,
\]

error propagators are defined, with \( (M_{\downarrow})_\ell^{-1} = (M_{\uparrow})_{\ell+1}^{-1} = A_\ell^{-1} \). Let \( E_{\downarrow} = (E_{\downarrow})_0 \) and \( E_{\uparrow} = (E_{\uparrow})_0 \). The general V-cycle, therefore, is the product of the “fine-to-coarse” and “coarse-to-fine” error propagators,

\[
E_V = E_{\uparrow} E_{\downarrow}.
\]

Given \( k_j \) iterations of the smoothing iteration in (2.3) for level \( j \), is it possible to construct a better order \( k_j \) polynomial than \( p_{k_j} (S_j A_j) = G_{k_j} = (I - \omega S_j A_j)^{k_j} \)? Following [40, 1], we wish to solve:

\[
\min_{p_k \in P_k, p_k(0) = 1} \lambda \in [\lambda_{\min}, \lambda_{\max}] \max |p(t)|.
\]
where \( \mathbb{P}_k \) is the space of all polynomials with degree less than or equal to \( k \). Taking \( \mathcal{E} \) to be the interval \([\lambda_{\text{min}}, \lambda_{\text{max}}]\), the solution to the minimax problem in (2.7) are the shifted and scaled Chebyshev polynomials of the 1st-kind

\[
T_k(\lambda) = \frac{1}{\sigma_k} T_k \left( \frac{\theta - \lambda}{\delta} \right) \quad \text{with} \quad \sigma_k := T_k \left( \frac{\theta}{\delta} \right).
\]

\( T_k(\cdot) \) is the Chebyshev polynomial of the 1st-kind of order \( k \); \( \theta \) is the midpoint of the interval \([\lambda_{\text{min}}, \lambda_{\text{max}}]\),

\[
\theta = \frac{\lambda_{\text{min}} + \lambda_{\text{max}}}{2};
\]

and \( \delta \) is the mid-width of the interval,

\[
\delta = \frac{\lambda_{\text{max}} - \lambda_{\text{min}}}{2}.
\]

The Chebyshev polynomials of the 1st-kind enjoy a three-term recurrence relation that is used to derive Algorithm 2.1 [40]. While \( \lambda_{\text{max}} \) is taken to be the largest eigenvalue of \( S_j A_j \), how should \( \lambda_{\text{min}} \) be chosen? \( \lambda_{\text{min}} \) is chosen as a small factor of \( \lambda_{\text{max}} \). Previous works considered factors such as 1/30 [1], 3/10 [2], 1/4 [43], and 1/6 [47]. Two approaches utilizing the 1st-kind Chebyshev smoother are considered in this study. The first uses \( \lambda_{\text{min}} = 0 \).1 \( \lambda_{\text{max}} \) and is denoted as 1st-Cheb. The second optimizes the for \( \lambda_{\text{min}} \) and is denoted as 1st-Cheb, \( \lambda_{\text{opt}} \).

**Algorithm 2.1** Chebyshev smoother, 1st-kind

\[
\theta = \frac{1}{2}(\lambda_{\text{max}} + \lambda_{\text{min}}), \quad \delta = \frac{1}{2}(\lambda_{\text{max}} - \lambda_{\text{min}}), \quad \sigma = \frac{\theta}{\delta}, \quad \rho_0 = \frac{1}{\sigma}
\]

\[\begin{align*}
\underline{z}_0 &= \underline{z}_0, \quad \underline{r}_0 = S(b - A\underline{z}_0), \quad \underline{d}_0 = \frac{1}{\delta} \underline{z}_0 \\
\text{for} \ i = 1, \ldots, k - 1 \text{ do} & \\
\underline{z}_i &= \underline{z}_{i-1} + \underline{d}_{i-1} \\
\underline{z}_i &= \underline{z}_{i-1} - S A \underline{d}_{i-1}, \quad \rho_i = \frac{1}{2\sigma - \rho_{i-1}} \\
\underline{d}_i &= \rho_i \underline{d}_{i-1} + \frac{2\rho_i}{\delta} \underline{z}_i \\
\text{end for} & \\
\underline{z}_k &= \underline{z}_{k-1} + \underline{d}_{k-1} \\
\text{return} & \underline{z}_k
\end{align*}\]

Is it possible to further improve the polynomial smoother and remove the ad-hoc \( \lambda_{\text{min}} \) parameter? The polynomial smoother can be chosen in such a way to minimize an error bound [23], such as the two-level bound proposed by Hackbusch [17] in (2.9). Without loss of generality, let \( \rho(SA) = 1 \). Let \( G = G_k(SA) \) be a \( k \)-order polynomial in \( SA \). The two-level bound from Hackbusch [17] is re-written as [23]:

\[
\|E\|_A = \|(I - PA_s^{-1}P^T A)G_k\|_A
\]

\[
\leq C_0^{1/2} \sup_{0 < \lambda \leq 1} \lambda^{1/2}|p_k(\lambda)|,
\]

where \( C_0 \) is the multigrid approximation property constant, which for a given level \( j \)}
is

\[ C_j := \left\| A_j^{-1} - P_j^j A_j^{-1} \left( P_j^j \right)^T \right\|_{A_j, S_j}^2 \]

\[ := \sup \left\{ \left\| A_j^{-1} - P_j^j A_j^{-1} \left( P_j^j \right)^T \right\|_{A_j}^2 \right\} \]

(2.10)

Solving the weighted minimax problem in (2.9) yields the solution [23]:

\[ p_k(\lambda) = \frac{1}{2k + 1} W_k(1 - 2\lambda), \]

where \( W_k \) is the Chebyshev polynomial of the 4th-kind of order \( k \). Chebyshev polynomials of the 4th-kind satisfy the same recurrence as that of the first-kind, with different initial conditions [23, 27]:

(2.12) \[ W_n(x) = 2xW_n(x) - W_{n-2}(x) \] with \( W_0(x) = 1 \), \( W_1(x) = 2x + 1 \).

Following this recurrence and relaxing the \( \rho(SA) = 1 \) assumption, a similar iterative algorithm to Algorithm 2.1 can be derived as Algorithm 2.2. The inclusion of the \( \beta_i \) parameters in Algorithm 2.2 comes as a result of optimizing (2.14) in the context of the multi-level error bound from Lemma 2.1 [23]. For the standard 4th-kind Chebyshev polynomial, \( \beta_i := 1 \) for all \( i \). The simple smoothing iteration with \( \omega = 3/2 \), 1st-kind Chebyshev smoothing with \( \lambda_{\min} = 0.3\lambda_{\max} \), and the 4th-kind Chebyshev smoothing polynomials, with and without optimal \( \beta_i \), are shown in Figure 1. Minimizing (2.14) with respect to \( \lambda_{\min} \) in the 1st-kind Chebyshev smoothing iteration is also shown.

Algorithm 2.2 Chebyshev smoother, (Opt.) 4th-kind

\[ x_0 = x, \quad r_0 = b - Ax_0 \]

\[ d_0 = \frac{4}{3\lambda_{\max}} Sx_0 \]

for \( i = 1, \ldots, k-1 \) do

\[ x_i = x_{i-1} + \beta d_{i-1}, \quad L_i = L_{i-1} - Ad_{i-1} \]

\[ d_i = \frac{2i-1}{2i+3} d_{i-1} + \frac{8i+4}{2i+3} \lambda_{\max} Sx_i \]

end for

\[ x_k = x_{k-1} + \beta_k d_{k-1} \]

return \( x_k \)

Given \( 2k \) smoothing steps, what is the optimal way to distribute the number of smoothing steps for the pre-smoother, \( m \), and the post-smoother, \( n \), with \( m + n = 2k \)? The multi-level error bound in Lemma 2.1, developed by Lottes in [23], provides a theoretical framework to answer the question.

**Lemma 2.1.** Let the smoother iteration (on each level \( j \)) be given by

\[ G_j = p_{k_j}(S_j A_j) \]

where \( S_j \) is SPD and scaled such that \( \rho(S_j A_j) = 1 \), and \( p_{k_j}(x) \) is a \( k_j \)-order polynomial satisfying \( p_{k_j}(0) = 1 \) and \( |p_{k_j}(x)| < 1 \) for \( 0 < x \leq 1 \), possibly different on each level.

\[ ^1 \text{Tabulated } \beta_i \text{ coefficients for the 4th-kind Chebyshev polynomials are available in Table 5.} \]
Then the V-cycle contraction factor

\[(2.13) \quad \|E_j\|_A^2 \leq \max_{j=0,\ldots,t-1} \frac{C_j}{C_j + \gamma_j^{-1}} \]

where $C_j$ is the approximation property constant for level $j$, defined in (2.10), and

\[(2.14) \quad \gamma_j = \sup_{0<\lambda \leq 1} \frac{\lambda p_k(\lambda)^2}{1 - p_k(\lambda)^2}. \]

Let us restrict the order considered in Lemma 2.1 to the case $k_j = k$ for all
\( j \in 0, \ldots, \ell - 1 \), then \( \gamma_j = \gamma \) is constant across all levels. Further, let us define
\[
(2.15) \quad C := \max_{j \in 0, \ldots, \ell - 1} C_j.
\]

Lemma 2.1 simplifies to
\[
(2.16) \quad \|E_\gamma\|_A^2 \leq \frac{C}{C + \gamma^{-1}(k)} := V(C, k).
\]

Rather than directly considering the effect of the choice of \( m \) and \( n \) on the multi-grid error contraction factor, we instead consider the effect on the error contraction bound presented in (2.16). The problem of interest, therefore, is to find \( m, n \)
\[
(2.17) \quad m^*, n^* := \arg \min_{m, n, m+n=2k} V(C, m) \cdot V(C, n)
\]
\[
= \arg \max_{m, n, m+n=2k} C \left( \gamma^{-1}(m) + \gamma^{-1}(n) \right) + \gamma^{-1}(m) \cdot \gamma^{-1}(n),
\]
where \( \gamma^{-1}(k) \) is the inverse of \( \gamma \), as defined in (2.14), for a \( k \)-order polynomial.

Lottes [23] notes that, for the simple smoother iteration with weight \( \omega \),
\[
(2.18) \quad \gamma_s^{-1}(k) = 2\omega k.
\]
The 4th-kind Chebyshev polynomial, however, has
\[
(2.19) \quad \gamma_4^{-1}(k) = \frac{4}{3} k(k+1),
\]
while the optimized 4th-kind Chebyshev polynomial is
\[
(2.20) \quad \gamma_4^{opt}(k) = \frac{4}{\pi^2}(2k+1)^2 - \frac{2}{3};
\]
Most notably, both 4th-kind Chebyshev polynomials types improve the \( O(k) \) simple smoother iteration to \( O(k^2) \) as \( k \to \infty \). To apply Lemma 2.1 to the 1th-kind Chebyshev polynomial, without loss of generality, let us assume that \( \lambda_{max} = 1 \). For the 1st-kind Chebyshev polynomials with fixed \( \lambda_{min} \),
\[
(2.21) \quad \gamma_1^{-1}(k) = \begin{cases} 
T_k \left( \frac{\lambda_{min}+1}{\lambda_{min}-1} \right)^2 - 1 & \text{if } k \leq k^* \\
\frac{4kU_{k-1} \left( \frac{\lambda_{min}+1}{\lambda_{min}-1} \right)}{(\lambda_{min}-1)T_k \left( \frac{\lambda_{min}+1}{\lambda_{min}-1} \right)} & \text{if } k > k^*
\end{cases},
\]
where \( T_\xi \) and \( U_\xi \) are the \( \xi \)th-order Chebyshev polynomials of the first and second kinds, respectively. For \( \lambda_{min} = 0.1 \), \( k^* = 3 \). As \( k \to \infty \), (2.21) scales as
\[
(2.22) \quad \gamma_1^{-1}(k) \sim \sqrt{\frac{1}{\lambda_{min}}} k.
\]
However, by choosing \( \lambda_{min} \) such that \( \gamma^{-1} \) is maximized,
\[
(2.23) \quad \gamma_1^{opt}(k) \sim 2.38k^{1.73}
\]
for large $k$. As an aside, a correlation for $\lambda_{\min}^*$ with 1% relative error and 0.1% absolute error for $k \in [1, 50]$ is given by

$$(2.24) \quad \lambda_{\min}^* \approx \frac{1.69}{k^{1.68} + 2.11k + 1.98}.$$ 

Due to symmetry of (2.17), the error bound for $(m, n)$ is the same as that of $(n, m)$. To start to assess (2.17), we first consider the case with $m = n = k$ compared to $m = 2k, n = 0$. For the simple smoother with fixed $\omega$, the error-bound is minimized with $m = n = k$ if and only if

$$(2.25) \quad C > \frac{2 (k + 1)^2}{3}.$$ 

The optimized 4th-kind Chebyshev polynomial has a similar condition, provided

$$(2.26) \quad C > \frac{2 \left(6 (2k + 1)^2 - \pi^2 \right)^2}{3 \pi^2 \left(-12 (2k + 1)^2 + 6 (4k + 1)^2 + \pi^2 \right)}.$$ 

For the 1st-kind Chebyshev polynomial with fixed $\lambda_{\min}$,

$$(2.27) \quad C \gtrsim 1.55 e^{1.45k}$$ 

as $k \to \infty$. However, for $k \in [1, 3], m = 2k, n = 0$ outperforms $m = n = k$, irrespective of $C$. Lastly, for the 1st-kind Chebyshev polynomial with $\lambda_{\min}$ chosen to maximize $\gamma^{-1}$,

$$(2.28) \quad C \gtrsim 1.81 k^{1.73}.$$ 

With the regions where $m = 2k, n = 0$ outperforms $m = n = k$ determined, the authors wish to solve the more general optimization problem in (2.17). In lieu of a general solution, however, the authors opt to prove that the optimal $m$ and $n$ occurs at either $m = 2k, n = 0$ or $m = n = k$ for all $C > 0$. To do so, the authors utilized sympy [30] to solve for the region where a tentative $(\tilde{m}, \tilde{n})$ outperforms $m = 2k, n = 0$ and the region where $(\tilde{m}, \tilde{n})$ outperforms $m = n = k$. The intersection of these two regions, therefore, is the region where $(\tilde{m}, \tilde{n})$ outperforms both $m = 2k, n = 0$ and $m = n = k$. With exception to (4.2) and (5.1) for $C < 4$ in the case of 1st-kind Chebyshev polynomial smoothing with $\lambda_{\min} = 0.1$, the authors found that the optimal $(m, n)$ is either $(m = 2k, n = 0)$ or $(m = n = k)$ for all $C > 0$ for the smoothers considered, up to $k = 50$.

A natural next question is on the expected improvement using the $(2k, 0)$ scheme over the symmetric $(k, k)$ scheme. This is done by taking the ratio of the error bound (2.17) with $(k, k)$ and $(2k, 0)$ for the various polynomial smoothers considered. For the 1st-kind Chebyshev polynomial with $\lambda_{\min} = 0.1$, the expected improvement in the multigrid convergence is no more than 6% for $k = 1$, and quickly decreases for larger $k$. Optimizing the bound with respect to $\lambda_{\min}$, however, the 1st-kind Chebyshev polynomial with $(2k, 0)$ can outperform the symmetric $(k, k)$ scheme by 9%. For both the 4th-kind and optimized 4th-kind Chebyshev polynomials, however, the $(2k, 0)$ scheme outperforms the symmetric $(k, k)$ scheme by 15% as $k \to \infty$. A numerical example demonstrating the applicability of this analysis based on Lemma 2.1 and (2.17) is given in section 3.
3. Finite Differences with Geometric Multigrid Poisson. The Poisson equation (1.1) is considered with \( d = 2 \). Let \( \Omega := [0, L_x] \times [0, L_y] \) be the domain of interest, with the boundary condition \( u|_{\partial \Omega} = 0 \). A finite difference grid of \((n+1) \times (n+1)\) points is considered, \( n = 128 \). For the purposes of this study, \( u(x, y) = \sin(3\pi x/L_x) \sin(4\pi y/L_y) + g \), where \( g \) is the same random vector with \( g|_{\partial \Omega} = 0 \). \( L_x \geq 1 \) is varied, while \( L_y \) is fixed at unity. A geometric multigrid V-cycle with Chebyshev-accelerated Jacobi smoothing is employed as a preconditioner for GMRES(20). For the comparison with the bounds presented in Lemma 2.1, multigrid is also considered as a solver. Each coarser level is discretized as \((n_c + 1) \times (n_c + 1)\), with \( n_c = n/2 \), as well as aggressive coarsening with \( n_c = n/8 \). This is repeated until there is only a single degree of freedom. As this case is meant to represent our target problem for unstructured multigrid, semi-coarsening is not employed. On the coarsest level, the single degree of freedom system is solved exactly. We choose a relative residual reduction of \( 10^{-6} \) as the stopping criterion. A variety of smoothing orders are considered for all orders \( m \) pre-smoothing and orders \( n \) post-smoothing, with \( m + n = 2k \) up to \( k = 10 \).

Convergence results are based on the observed average solver convergence rate

\[
\rho = \exp \left( \frac{1}{N} \log \frac{\|r_N\|}{\|r_0\|} \right),
\]

where \( N \) is the number of iterations. The first question, as posed in the optimization problem (2.17), is regarding the optimal choice of V-cycle smoothing \((m, n)\) with \( m + n = 2k \). Recall, however, that the authors demonstrated the optimality of either the one-sided \((2k, 0)\) or symmetric \((k, k)\) V-cycles for all orders up to \( k = 50 \), with a few exceptions for small values of \( C \). The authors observe that, as predicted in (2.16) and (2.17), the convergence rate of \((m, n)\) is nearly equivalent to \((n, m)\). Further, with few exceptions, either the one-sided \((2k, 0)\) or symmetric \((k, k)\) V-cycles yielded the smallest convergence rate, as defined in (3.1).

Fig. 2. Critical \( C^* \) at which \((2k, 0)\) converges faster than \((k, k)\) for \( C > C^* \).
With our consideration now on the use of the symmetric \((k, k)\) or one-sided \((2k, 0)\) V-cycles, we can now confirm our theoretical prediction of when to apply these methods, as outlined in (2.25)–(2.28). The convergence rate for the symmetric \((k, k)\) V-cycle is denoted \(\rho V(C, k)\), while the one-sided \((2k, 0)\) V-cycle is \(\rho \downarrow (C, 2k)\). Values of \((C, k)\) at which \(\rho V(C, k) \geq \rho \downarrow (C, 2k)\) are shown in Figure 2. Predicted results are from (2.25)–(2.28), which are only applicable to multigrid as a solver, not preconditioner. Observed results are from 2D finite difference example. At a given \(k\), \(C > C^*\) indicates that the one-sided \((2k, 0)\) V-cycle yields a lower error bound than the symmetric \((k, k)\) V-cycle. Conversely, \(C < C^*\) indicates that the symmetric V-cycle yields better convergence. For the 1st-kind with optimized \(\lambda_{\text{min}}\) coefficient, 4th-kind, and optimized 4th-kind Chebyshev smoothers, the predicted domain in which to apply the one-sided \((2k, 0)\) V-cycle over the symmetric \((k, k)\) V-cycle shows agreement with the results obtained by experiment using multigrid as a solver (Figure 2a). As noted by (2.27), when \(k > 3\), the 1st-kind Chebyshev smoother does not benefit from applying the one-sided \((2k, 0)\) V-cycle over the symmetric \((k, k)\) V-cycle. At the same time, however, for \(k \leq 3\), the one-sided \((2k, 0)\) V-cycle outperforms the symmetric \((k, k)\) V-cycle, irrespective of \(C\). Despite the applicability of Lemma 2.1 being limited to multigrid as a solver, the predicted domain in which to apply the one-sided \((2k, 0)\) V-cycle as a preconditioner is similar to that of using multigrid as a solver (Figure 2b).

The maximum and minimum ratio of the error contraction rates, along with the predicted performance, are shown in Figure 3. While the predicted performance benefit of applying the one-sided V-cycle approach is limited, nevertheless, Figure 3a and Figure 3b demonstrate that the one-sided V-cycle offers an improvement compared to the symmetric V-cycle for problems with moderate values of \(C\). However, when applied to relatively easy problems (\(C \approx 1\)), the one-sided V-cycle is a poor choice.

Results for \(n_c = n/2\) are shown in Figure 4, \(n_c = n/8\) in Figure 5. In both, multigrid is used to precondition the GMRES(20) solver. In the case \(n_c = n/2\) and
OPT. CHEBY. SM. AND V-CYCLE

$L_x = 1$ ($C \approx 4$), shown in Figure 4a,d, the work required, as measured by fine-grid matrix vector products, is minimized for relatively low orders. Further, applying the bounds shown in Figure 2a, this is the scenario in which the one-sided V-cycle approach should not be utilized. However, even at moderate grid aspect ratios, such as $L_x = 8$, $C \approx 132$ becomes large enough to justify the usage of the one-sided V-cycle approach, especially for the 4th-kind Chebyshev smoothers (Figure 4b,c) at moderate orders $k = 6, \tilde{k} = 12$. This effect becomes even more apparent when $L_x = 64$ ($C \approx 3668$), shown in Figure 4c,f. In the results for $n_c = n/2$, the 1st-kind with optimized $\lambda_{\text{min}}$, 4th-kind, and optimized 4th-kind Chebyshev smoothers greatly outperform the 1st-kind Chebyshev smoother, especially at high orders. Further, it is observed that both the iteration count and total work in matrix-vector products is minimized at high orders for the 1st-kind with optimized $\lambda_{\text{min}}$, 4th-kind, and optimized 4th-kind Chebyshev smoothers. This is not the case for the 1st-kind Chebyshev smoother, however. This effect of lowering both the iteration count and work has the additional benefit of reducing the number of coarse grid solves required for each iteration, whose cost is not factored in this analysis.

In the aggressive coarsening case ($n_c = n/8$), both the number of matrix-vector products and iterations are reduced through using higher-order Chebyshev smoothing irrespective of the grid aspect ratio for all but the standard 1st-kind Chebyshev smoothers. Secondly, the one-sided $(2k, 0)$ V-cycle approaches generally outperforms the full, symmetric $(k, k)$ V-cycle approach. There are two important implications from this result: first, there exists considerable benefit in transitioning a multigrid
M. PHILLIPS AND P. FISCHER

Fig. 5. **FD, $n_c = n/8$. Multigrid as preconditioner for GMRES(20).**

solver to either construct high-quality estimates for $\lambda_{\text{min}}^{\text{opt}}$ for the 1st-kind Chebyshev smoother, such as those provided in correlation (2.24), or utilize one of the 4th-kind Chebyshev smoothers; and second, additional performance can be achieved by using the one-sided $(2k,0)$ V-cycle approach at the expense of symmetry, the implication of which is further discussed in subsection 6.1.

Let us also consider the trade-offs associated with using $n_c = n/2$ and $n_c = n/8$ as coarsening strategies. Despite improvements in the convergence rate from using a different V-cycle approach or Chebyshev smoother, the number of matrix-vector products required by the solve is greater for the aggressive coarsening case. However, the grid complexity

$$
\sum_{j=0}^{\ell} \text{nnz}(A_j) / \text{nnz}(A_0)
$$

for $n_c = n/2$ is 1.568, while for $n_c = n/8$ it is only 1.023. Therefore, the amount of work done per cycle is less for the aggressive coarsening case. An additional benefit of the aggressive coarsening strategy is decreasing the number of multigrid levels from 7 with $n_c = n/2$ to 3 with $n_c = n/8$. This feature is especially attractive in a parallel computing context where each additional multigrid level requires additional communication cost. Significant effort has been spent on improving the parallel scalability of multigrid methods, especially in the AMG context, see [42, 10, 4]. One strategy to achieve better scalability is to rely on aggressive coarsening strategies, which require more robust smoothers, such as the Chebyshev smoothers discussed here.
Table 1

| $L_x$ | Coarsening | Complexity | Solver Configuration | Mat-Vec | Iterations |
|-------|------------|------------|----------------------|---------|------------|
| 1     | 2          | 1.568      | $4^{th}$-Cheb, Jacobi(2, 2) | 20      | 4          |
| 8     | 2          | 1.568      | $4^{th}$-Cheb, Jacobi(14, 0) | 75      | 5          |
| 64    | 2          | 1.568      | $4_{opt}^{th}$-Cheb, Jacobi(20, 0) | 252     | 12         |
| 128   | 2          | 1.568      | $4_{opt}^{th}$-Cheb, Jacobi(20, 0) | 252     | 12         |

The results shown in Figure 4 and Figure 5 are summarized in Table 1. The solver configuration yielding the lowest number of matrix-vector products is listed for each case. $4_{opt}^{th}$-Cheb, Jacobi(20,0), for example, denotes a 20th order Chebyshev smoother of the optimized 4th-kind, using one-sided smoothing (thereby, having the same cost per iteration as order 9 with the symmetric V-cycle). This solver configuration yields the lowest number of matrix-vector products for $L_x = 128$ with aggressive coarsening. We see that, for high aspect ratio grids and aggressive coarsening, the one-sided $(2k, 0)$ V-cycle offers superior performance to the symmetric $(k, k)$ V-cycle approach.

4. High Order Preconditioners. Let us now consider preconditioners for the Poisson equation (1.1) for $d = 3$ arising from the spectral element method (SEM) discretization of the incompressible NS equation, which typically encompasses the majority of the solution time in nekRS [14]. Two classes of preconditioners prove most effective: $p$-geometric multigrid (pMG) and low-order discretizations.

4.1. $p$-Geometric Multigrid. $p$-geometric multigrid (pMG) is used as a preconditioner for the Pressure poisson equation discretized using the spectral element method. Since the multigrid hierarchy is constructed by varying the polynomial order, $p$, of each level, geometric coarsening can be robustly applied up to $p = 1$, even in the unstructured case. Typical multigrid schedules for $p = 7$, for example, are V-cycles with orders $(7, 5, 3, 1)$ or $(7, 3, 1)$. In addition to the Chebyshev-accelerated Jacobi smoother, Chebyshev-accelerated Schwarz smoothers are also considered, as discussed in [35]. The Schwarz smoothers are outlined below.

The SE-based additive Schwarz method (ASM) presented in [24, 22] solves local Poisson problems on subdomains that are extensions of the spectral elements. The formal definition of the ASM preconditioner (or, in this case, pMG smoother) is

$$S_{ASM,E} = \sum_{e=1}^{E} W_e R_e^T \tilde{A}_e^{-1} R_e E,$$

where $R_e$ is the restriction matrix that extracts nodal values of the residual vector that correspond to each overlapping domain. 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 $W_e^2$ [41, 24].
In a standard Galerkin ASM formulation, one would use $A_e = R_e A R_e^T$, but such an approach would compromise the $O(p^3)$ storage complexity of the SE method. To construct fast inverses for $A_e$, we approximate each deformed element as a simple box-like geometry. 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. 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 $O(E p^4)$ time in $\mathbb{R}^3$. The extended-box Poisson operator is

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

where each $B_*, A_*$ represents the extended 1D mass-stiffness matrix pairs along the given dimension \([9]\). The FDM begins with a preprocessing step of solving a series of small, $\bar{p} \times \bar{p}$, generalized eigenvalue problems,

$$A_* s_i = \lambda_i B_* s_i$$

and defining $S_* = (s_1 \ldots s_{\bar{p}})$ and $\Lambda_* = \text{diag}(\lambda_i)$, to yield the similarity transforms

$$S_*^T A_* S_* = \Lambda_*,$$

$$S_*^T B_* S_* = I.$$

From these, the inverse of the local Schwarz operator is

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

where $D$ is a diagonal matrix defined as

$$D := I \otimes I \otimes \Lambda_x + I \otimes \Lambda_y \otimes I + \Lambda_z \otimes I \otimes I.$$

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 [46] is used to solve the system on the CPU with the same parameters described in subsection 4.2, except using Chebyshev smoothing. A single BoomerAMG V-cycle iteration is used in the coarse-grid solve.

Presently, we also consider a restrictive additive Schwarz (RAS) version of (4.1), wherein overlapping values are not added after the action of the local FDM solve, following [5]. RAS has the added benefit of reducing the amount of communication required in the smoother. Similar to ASM, RAS is non-symmetric. Attempts to symmetrize the operator tend to have a negative impact on the convergence rate [5]. The smoother $S_{ASM}$ or $S_{RAS}$ described here is then used as the smoother considered in Chebyshev-acceleration, as shown in Algorithm 2.2 and Algorithm 2.1. The multigrid schedule for the cases in section 5 are $(7, 5, 3, 1)$ and $(7, 3, 1)$ for Chebyshev-Jacobi and Chebyshev-Schwarz respectively.

While the weighted ASM method in (4.1) and RAS method are not SPD, experimental evidence suggests that the complex spectra of $S_{ASM} A$ and $S_{RAS} A$ have small imaginary part. Following Manteuffel [26], Algorithm 2.1 can be adapted to the non-SPD case by replacing the interval $[\lambda_{\text{min}}, \lambda_{\text{max}}]$ with an ellipse centered about $\theta$ with a focal distance $\delta$ that encloses the convex hull of the spectra. However, since the eigenvalues that have an imaginary part have moduli much less than $\lambda_{\text{max}}(SA)$, adapting the algorithm to the non-SPD case is not necessary. Numerous numerical
results ([14, 31, 37, 35], etc.) confirm the efficacy of this approach, as pMG with Chebyshev-accelerated ASM smoothing is the default preconditioner in nekRS.

4.2. Preconditioning with Low Order Operators. In [33], 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 [34, 32, 3, 6].

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. boomerAMG [46], is used with the following setup to solve the low order system:

- PMIS coarsening
- 0.25 strength threshold
- Extended + i interpolation ($p_{max} = 4$)
- L1 Jacobi smoothing
- One V-cycle for preconditioning
- Smoothing on the coarsest level

We denote this preconditioning strategy as SEMFEM.

5. High Order Cases. We describe four model problems that are used to test the high-order $p$-multigrid (pMG) preconditioners. The first is a stand-alone Poisson solve with the Kershaw mesh ($\varepsilon = 1, 0.3, 0.05$). The others are modest-scale NS problems, where the pressure Poisson problem is solved over 2,000 timesteps. Details regarding the spectral element discretization are given in [35]. The problem sizes are listed in Table 2 and range from small ($n=16M$ points) to moderate ($n=180M$).

5.1. Poisson. The Kershaw family of meshes [19, 18] has been proposed as the basis for a high-order Poisson-solver benchmark by Center for Efficient Exascale Dis...

\footnote{Larger cases for full-scale runs on Summit with $n=51B$ are reported in [31].}
cretization (CEED) within the DOE Exascale Computing Project (ECP). This family is parameterized 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 \( \varepsilon \) decreases, the mesh deformation and aspect ratio increase along with it. The Kershaw mesh is shown in Figure 6 for \( \varepsilon = 1, 0.3, 0.05 \). The domain \( \Omega = [-1/2, 1/2]^3 \) with Dirichlet boundary conditions on \( \partial \Omega \). The right hand side for (1.1) is set to

\[
\begin{equation}
\begin{split}
f(x, y, z) = 3\pi^2 \sin(\pi x) \sin(\pi y) \sin(\pi z).
\end{split}
\end{equation}
\]

The linear solver terminates after reaching a relative residual reduction of \( 10^{-8} \). For the 1st-Cheb, \( \lambda_{\min}^{opt} \) smoother method, a random right-hand side is used to tune \( \lambda_{\min} \), rather than using the correlation from (2.24). The solver used is GMRES(30) preconditioned with a single pMG V-cycle. Since this test case solves the Poisson equation, there is no timestepper needed for the model problem.

5.2. Navier-Stokes. For the pressure-Poisson tests, three flow cases are considered, as depicted in Figure 7. The first three cases correspond 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 [21]. The 67 pebble case is constructed using an alternate Voronoi cell approach, and includes chamfers [38]. As such, the 67 pebble case is a more complex geometry. 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 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 \( t = 10 \) convective
time units is used for the 146, 1568, and 67 pebble cases, respectively, to provide an initially turbulent flow.

In all cases, solver results are collected over 2,000 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, \( \bar{u} \). Projection is standard practice in nekRS as it can reduce the initial residual by orders of magnitude at the cost of just two matrix-vector products in \( A \) per step [16]. The solver used is GMRES(15) preconditioned with a single pMG V-cycle.

6. **High Order Results.** Here we consider the solver performance results for the test cases of section 5. 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 2 IBM POWER9 processors and 6 NVIDIA V100 GPUs. Each case is preconditioned using pMG with a schedule of \((7,5,3,1)\) for Jacobi-based and \((7,3,1)\) for Schwarz-based Chebyshev smoothing. While \( p = 1 \) is treated as the coarse grid level for pMG, the problem sizes still scale linearly with the number of spectral elements. At the \( p = 1 \) level, boomerAMG is used on the CPU is used with the settings described in subsection 4.2.

6.1. **Loss of Symmetry and GMRES versus CG.** One concern regarding the one-sided V-cycle approach is the loss of symmetry in the preconditioner, requiring the use of GMRES instead of CG as a KSP solver. There are, however, several strategies to mitigate this issue. Using restarted GMRES(\( m \)) bounds the orthogonalization cost for \( n \) degrees of freedom to at most \( O(m^2n) \) per iteration. Further, an effective preconditioning strategy can reduce this cost further by ensuring that \( k \) is small. An additional concern using GMRES(\( m \)) is the \( O(m) \) all-reduce operations per iteration. However, by utilizing classical Gram-Schmidt orthogonalization, GMRES(\( m \)) requires only two all-reduce operations per iteration. Concerns over the loss of orthogonality are avoided by keeping \( m \) small (e.g., \( m = 15 \) or \( m = 30 \)). While not used in this current work, Thomas and coworkers demonstrated that post-modern GMRES reduces the number of synchronizations to a single all-reduce while preserving backward stability [44].

Kershaw (subsection 5.1) results utilizing \( 1^{st} \) Cheb, Jacobi(3,3), \((7,5,3,1)\), pMG as a symmetric preconditioner with CG and GMRES(30) as the KSP solvers are shown in Table 3. Since the Kershaw case described in subsection 5.1 uses a \( 10^{-8} \) residual reduction for the convergence criterion of the solver, it follows that GMRES should yield a lower iteration count than CG. This is because CG minimizes the \( A \)-norm of the error vector, while GMRES minimizes the \( L_2 \)-norm of the error vector [40]. A more meaningful convergence criteria would be on the \( L_2 \)-norm of the error vector, however, this would not be known. As such, the results presented in Table 3 are meant to illustrate that the time to apply a single GMRES iteration is similar to CG. Consider that, for this case, GMRES(30) is employed, thereby over-estimating the cost per iteration for the NS cases wherein GMRES(15) is used. Lastly, the most effective preconditioning strategies utilize Schwarz-based Chebyshev smoothing. The ASM and RAS methods considered herein are asymmetric, and thus are not suitable for use with CG.

6.2. **Kershaw Results.** Results for the Kershaw case for \( \varepsilon = 1.0, 0.05 \) are shown in Figure 8 and Figure 9. To mitigate the effects of system noise, the reported solve

\[ \text{Results for Kershaw with } \varepsilon = 0.3 \text{ are shown in Figure 12.} \]
Comparison of CG and GMRES(30) for the Kershaw cases using 1st-Cheb, Jacobi(3,3), (7,5,3,1) as preconditioner.

| ε  | CG iter. | Time per CG iter. | GMRES iter. | Time per GMRES iter. |
|----|----------|-------------------|-------------|-----------------------|
| 1  | 20       | 1.26 x 10^{-2}    | 9           | 1.27 x 10^{-2}        |
| 0.3| 286      | 1.38 x 10^{-2}    | 123         | 1.50 x 10^{-2}        |
| 0.05| 1000      | 1.39 x 10^{-2}    | 474         | 1.52 x 10^{-2}        |

Times are based on minimum time to solution over 50 trials. The reported number of matrix-vector products are for the finest grid, $p = 7$.

When $\varepsilon = 1$, the time to solution is minimized by utilizing a symmetric V-cycle with relatively low-order Chebyshev-accelerated RAS smoothing. For this case, SEMFEM is a comparatively poor method. While the iteration count is decreased with respect to increasing order, the overall cost of applying the heavier smoother translates to a higher cost per solve. This can especially be observed in the increased work requirement in terms of matrix-vector products, Figure 8d-f. At larger scales, however, the scalability of the AMG coarse grid solve may dominate the cost of the preconditioner, and thus anything to reduce the number of coarse grid solves may prove beneficial. In this scenario one should expect the multigrid approximation property constant (2.10) to be quite low for the case with no geometric deformation. In this regime, the theoretical prediction in Figure 2, states that the convergence is improved using a symmetric $(k,k)$ V-cycle with as opposed to the one-sided $(2k,0)$ V-cycle. Benchmarking demonstrates that a single boomerAMG V-cycle iteration for $p = 1$ on the CPU is nearly 12 times the cost of a matrix-vector product. For larger cases, wherein the number of AMG levels needed for a single boomerAMG V-cycle increases, the relative cost of the coarse grid solve will increase relative to the cost of a matrix-vector product.

For $\varepsilon = 0.05$, the fastest time to solution is reached using SEMFEM preconditioning. The pMG methods are significantly more expensive. For the 4th and opt. 4th-kind Chebyshev-accelerated RAS schemes considered, the time to solution is lowered by increasing the order. Remarkably, the 4th-kind and optimized 4th-kind Chebyshev schemes are generally equivalent to, if not better than, optimizing the $\lambda_{\min}$ parameter for the standard 1st Chebyshev scheme. This allows for increased performance with high order Chebyshev smoothing, without the requirement of tuning an additional parameter. Further, with the extreme geometric deformation, the multigrid approximation property constant (2.10) is expected to be quite large for this case. The results in Figure 2 predict that the one-sided approach yields a better convergence rate. The results confirm this theoretical expectation.

A summary of the results for the Kershaw case is shown in Table 4. This table reports the solver yielding the lowest time to solution (in seconds), $T_S$, the iteration count, and the speedup over the time to solution of the default nekRS solver, $T_D$. The ratio between the time spent doing coarse grid solves for the default solver, $(T_{crs})_D$, and the time spent doing coarse grid solves for the fastest solver, $(T_{crs})_S$, is reported. The default nekRS solver is 1st-Cheb, ASM(3,3),(7,3,1). For the Kershaw case, these

---

4 In large-scale fluid mechanics applications, this overhead is easily amortized over the $10^4$–$10^6$ timesteps required. Reasonably good values of $\lambda_{\min}^{opt}$ do not depend on the RHS, allowing this to be part of the setup cost of the preconditioner.
tables demonstrate that optimizing $\lambda_{\text{min}}$ in the 1st-kind Chebyshev scheme greatly improves the solver performance. While $\varepsilon = 1, 0.3$ do not benefit from the use of a one-sided V-cycle, the use of the one-sided V-cycle, in conjunction with the optimized 4th-kind Chebyshev smoother, is able to increase the solver speedup relative to the default solver by another 13% for $\varepsilon = 0.05$. A 75% speedup is achieved over the default solver for $\varepsilon = 1, 0.05$. For $\varepsilon = 0.3$, a much more modest 35% is achieved.

6.3. Navier-Stokes. Results for the 1568 and 67 pebble cases are shown in Figure 10 and Figure 11, respectively. Since the 4th and optimized 4th-kind Chebyshev smoothers are comparable to the 1st!kind Chebyshev smoother with optimized $\lambda_{\text{min}}$ as shown in subsection 6.2, this smoother is omitted from these cases.

The lowest time to solution for the 1568 pebble case is achieved using 4th-Cheb, ASM(12, 0), Table 4. Figure 10d,g shows that the number of matrix-vector products remains nearly constant with respect to the order, yielding a lower time to solution by minimizing the coarse grid cost. The performance of the 1st-kind Chebyshev smoother, however, plateaus around $k = 3, \tilde{k} = 6$, especially for the Schwarz-based smoothers, see Figure 10g,h. Although some improvement is observed through the use of the one-sided V-cycle, an additional benefit is observed by using the alter-

---

5Results for the 146 pebble case are shown in Figure 13.
nate Chebyshev smoothers from Lottes’s work [23]. Without the added benefit of the one-sided V-cycle, the fastest solver for this case yields a 17% speedup over the default. However, enabling this one-sided V-cycle approach further increases the solver performance to a 27% speedup over the default solver (Table 4).

The 67 pebble case is distinct from the other two pebble meshes. The 146 and 1568 pebble cases are meshed using an all-hex meshing strategy developed by Lan and coworkers [21]. The 67 pebble case includes chamfers, and is meshed using an alternate Voronoi cell approach [38]. The resulting mesh proves to be much more challenging for the pMG based strategies. Nevertheless, pMG with one-sided Chebyshev smoothing prove promising for this case. The time to solution is minimized at relatively high orders with RAS-based Chebyshev. This is further improved through the use of the one-sided V-cycle. The total work required per solve is reduced at higher orders for the Schwarz-based approaches. Through tuning the pMG parameters, a significant speedup of 81% is achieved over the default solver. In this case, the fastest pMG preconditioner is $4^{th}$-opt-Cheb, RAS(12,0).

7. Conclusions. In this work, we explore the use of one-sided V-cycle using Chebyshev smoothing as a preconditioner, both in a 2D finite difference example, as well as a pMG preconditioner for the pressure Poisson equation arising from the spectral element discretization of the NS equation. Further, we explore the efficacy of
novel Chebyshev smoothers based on the work of Lottes [23] and demonstrate their improvement over the 1st-kind Chebyshev smoothers. This improvement enables increased solver performance, especially at high Chebyshev degrees. The benefit to this approach is further decreasing the coarse grid cost.

The authors plan on applying the Lottes’s novel Chebyshev smoothers to AMG solvers, such as BoomerAMG [46] and Trilinos/MueLu [36]. The application of this work are two fold. First, improved AMG solvers will benefit the low-order SEMFEM preconditioning strategy for high-order finite elements. Secondly, improvements in
AMG solvers will benefit the overall solver community. The authors also plan on applying the ideas developed herein to larger problems.

**Acknowledgments.** 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 James Lottes for his insightful comments and suggestions, in addition to his reading of the manuscript. The authors thank YuHsiang Lan, David Alan Reger, and Haomin Yuan for providing visualizations and mesh files.

**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] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang, Multigrid smoothers for ultra-parallel computing, SIAM Journal on Scientific Computing, 33 (2011), pp. 2864–2887.

[3] P. D. Bello-Maldonado and P. F. Fischer, Scalable low-order finite element preconditiners for high-order spectral element poisson solvers, SIAM Journal on Scientific Computing, 41 (2019), pp. S2–S18.

[4] A. Biesz, W. D. Gropp, and L. N. Olson, Reducing communication in algebraic multigrid with multi-step node aware communication, The International Journal of High Performance Computing Applications, 34 (2020), pp. 547–561.

[5] X.-C. Cai and M. Sarkis, A restricted additive schwarz preconditioner for general sparse linear systems, Siam journal on scientific computing, 21 (1999), pp. 792–797.

[6] C. Canuto, P. Geravasio, and A. Quarteroni, Finite-element preconditioning of g-ni spectral methods, SIAM Journal on Scientific Computing, 31 (2010), pp. 4422–4451.

[7] N. Chalmers, A. Karakus, A. P. Austin, K. Swirydowicz, and T. Warburton, libParanu- mal: a performance portable high-order finite element library.

[8] T. Chan and W. Wan, Analysis of projection methods for solving linear systems with multiple right-hand sides, SIAM J. Sci. Comput., 18 (1997), pp. 1698–1721.

[9] M. Deville, P. Fischer, and E. Mund, High-order methods for incompressible fluid flow, Cambridge University Press, Cambridge, 2002.

[10] R. D. Falgout and J. B. Schroder, Non-galerkin coarse grids for algebraic multigrid, SIAM Journal on Scientific Computing, 36 (2014), pp. C309–C334.

[11] N. Fehn, P. Munch, W. A. Wall, and M. Kronbichler, Hybrid multigrid methods for high-order discontinuous galerkin discretizations, Journal of Computational Physics, 415 (2020), p. 109538.

[12] N. Fehn, W. Wall, and M. Kronbichler, Efficiency of high-performance discontinuous Galerkin spectral element methods for under-resolved turbulent incompressible flows, Int. J. Numer. Methods Fluids, (2018).

[13] P. Fischer, An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations, J. Comput. Phys., 133 (1997), pp. 84–101.

[14] P. Fischer, S. Kerkmeeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, et al., Nekrs, a gpu-accelerated spectral element navier–stokes solver, Parallel Computing, 114 (2022), p. 102982.

[15] P. Fischer, N. Miller, and H. Tufo, An overlapping Schwarz method for spectral element simulation of three-dimensional incompressible flows, in Parallel Solution of Partial Differential Equations, P. Bjørstad and M. Luskin, eds., Berlin, 2000, Springer, pp. 158–180.

[16] 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.

[17] W. Hackbusch, Multi-grid convergence theory, in Multigrid methods, Springer, 1982, pp. 177–219.

[18] D. S. Kershaw, Differentiation of the diffusion equation in lagrangian hydrodynamic codes, Journal of Computational Physics, 39 (1981), pp. 375–395.

[19] 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, Mar. 2021, https://doi.org/10.5281/zenodo.4672664, https://doi.org/10.5281/zenodo.4672664.

[20] 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.

[21] Y.-H. Lan, P. Fischer, E. Merzari, and M. Min, All-hex meshing strategies for densely packed spheres, preprint arXiv:2106.00196, (2021).

[22] S. Loseel, R. Nabben, and D. B. Szyld, On hybrid multigrid-schwarz algorithms, Journal of Scientific Computing, 36 (2008), pp. 165–175.

[23] J. Lottes, Optimal polynomial smoothers for multigrid v-cycles, preprint arXiv:2202.08830, (2022).

[24] 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.

[25] R. Lynch, J. Rice, and D. Thomas, Direct solution of partial difference equations by tensor product methods, Numer. Math., 6 (1964), pp. 185–190.

[26] T. A. Manteuffel, The tchebychev iteration for nonsymmetric linear systems, Numerische Mathematik, 28 (1977), pp. 307–327.
8. Supplementary Material.

8.1. Coefficients for the Optimized 4th-kind Chebyshev Smoother. Values for $\beta$ for the optimized 4th-kind Chebyshev smoother developed by Lottes are tabulate in Table 5.
8.2. Geometric $p$ Multigrid for Pressure Poisson. Mesh quality metrics for the high-order cases are shown in Table 6.

8.2.1. Poisson. Kershaw results for $\varepsilon = 0.3$ are shown in Figure 12. When $\varepsilon = 0.3$, however, higher-order Chebyshev smoothing can yield a lower time to solution. As shown in Figure 12d, the time to solution for the $4^{th}$ and $4^{th}_{opt}$ Chebyshev schemes tend to improve with high orders, even up to $k = 6$ for the symmetric V-cycle (or $\tilde{k} = 12$ for the one-sided V-cycle). This demonstrates a major improvement over the standard $1^{st}$ Chebyshev scheme, which tends to yield a minimum time to solution at $k = 3$. Similar to the $\varepsilon = 1$ case, SEMFEM is not the preconditioner yielding the fastest time to solution. However, this approach becomes comparable to Jacobi-based Chebyshev smoothing.

8.2.2. Navier-Stokes. pMG results for the 146 pebble Navier-Stokes case mentioned in subsection 5.2 are shown in Figure 13.
Fig. 13. 146 pebble results.
Table 5

Tabulated values of $\beta$ for the optimized 4th-kind Chebyshev smoother.

| $k$ | $\beta_1^{(k)}$ | $k$ | $\beta_1^{(k)}$ | $k$ | $\beta_1^{(k)}$ |
|-----|-----------------|-----|-----------------|-----|-----------------|
| 1   | 1.12500000000000 | 10  | 1.00030312229652 | 14  | 1.00011490538261 |
| 2   | 1.02387287570313 | 1.00304840660796 | 1.00115246376914 |
|     | 1.26408905371085 | 1.01077022715387 | 1.00405357332624 |
| 3   | 1.00842544782028 | 1.02619011597640 | 1.00979505731535 |
|     | 1.08867383928730 | 1.0523174933755 | 1.0194130472994 |
|     | 1.33753125906182 | 1.09255743207549 | 1.0340124535436 |
| 4   | 1.00391310422785 | 1.1508376663972 | 1.054805966629 |
|     | 1.04035811188593 | 1.23172250870894 | 1.08311420301813 |
|     | 1.14863985462540 | 1.34068020244660 | 1.1204081660892 |
|     | 1.38268692410000 | 1.48386124407111 | 1.1683095655446 |
| 5   | 1.00212930146164 | 1.0023058595209 | 1.22872122288238 |
|     | 1.02173711549260 | 1.00316750240284 | 1.30365305707817 |
|     | 1.07872431926033 | 1.00817249363004 | 1.3954810453678 |
|     | 1.19810659296633 | 1.01982968566342 | 1.50681646290583 |
|     | 1.41322542791682 | 1.03950210235234 | 1.50009404750752 |
| 6   | 1.00128517255940 | 1.06965042700541 | 1.00094291696343 |
|     | 1.01304293035233 | 1.11305754295742 | 1.00331449056444 |
|     | 1.0467821524113 | 1.17208762755664 | 1.0080294833816 |
|     | 1.11616489419675 | 1.25283005676792 | 1.01584236529140 |
|     | 1.23892020184444 | 1.35725579915195 | 1.02772083317705 |
|     | 1.43524297106744 | 1.49101672563419 | 1.0445935422831 |
| 7   | 1.00083463497912 | 1.00017947200828 | 1.06750761206125 |
|     | 1.00843949430122 | 1.00180189140819 | 1.09760925458899 |
|     | 1.03088707768713 | 1.00634861907307 | 1.1361355366157 |
|     | 1.07408340920003 | 1.01537864566306 | 1.18452361426236 |
|     | 1.15036186707366 | 1.03056942830760 | 1.24432087304475 |
|     | 1.27116474046139 | 1.05376019693943 | 1.31728090833932 |
|     | 1.4518658649364 | 1.08699862592072 | 1.40536543893560 |
|     | 1.6083463497912 | 1.13259138097193 | 1.51077872501845 |
| 8   | 1.00057246631197 | 1.19316273351872 | 1.60007794828179 |
|     | 1.00574726624115 | 1.27171293675110 | 1.00078126847253 |
|     | 1.02050187922941 | 1.37169337969799 | 1.00274487974401 |
|     | 1.05019803445565 | 1.49708418575562 | 1.00662291012716 |
|     | 1.1015572984941 | 1.9316273351872 | 1.0130985883971 |
|     | 1.18060428068566 | 1.00014249121559 | 1.02289448329337 |
|     | 1.29838585382576 | 1.0041290632629 | 1.0367324109983 |
|     | 1.46486073151099 | 1.0050320896298 | 1.05559875719896 |
| 9   | 1.00040960072832 | 1.01216910518495 | 1.06024848045560 |
|     | 1.00412439506106 | 1.02414874342792 | 1.08024848045560 |
|     | 1.01460212482666 | 1.04238158880820 | 1.11172607131497 |
|     | 1.03561113626671 | 1.06842008128700 | 1.15112543430172 |
|     | 1.0713992529194 | 1.10399010936759 | 1.1996584614973 |
|     | 1.1268273710962 | 1.15012748242645 | 1.2586841744946 |
|     | 1.2078219047029 | 1.21178181910125 | 1.3296241265664 |
|     | 1.3212193716746 | 1.2885426865128 | 1.41421360695576 |
|     | 1.47529642820699 | 1.38432619380991 | 1.5142791730346 |
|     |                   | 1.5022918757368 |                   |
Table 6
Mesh quality metrics for cases from Figure 6 and Figure 7.

| Case Name    | Scaled Jacobian (min/max/avg) | Aspect Ratio (min/max/avg) |
|--------------|-------------------------------|-----------------------------|
| K. (ε = 1)   | 1 / 1 / 1                     | 1 / 1 / 1                   |
| K. (ε = 0.3) | 0.316 / 1 / 0.841             | 1.08 / 20.1 / 4.64          |
| K. (ε = 0.05)| $1.86 \times 10^{-2}$ / 1 / 0.733 | 1.1 / 162 / 21.7              |
| 146 pebble   | $4.31 \times 10^{-2}$ / .977 / .419 | 1.07 / 56.9 / 7.14          |
| 1568 pebble  | $2.59 \times 10^{-2}$ / .99 / .371 | 1.12 / 108 / 12.6           |
| 67 pebble    | $5.97 \times 10^{-3}$ / .970 / .38 | 1.17 / 204 / 13.2           |