LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-473191

Multigrid Smoothers for Ultra-Parallel Computing

A. H. Baker, R. D. Falgout, T. V. Kolev, U. M. Yang

March 11, 2011

SIAM Journal on Scientific Computing

MULTIGRID SMOOTHERS FOR ULTRA-PARALLEL COMPUTING

ALLISON H. BAKER*, ROBERT D. FALGOUT*, TZANIO V. KOLEV*, AND ULRIKE MEIER YANG*

Abstract. This paper investigates the properties of smoothers in the context of algebraic multigrid (AMG) running on parallel computers with potentially millions of processors. The development of multigrid smoothers in this case is challenging, because some of the best relaxation schemes, such as the Gauss-Seidel (GS) algorithm, are inherently sequential. Based on the sharp two-grid multigrid theory from [22, 23] we characterize the smoothing properties of a number of practical candidates for parallel smoothers, including several C-F, polynomial, and hybrid schemes. We show, in particular, that the popular hybrid GS algorithm has multigrid smoothing properties which are independent of the number of processors in many practical applications, provided that the problem size per processor is large enough. This is encouraging news for the scalability of AMG on ultra-parallel computers. We also introduce the more robust ℓ_1 smoothers, which are always convergent and have already proven essential for the parallel solution of some electromagnetic problems [29].

1. Introduction. Multigrid (MG) linear solvers are optimal methods because they require $O(N)$ operations to solve a sparse system with N unknowns. Consequently, multigrid methods have good scaling potential on parallel computers, since we can bound the work per processor as the problem size and number of processors are proportionally increased (weak scaling). Near ideal weak scaling performance has been demonstrated in practice. For example, the algebraic multigrid (AMG) solver BoomerAMG [25] in the *hypre* software library [26] has been shown to run effectively on more than 125 thousand processors [21, 5].

One critical component of MG is the smoother, a simple iterative method such as Gauss-Seidel (GS). In the classical setting, the job of the smoother is to make the underlying error smooth so that it can be approximated accurately and efficiently on a coarser grid. More generally, the smoother must eliminate error associated with large eigenvalues of the system, while the coarse-grid correction eliminates the remaining error associated with small eigenvalues.

Some of the best smoothers do not parallelize well, e.g., lexicographical GS. Others used today, while effective on hundreds of thousands of processors, still show some dependence on parallelism and may break down on the millions of processors expected in the next generation machines (we use the term processor here in a generic sense, and distinguish it from cores only when necessary). One such smoother is the *hybrid GS* smoother used in BoomerAMG, which uses GS independently on each processor and updates in a Jacobi-like manner on processor boundaries. In practice hybrid GS is effective on many problems. However, because of its similarity to a block Jacobi method, there is no assurance of obtaining the good convergence of lexicographical GS. In fact, Hybrid GS may perform poorly or even diverge on certain problems, and its scalability has often been cited as a concern as the number of blocks increase with increasing numbers of processors or as block sizes decrease (see, e.g. [1, 16, 37]). For these reasons, previous papers have studied alternatives such as using polynomial smoothers [1] or calculating weighting parameters for hybrid GS [38]. Yet despite its shortcomings, hybrid GS remains the default option in *hypre* because of its overall efficiency and robustness. Therefore, one of the main purposes of this paper is to

*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551. This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-JRNL-435315).

better understand the potential of block smoothers like hybrid GS on millions of processors. We show that these hybrid smoothers can in fact exhibit good smoothing properties independent of parallelism, as long as the blocks satisfy certain properties (e.g., the blocks have some minimal size).

There are many other well-known smoothers that exhibit parallel-independent smoothing properties. In particular, methods like weighted Jacobi (both pointwise and blockwise), red/black GS, Chebyshev and Krylov-based polynomial methods have been extensively studied in classical works such as [13, 35, 24, 7]. In practice, each of these methods have their drawbacks. For example, weighted Jacobi requires the estimation of an ideal weight [38] and Chebyshev involves estimating an eigenvalue interval [1]. For multi-colored GS [1], the number of parallel communications required per iteration is proportional to the number of colors, hence it tends to be slow, especially on coarser grids in AMG where the number of colors is difficult to control. Therefore, the secondary purpose of this paper is to study and identify smoothers that are practical for AMG in the context of millions of processors. To this end, we analyze a variety of candidates for smoothing, revisiting some of the classics as well, under a common framework based on the recent two-grid theory in [22, 23]. In addition, to both complement and support the analysis, we perform numerical experiments for the most promising parallel smoothers using both message-passing (with MPI) and threading (with OpenMP).

The structure of the paper is as follows. In Section 2, we introduce our approach for doing smoothing analysis in general, and we then analyze several specific classes of smoothers in Section 3 through Section 6, including C-F, polynomial, hybrid, and ℓ_1 smoothers. We present numerical experiments in Section 7, and we make concluding remarks in Section 8.

2. Smoothing Analysis. Our smoothing analysis is based on the two-grid variational multigrid theory from [22], which was developed for general relaxation and coarsening processes. In this section, we first summarize this theory and then describe our general approach for applying it to smoother analysis. We represent the standard Euclidean inner product by $\langle \cdot,\ \cdot \rangle$ with associated norm, $\|\cdot\| := \langle \cdot,\ \cdot \rangle^{1/2}$. The A-norm (or energy norm) is defined by $\|\cdot\|_A := \langle A\ \cdot,\ \cdot \rangle^{1/2}$ for vectors, and as the corresponding induced operator norm for matrices.

Consider solving the linear system of equations

$$A\mathbf{u} = \mathbf{f}, \tag{2.1}$$

where $\mathbf{u}, \mathbf{f} \in \mathbb{R}^n$ and A is a symmetric positive definite (SPD) matrix. Define the *smoother* (relaxation) error propagator by

$$I - M^{-1}A, \tag{2.2}$$

and assume that the smoother is convergent (in energy norm $\|\cdot\|_A$), i.e. assume that $M^T + M - A$ is SPD. This is equivalent to the condition $\langle A\mathbf{x},\ \mathbf{x}\rangle \leq 2\langle M\mathbf{x},\ \mathbf{x}\rangle$ for all $\mathbf{x}$, since $\langle M\mathbf{x},\ \mathbf{x}\rangle = \langle M^T\mathbf{x},\ \mathbf{x}\rangle$. Note that we often refer to the matrix M as the smoother. Denote the symmetrized smoother by

$$\widetilde{M} = M^T(M^T + M - A)^{-1}M, \tag{2.3}$$

so that $I - \widetilde{M}^{-1}A = (I - M^{-1}A)(I - M^{-T}A)$. Let $P : \mathbb{R}^{n_c} \mapsto \mathbb{R}^n$ be the *interpolation* (or *prolongation*) operator, where $\mathbb{R}^{n_c}$ is some lower-dimensional (coarse) vector space

of size n_c. The two-grid multigrid error transfer operator with no post-smoothing steps is then given by

$$E_{TG} = (I - P(P^T AP)^{-1}P^T A)(I - M^{-1}A), \tag{2.4}$$

where P^T is the *restriction* operator and $A_c = P^T AP$ is the Galerkin *coarse-grid operator*. Note that coarse-grid correction involves an A-orthogonal projection onto range(P).

Let $R : \mathbb{R}^n \mapsto \mathbb{R}^{n_c}$ be any matrix for which $RP = I_c$, the identity on $\mathbb{R}^{n_c}$, so that PR is a projection onto range(P). We can think of R as defining the *coarse-grid variables*, i.e., $\mathbf{u}_c = R\mathbf{u}$. Also, let $S : \mathbb{R}^{n_s} \mapsto \mathbb{R}^n$ be any full-rank matrix for which $RS = 0$, where $n_s = n - n_c$. Here, the unknowns $\mathbf{u}_s = S^T\mathbf{u}$ are analogous to the fine-grid-only variables (i.e., F-points) in AMG. In addition, R and S form an orthogonal decomposition of $\mathbb{R}^n$: any $\mathbf{e}$ can be expressed as $\mathbf{e} = S\mathbf{e}_s + R^T\mathbf{e}_c$, for some $\mathbf{e}_s$ and $\mathbf{e}_c$. The next theorem summarizes one of the main convergence results in [22].

THEOREM 2.1. *(see Theorem 2.2 in [22])*

$$\|E_{TG}\|_A^2 \leq 1 - \frac{1}{K}, \quad \textit{where} \quad K = \sup_{\mathbf{e}} \frac{\|(I - PR)\mathbf{e}\|_{\widetilde{M}}^2}{\|\mathbf{e}\|_A^2} \geq 1. \tag{2.5}$$

Theorem 2.1 gives conditions that P must satisfy in order to achieve a fast uniformly convergent multigrid method. It is clear that to make K small, eigenvectors of A belonging to small eigenvalues must either be interpolated accurately by P or else attenuated efficiently by the smoother (since the denominator is small for these eigenvectors). For brevity, we refer to these as small eigenvectors. The choice of which small eigenvectors to eliminate by smoothing and which to eliminate by coarse-grid correction depends on the "localness" of the modes. Essentially, modes that can be eliminated by a local process (i.e., one that is equivalent to applying an operator with a comparable sparse nonzero structure to A) should be handled by the smoother.

In the next two sections, we discuss two approaches for employing Theorem 2.1 to analyze smoothers. In Section 2.1, we use the idea of measuring (or bounding) the two-grid convergence factor by assuming an ideal interpolation operator. This is essentially what is done in classical smoothing factor analysis introduced in [13], where the smoothing factor measures the effectiveness of relaxation on the oscillatory Fourier modes, motivated by the assumption that interpolation (our ideal interpolation) eliminates the smooth Fourier modes. An important aspect of this approach is that it is explicitly tied to the (ideal) coarse-grid correction.

The comparative approach described in Section 2.2 is similar to most other smoother analyses, where either weighted Richardson or Jacobi relaxation is used to measure the relative quality of a smoother [24, 9, 10, 32, 33, 31, 34, 11, 12]. A general comparison lemma similar to our Lemma 2.3 was stated in [31]. One limitation of this approach is that coarse-grid correction is not explicitly taken into account, so in cases such as Maxwell's equation, care must be taken to compare with a suitable smoother that is already known to work well for Maxwell's equation.

2.1. Smoothing Analysis with Ideal Interpolation. One approach for using the above theory to do smoothing analysis is to consider the best K in Theorem 2.1 by substituting the P that minimizes the following for a given R

$$K_\star = \inf_{P\,:\,RP=I_c} \sup_{\mathbf{e}} \frac{\|(I - PR)\mathbf{e}\|_{\widetilde{M}}^2}{\|\mathbf{e}\|_A^2}. \tag{2.6}$$

The following theorem evaluates this inf-sup problem.

THEOREM 2.2. *(see Theorem 3.1 in [22]) Assume that R, S, and P satisfy $RS = 0$ and $RP = I_c$ as above. Then $K_\star$ in (2.6) is given by*

$$K_\star = \sup_{\mathbf{e}_s} \frac{\langle S^T \widetilde{M} S \mathbf{e}_s,\ \mathbf{e}_s \rangle}{\langle S^T A S \mathbf{e}_s,\ \mathbf{e}_s \rangle} = \frac{1}{\lambda_{\min}((S^T \widetilde{M} S)^{-1}(S^T A S))}, \tag{2.7}$$

and the corresponding minimizer is

$$P_\star = (I - S(S^T A S)^{-1} S^T A) R^T. \tag{2.8}$$

Equation (2.8) defines the so-called *ideal interpolation* operator. Notice that, if $K_\star$ is uniformly bounded with respect to parameters such as the mesh spacing, then using $P_\star$ as the interpolation operator results in a uniformly convergent two-grid method. Since the inverse of $S^T A S$ may not be sparse, this is generally not a good practical choice for interpolation. However, it is reasonable to use $P_\star$ (and hence $K_\star$) to analyze smoothing.

We consider two settings in the analysis that follows, depending on the particular smoother. The first is the classical AMG setting where the coarse-grid variables $R\mathbf{u}$ are a subset of the fine-grid variables:

$$R^T = \begin{bmatrix} 0 \\ I_c \end{bmatrix}; \quad S = \begin{bmatrix} I_f \\ 0 \end{bmatrix}; \quad P_\star = \begin{bmatrix} -A_{ff}^{-1} A_{fc} \\ I_c \end{bmatrix}. \tag{2.9}$$

The second setting corresponds more closely to the classical smoothing factor analysis [13], where the coarse-grid variables span the space of the n_c "smallest" (they do not have to strictly be the smallest as we discuss later) eigenvectors of A:

$$R^T = [\mathbf{v}_1, \ldots, \mathbf{v}_{n_c}]; \quad S = [\mathbf{v}_{n_c+1}, \ldots, \mathbf{v}_n]; \quad P_\star = R^T. \tag{2.10}$$

2.2. Comparative Smoothing Analysis. Direct evaluation of $K_\star$ in (2.6) is not always straightforward. However, one useful technique that we use below is to compare the $K_\star$ for one smoother to that of another with well-known smoothing properties (e.g., Gauss-Seidel). Writing $K = K(M)$ in (2.5) as a function of the smoother (similarly for $K_\star$), we articulate this approach in the next lemma.

LEMMA 2.3. *Suppose that M_1 and M_2 are two convergent smoothers for A that satisfy*

$$\langle \widetilde{M}_1 \mathbf{x},\ \mathbf{x} \rangle \leq c \langle \widetilde{M}_2 \mathbf{x},\ \mathbf{x} \rangle \tag{2.11}$$

for all $\mathbf{x}$, with a fixed constant c. Then, for any choice of the interpolation operator in the two-grid multigrid method, we have that

$$K(M_1) \leq c K(M_2),$$

and in particular, $K_\star(M_1) \leq c K_\star(M_2)$. In other words, multigrid methods using M_1 and M_2 have comparable parallel scalability properties, provided c is independent of the problem size and the number of processors. Therefore, when (2.11) holds, we say that M_1 has multigrid smoothing properties comparable to M_2.

Proof. The proof follows immediately from (2.5) and (2.6). ∎

REMARK 2.1. *Note that the above result can also be analogously stated in terms of the sharp two-grid theory of [23] since we can write the constant $K_\sharp$ in that theory as*

$$K_\sharp = \sup_{\mathbf{e}} \frac{\left\|(I-\pi_{\widetilde{M}})\mathbf{e}\right\|_{\widetilde{M}}^2}{\|\mathbf{e}\|_A^2} = \sup_{\mathbf{v}\in\text{range}(I-\pi_A)} \inf_{\mathbf{w}\,:\,\mathbf{v}=(I-\pi_A)\mathbf{w}} \frac{\langle \widetilde{M}\mathbf{w},\ \mathbf{w}\rangle}{\langle A\mathbf{v},\ \mathbf{v}\rangle},$$

where $\pi_X = P(P^TXP)^{-1}P^TX$ denotes the X-orthogonal projection onto range(P) *for any SPD matrix X.*

In this paper, we determine the constant c in Lemma 2.3 through direct analysis. However, we can also write c in terms of a few general constants from [22] and instead estimate those to do the analysis. For brevity, we do not include the latter approach in this paper, but provide them in a separate technical report [4].

3. The C-F Smoother. In this section, we apply the smoothing analysis theory from the previous section to the so-called C-F smoother. C-F smoothing corresponds to applying a smoother first to the coarse points (C-points) and then to the fine points (F-points). That C-F smoothers can be effective in practice is evident if one considers, for example, that C-F Jacobi for a standard finite difference Laplacian problem on a structured grid is equivalent to red-black Gauss-Seidel when red-black coarsening is used (as happens in AMG).

More formally, the C-F smoother is defined by

$$I - M_{CF}^{-1}A; \quad M_{CF} = \begin{bmatrix} M_{ff} & A_{fc} \\ 0 & M_{cc} \end{bmatrix}. \tag{3.1}$$

This smoother converges if and only if the following are convergent:

$$I_f - M_{ff}^{-1}A_{ff}; \quad I_c - M_{cc}^{-1}A_{cc}.$$

Therefore, one can consider using any of the convergent smoothers discussed in the following sections as the M_{ff} and M_{cc} matrices of a C-F smoother. This is typically advantageous since the principle submatrices A_{ff} and A_{cc} have better properties than A in terms of conditioning and diagonal dominance. The following theorem shows that C-F smoothing is good if F-relaxation is fast to converge.

THEOREM 3.1. *Define S as in* (2.9)*. Then $K_\star$ in* (2.6) *for the C-F smoother satisfies*

$$K_\star = \frac{1}{1-{\varrho_f}^2}; \quad \varrho_f = \|I_f - M_{ff}^{-1}A_{ff}\|_{A_{ff}}.$$

Proof. Similarly to (2.3), define

$$\widetilde{M_{ff}} = M_{ff}^T(M_{ff}^T + M_{ff} - A_{ff})^{-1}M_{ff}\,. \tag{3.2}$$

From (2.3) and the definition of M_{CF} above, we have

$$\widetilde{M} = \begin{bmatrix} M_{ff}^T & 0 \\ A_{cf} & M_{cc}^T \end{bmatrix} \begin{bmatrix} (M_{ff}^T + M_{ff} - A_{ff})^{-1} & 0 \\ 0 & (M_{cc}^T + M_{cc} - A_{cc})^{-1} \end{bmatrix} \begin{bmatrix} M_{ff} & A_{fc} \\ 0 & M_{cc} \end{bmatrix},$$

and therefore $S^T\widetilde{M}S = \widetilde{M_{ff}}$. This implies by Theorem 2.2

$$K_\star = \frac{1}{\lambda_{\min}(\widetilde{M_{ff}}^{-1}A_{ff})} = \frac{1}{1-\lambda_{\max}[(I-M_{ff}^{-1}A_{ff})(I-M_{ff}^{-T}A_{ff})]}.$$

Let $E_{ff} = I_{ff} - M_{ff}^{-1}A_{ff}$ and let $\rho(\cdot)$ denote the spectral radius of a matrix. Then, using the definition of ϱ_f and the fact that $\|B\| = \|B^T\|$ for any matrix B, we have

$$\begin{aligned}
\varrho_f{}^2 &= \|E_{ff}\|_{A_{ff}}^2 = \|A_{ff}^{1/2}E_{ff}A_{ff}^{-1/2}\|^2 = \|A_{ff}^{-1/2}E_{ff}^TA_{ff}^{1/2}\|^2 \\
&= \rho(A_{ff}^{1/2}E_{ff}A_{ff}^{-1}E_{ff}^TA_{ff}^{1/2}) = \rho(E_{ff}A_{ff}^{-1}E_{ff}^TA_{ff}) \\
&= \rho[(I-M_{ff}^{-1}A_{ff})(I-M_{ff}^{-T}A_{ff})]\,,
\end{aligned}$$

which completes the proof. □

From the above, we see that C-F smoothing is a natural smoother to use when coarse grids are selected based on compatible relaxation (CR) [14, 22], because ϱ_f is estimated as part of the CR coarsening algorithm.

4. Polynomial Smoothers. Polynomial smoothers are of practical interest for parallel computing for a couple of reasons. First, their application requires only the matrix-vector multiply routine, which is often highly-optimized on modern parallel machines. Second, they are unaffected by the parallel partitioning of the matrix, the number of parallel processes, and the ordering of the unknowns. However, as mentioned previously, one drawback is the need to calculate eigenvalue estimates. Unlike the smoothed aggregation variant of AMG, eigenvalue estimates are not needed by classical AMG, so this computational cost is extra.

We now apply the smoothing analysis from Section 2 to polynomial smoothers. Let $p_\nu(x)$ be a polynomial of degree $\nu \geq 0$ such that $p_\nu(0) = 1$, and consider the smoother

$$I - M^{-1}A = p_\nu(A). \tag{4.1}$$

The following theorem gives conditions for a good polynomial smoother.

THEOREM 4.1. *Let $A = V\Lambda V^T$ be the eigen-decomposition of A with eigenvectors $\mathbf{v}_k$ and associated eigenvalues λ_k, and define S as in* (2.10)*. Then $K_\star$ in* (2.6) *for the polynomial smoother satisfies*

$$K_\star = \left(1 - \max_{k>n_c} p_\nu(\lambda_k)^2\right)^{-1}.$$

Minimizing $K_\star$ over all p_ν, we have

$$\min_{p_\nu} K_\star \leq \left(1 - \left(\min_{p_\nu}\max_{x\in[\alpha,\beta]}|p_\nu(x)|\right)^2\right)^{-1}; \quad \alpha \leq \lambda_{n_c+1} \leq \lambda_n \leq \beta.$$

Proof. Order the eigenvectors in V so that we can write $S = VS_i$, $S_i = [I_s, 0]^T$. Then, since $I - \widetilde{M}^{-1}A = (I-M^{-1}A)(I-M^{-T}A)$, we have

$$\begin{aligned}
S^T\widetilde{M}S &= S^T(A^{-1} - (I-M^{-1}A)A^{-1}(I-M^{-1}A)^T)^{-1}S \\
&= S_i^TV^T(A^{-1} - p_\nu(A)^2A^{-1})^{-1}VS_i \\
&= S_i^T(\Lambda^{-1} - p_\nu(\Lambda)^2\Lambda^{-1})^{-1}S_i \\
&= (\Lambda_s^{-1} - p_\nu(\Lambda_s)^2\Lambda_s^{-1})^{-1}.
\end{aligned}$$

Since $S^T AS = \Lambda_s$, then

$$(S^T \widetilde{M} S)^{-1}(S^T AS) = I_s - p_\nu(\Lambda_s)^2,$$

and the first result follows from Theorem 2.2. The second result follows trivially from the first since we are maximizing over a larger set $[\alpha, \beta]$ containing λ_k, $k > n_c$. □

In the following two subsections, we first discuss the optimal polynomial smoother according to Theorem 4.1 and then briefly overview several other choices of polynomials that may also be good smoothers for AMG in practice.

4.1. Chebyshev Smoothers. The min-max problem in Theorem 4.1 has a classical solution $q_\nu(x)$ in terms of Chebyshev polynomials (see, e.g., [2]). Let $T_k(t)$ be the Chebyshev polynomial of degree k defined by the recursion

$$T_0(t) = 1; \quad T_1(t) = t; \quad T_k(t) = 2tT_{k-1}(t) - T_{k-2}(t), \ k = 2, 3, \ldots \tag{4.2}$$

By letting $t = \cos(\xi) \in [-1, 1]$, it is easy to show that the explicit form of these polynomials is $T_k(t) = \cos(k\xi)$. The polynomial $q_\nu(x)$ is given by

$$q_\nu(x) = \frac{T_\nu\left(\frac{\beta+\alpha-2x}{\beta-\alpha}\right)}{T_\nu\left(\frac{\beta+\alpha}{\beta-\alpha}\right)}, \tag{4.3}$$

and has the required property that $q_\nu(0) = 1$. It also satisfies

$$-1 < q_\nu(x) < 1 \text{ for } x \in (0, \beta],$$

which implies that the smoother (4.1) with $p_\nu = q_\nu$ is convergent as long as the spectrum of A is contained in the interval $(0, \beta]$. To show the above inequality with $\alpha, \beta > 0$ observe that the Chebyshev polynomial $T_\nu(x)$ equals 1 for $x = 1$ and is strictly monotonically increasing for $x > 1$ (see, e.g., (5.28) in [2]). Therefore, $x \in [\alpha, \beta]$ implies $T_\nu\left(\frac{\beta+\alpha}{\beta-\alpha}\right) > 1 \geq \left|T_\nu\left(\frac{\beta+\alpha-2x}{\beta-\alpha}\right)\right|$, while $|q_\nu(x)| < 1$ due to $\frac{\beta+\alpha}{\beta-\alpha} > \frac{\beta+\alpha-2x}{\beta-\alpha} \geq 1$ for $x \in (0, \alpha]$.

Since $K_\star$ is a measure of the smoothing properties of the smoother (4.1), then Theorem 4.1 shows that a good choice for polynomial smoothing is $q_\nu(x)$ where the interval $[\alpha, \beta]$ contains the "large" eigenvalues of A. The upper bound β can easily be estimated using a few iterations of conjugate gradient (CG), but choosing a suitable α is not obvious in general. It is clear that α depends on the coarse-grid size, but it should also depend on the distribution of eigenvalues for the problem and possibly even the nature of the associated eigenvectors. To see this, consider a simple Laplace example on a unit domain discretized by standard finite differences. Assume full coarsening so that $n_c/n = 1/2^d$ where d is the dimension. We discuss three possible choices for α below.

First, note that the analysis above does not require that R be made up of the strictly smallest eigenvectors of A. Consider instead that R contains the smooth Fourier modes used in standard local Fourier analysis. In this case, it is easy to see from standard Fourier diagrams that α should be chosen such that

$$\alpha/\beta = 1/2 \text{ (1D)}, \ 1/4 \text{ (2D)}, \ 1/6 \text{ (3D)}. \tag{4.4}$$

The resulting Chebyshev polynomial smoothers were first derived almost 30 years ago in [35].

Now consider letting R contain the actual n_c smallest eigenvectors for the Laplace equation. Using Matlab, we get the estimates

$$\alpha/\beta \approx 0.5 \text{ (1D)},\ 0.32 \text{ (2D)},\ 0.28 \text{ (3D)}. \tag{4.5}$$

Consider again letting R contain the actual n_c smallest eigenvectors, but assume that the eigenvalues are distributed uniformly. Then, we have

$$\alpha/\beta = 1/2 \text{ (1D)},\ 1/4 \text{ (2D)},\ 1/8 \text{ (3D)}. \tag{4.6}$$

In practice, we set β by estimating $\lambda_{\max}$ with several iterations of CG and set $\alpha = a\beta$ for some fraction $0 \leq a \leq 1$. For example, we use $a = 0.3$ in the numerical experiments. A similar approach is used in [1], but with a small $a = 1/30$. It is not vital to estimate $\lambda_{\min}$ unless it is large. In that case, no coarse grid is needed, and the smoother should damp all eigenvectors equally well, i.e., α should approximate $\lambda_{\min}$.

4.2. Other Polynomial Smoothers. Although the above theory leads naturally to the Chebyshev polynomial in (4.3), there are several other polynomials in the literature that are also good smoothers. We briefly summarize some of the most notable here.

A related smoother to the Chebyshev polynomial in (4.3) is the following shifted and scaled Chebyshev polynomial used in the AMLI method [3]

$$q_\nu^+(x) = \frac{1 + T_\nu\left(\frac{\beta+\alpha-2x}{\beta-\alpha}\right)}{1 + T_\nu\left(\frac{\beta+\alpha}{\beta-\alpha}\right)}. \tag{4.7}$$

This has the required property that $q_\nu^+(0) = 1$, but satisfies

$$0 < q_\nu^+(x) < 1 \text{ for } x \in (0, \beta].$$

Another polynomial smoother of interest is used in both the smoothed aggregation (SA) and cascadic multigrid methods [8, 15, 36], and is given by

$$\phi_\nu(x) = (-1)^\nu \left(\frac{1}{2\nu+1}\right)\left(\frac{\sqrt{\beta}}{\sqrt{x}}\right) T_{2\nu+1}\left(\frac{\sqrt{x}}{\sqrt{\beta}}\right). \tag{4.8}$$

Note that (4.8) does not require the estimation of α. It can be shown that ϕ_ν is the minimizer of

$$\min_{p_\nu} \max_{x\in[0,\beta]} |\sqrt{x}\ p_\nu(x)|. \tag{4.9}$$

One way to interpret (4.9) is to think of the $\sqrt{x}$ term as serving the role of coarse-grid correction, since it is small for small eigenvalues. Because $\sqrt{x}$ operates on the entire eigenvalue interval and not just the smallest eigenvalues (as in true coarse-grid correction), the resulting smoother ϕ_ν does not damp the largest eigenvectors as much as the Chebyshev polynomial in (4.3). Note that it could be modified to satisfy (4.9) over the interval $[\alpha, \beta]$.

The MLS smoother in [1] is the product of ϕ_ν and a complementary (post) smoother of the form

$$I - \frac{\omega}{\lambda_{\max}(\phi_\nu^2 A)}\phi_\nu^2 A.$$

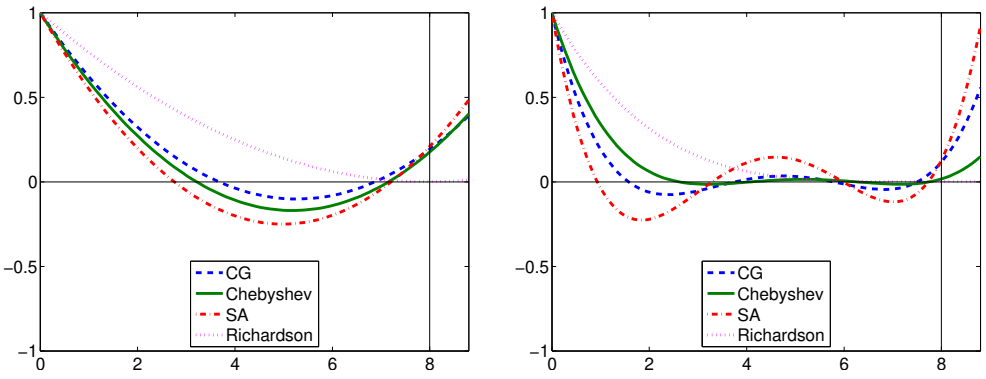

FIG. 4.1. *Various polynomials of order two (left) and four (right). The CG polynomial was generated by solving a 2D Laplace problem on a* 25×25 *grid with a random initial error. The Chebyshev polynomial* (4.3) *uses* $a = 0.3$. *The SA polynomial is given by* (4.8).

It has better overall smoothing properties than ϕ_ν alone, and it is particularly advantageous when using aggressive coarsening.

The polynomial smoother in [30] minimizes an equation like (4.9) over the interval $[\alpha, \beta]$, but with $\sqrt{x}$ in the equation replaced by $1/x$. This means that the amplitude of the polynomial increases over the interval $[\alpha, \beta]$. The polynomial is computed through a three-term recurrence.

The conjugate gradient method is also a good smoother [7]. Note that it converges to the Chebyshev polynomial in (4.3), but over the entire eigenvalue interval $[\alpha, \beta] = [\lambda_{\min}, \lambda_{\max}]$. Even though this is not a good value for α, it is only relevant asymptotically; for small ν, CG has good smoothing properties. Other Krylov methods such as conjugate residual (also called minimum residual or MINRES) typically have good smoothing properties as well [7].

In Figure 4.1 we plot several polynomials over the eigenvalue interval. Focusing on the fourth-order figure, note that the polynomial tails for $x > \beta = 8$ turn up steeply. For this reason, it is important not to underestimate $\lambda_{\max}$ in practice. Note also that the CG polynomial closely approximates Chebyshev. As previously mentioned, the SA polynomial does not damp the large eigenvectors as well as the others. The Richardson polynomial is given by $(I - \lambda_{\max}^{-1})^\nu$. We include it in the figure because it is the simplest smoother to understand and it is used in most classical smoothing analysis. In the interest of keeping the figure readable, we do not plot all of the polynomials in this section. Note, however, that they all have good smoothing properties, with mostly minor differences between them as noted in the text.

5. The Hybrid Smoother. The class of so-called hybrid smoothers can be viewed as the result of the straightforward parallelization of a smoother. For example, the easiest parallelization of GS is to have each process independently use GS on its domain and then exchange information from neighbor processors after each iteration, resulting in a Jacobi-like update at the processor boundaries. As noted in Section 1, hybrid smoothers, like hybrid GS in particular, are of interest because they are easy to implement and often quite effective in practice, even though convergence may not be guaranteed. In this section, we first formally define hybrid smoothers and apply the smoothing analysis theory from Section 2. We then discuss two particular hybrid smoothers, hybrid GS and Block Jacobi, in more detail, and, finally, we discuss the

use of weights with hybrid smoothers.

We define the *hybrid smoother* to be essentially an inexact block Jacobi method. Specifically, let $\Omega = \{1, \ldots, n\}$ and consider the non-overlapping partition of Ω,

$$\Omega = \bigcup_{k=1}^{p} \Omega_k.$$

Of particular practical interest in this paper is the case where Ω_k represents the unknowns on processor k so that p is the total number of processors, but the analysis below is for the general setting. Let A be partitioned into blocks A_{kl} of size $n_k \times n_l$ where the rows of A_{kl} are in Ω_k and the columns are in Ω_l. Let $I - B_k^{-1} A_{kk}$ be a smoother for A_{kk}. Then, the hybrid smoother is defined by

$$I - M_H^{-1} A; \quad M_H = \text{diag}\{B_k\}, \tag{5.1}$$

where $\text{diag}\{B_k\}$ denotes the block-diagonal matrix with blocks B_k.

If $B_k = A_{kk}$, then (5.1) is block Jacobi. As p increases, the convergence of block Jacobi approaches that of pointwise Jacobi. However, although (unweighted) pointwise Jacobi is often not a good smoother, we show below that block Jacobi and other hybrid smoothers can have good smoothing properties independent of p, as long as the blocks are sufficiently large. We also show that this threshold block size can be quite small.

As stated in the beginning of Section 2, we assume that the hybrid smoother is convergent, i.e., that $M_H^T + M_H - A$ is SPD. Assuming that the block smoothers converge in the stronger sense that

$$\langle B_k \mathbf{v}_k,\ \mathbf{v}_k \rangle \geq \langle A_{kk} \mathbf{v}_k,\ \mathbf{v}_k \rangle ,$$

it is relatively easy to show that one class of matrices for which the hybrid smoother is convergent is the class of block red-black matrices, i.e., matrices A that admit the following two-by-two form

$$A = \begin{bmatrix} A_{rr} & A_{rb} \\ A_{br} & A_{bb} \end{bmatrix},$$

with block-diagonal matrices A_{rr} and A_{bb} (see [4] for details). As a practical example of a block red-black matrix, consider a structured (i.e. topologically Cartesian) partitioning of a 5-point discretization in 2D.

To analyze the smoothing properties of the hybrid smoother, we introduce a constant, $\theta \geq 0$, which is a measure of the relative size of the block off-diagonal portion of A. First, define the sets

$$\Omega^{(i)} = \{j \in \Omega_k \,:\, i \in \Omega_k\}; \quad \Omega_o^{(i)} = \{j \notin \Omega_k \,:\, i \in \Omega_k\}. \tag{5.2}$$

Hence, $\Omega^{(i)}$ is the set of columns in the diagonal block for row i while $\Omega_o^{(i)}$ contains the remaining "off-diagonal" columns in row i. Now, with a_{ij} denoting the coefficients of A, define θ such that

$$a_{ii} \geq \theta \sum_{j \in \Omega_o^{(i)}} |a_{ij}| \quad \text{for all rows } i. \tag{5.3}$$

One can think of θ as an indicator of the quality of the parallel partitioning of the matrix, since large values of θ imply that most of the relatively significant entries

of A are contained in the local portion of the matrix on each processor, i.e. the off-processor entries of A are relatively small.

Under typical weak scaling, θ quickly stabilizes to a value independent of the number of processors as the processor topology saturates. In many practical applications this value satisfies $\theta > 1$ (required in the theory below). This, for example, is the case when A is diagonally dominant and each A_{kk} has at least two non-zero entries per row (i.e., the block sizes are large enough). Another example is the structured weak scaling of the 5-point discretization of the Laplacian operator in 2D, where we have $\theta = 2$.

To improve the values of θ one should consider parallel partitionings that group most of the strong connections for each i (relatively large $|a_{ij}|$) inside each processor. In finite element settings, better values for θ are obtained when the blocks correspond to element partitioning (as opposed to random partitioning of the degrees of freedom, see Section 7.3).

5.1. Hybrid Gauss-Seidel. In this section we consider the hybrid Gauss-Seidel smoother M_{HGS}, which is obtained when the blocks B_k in (5.1) are chosen to be Gauss-Seidel sweeps for A_{kk}. This smoother is of practical importance, for example, because it is the default option in the BoomerAMG code.

Let $A = D + L + L^T$, where D is the diagonal of A and L and L^T are its strictly lower and upper triangular parts. We first remark that M_{HGS} is convergent if $\theta > 1$ or if A is red-black both pointwise and blockwise as discussed above. Indeed, the θ condition implies

$$\langle (M_{HGS}^T + M_{HGS} - A)\mathbf{v},\ \mathbf{v}\rangle \geq \frac{\theta - 1}{\theta}\langle D\mathbf{v},\ \mathbf{v}\rangle\,, \tag{5.4}$$

while in the red-black case we have that both regular and block Jacobi are convergent, and therefore

$$\langle (M_{HGS}^T + M_{HGS})\mathbf{v},\ \mathbf{v}\rangle = \sum_k \langle A_{kk}\mathbf{v}_k,\ \mathbf{v}_k\rangle + \langle D\mathbf{v},\ \mathbf{v}\rangle > \langle A\mathbf{v},\ \mathbf{v}\rangle\,.$$

Note that if A has large positive off-diagonal entries, such as in discretizations of definite Maxwell problems, M_{HGS} may be divergent, even for large block sizes. This was the motivation in [29] to develop the ℓ_1 smoothers considered in the next section.

In the next theorem, we compare the smoothing properties of M_{HGS} to that of the standard Gauss-Seidel smoother $M_{GS} = D + L$.

THEOREM 5.1. *Assume that $\theta > 1$. Then*

$$K(M_{HGS}) \leq \frac{\theta}{\theta - 1}\left(1 + \frac{2}{\theta}\right)^2 K(M_{GS}).$$

Proof. Analogous to Theorem 6.2 from the next section, using (5.4) and the fact that

$$\|(M_{HGS} - M_{GS})^T\mathbf{x}\|_{D^{-1}}^2 \leq \frac{1}{\theta^2}\langle D\mathbf{x},\ \mathbf{x}\rangle \leq \frac{4}{\theta^2}\langle \widetilde{M_{GS}}\mathbf{x},\ \mathbf{x}\rangle.$$

□

By Theorem 5.1 we can conclude that hybrid GS is a convergent smoother with smoothing properties comparable to full GS provided that $\theta > 1$ is not close to 1, e.g., if A is diagonally dominant and each block is large enough to have at least two non-zero entries per row.

		$\|E_{TG}\|_A^2$		$K_\star$	
m	p	BJac	HGS	BJac	HGS
512	1	0.00	0.20	1.00	1.25
256	2	0.50	0.32	65.12	1.81
128	4	0.50	0.32	110.62	1.81
32	16	0.51	0.32	418.96	1.81
16	32	0.53	0.32	834.93	1.81
4	128	0.56	0.41	3334.24	1.81
2	256	0.56	0.39	6667.23	2.33
1	512	1.00	1.00	26664.93	26664.93

TABLE 5.1

Convergence factors and constants from Theorem 2.1 for (unweighted) block Jacobi (BJac) and hybrid GS (HGS) for a 1D Laplace problem with m unknowns per block and p blocks.

5.2. Block Jacobi. As mentioned in the beginning of Section 5, the hybrid smoother can be thought of as an inexact block Jacobi method. Since hybrid GS can be shown to have smoothing properties comparable to GS under certain conditions, it seems plausible that (unweighted) block Jacobi might have even better smoothing properties. In fact, block Jacobi is not a particularly good smoother, though it can exhibit smoothing properties independent of the number of blocks (processors).

As an example, consider again a standard Laplace problem on a unit domain with homogeneous Dirichlet boundary conditions. In Table 5.1, we report $\|E_{TG}\|_A^2$ from Theorem 2.1 for the ideal interpolation operator $P_\star$ in (2.9) for a coarsening factor of two in 1D. We also report the corresponding $K_\star$. From the table, we see that hybrid GS is a better smoother than block Jacobi, while both methods appear to have p-independent convergence factors for $m > 1$ (this is easily confirmed by fixing m and increasing p; not shown). At $m = 1$, both methods degenerate into unweighted pointwise Jacobi, which is known to have poor smoothing properties. We also see from the table that $K_\star$ is stable for hybrid GS but unbounded for block Jacobi (additional numerics shows that $K_\star$ depends on both m and p). This implies that the theoretical tools in Section 2 are not adequate for analyzing block Jacobi. Although not the best smoother choice in practice, we would like to get a deeper understanding of block Jacobi's smoothing properties. One approach might be to base the analysis on the sharp theory in [23], but we have not yet pursued this.

The observations from Table 5.1 also carry over to 2D (we have not done 3D experiments), but they are more pronounced. In particular, the convergence factor for $m \geq (2 \times 2)$ approaches 0.76 for block Jacobi instead of 0.56 as in 1D, while hybrid GS stays at 0.39. Another item worth noting is that the convergence of both methods degrades for larger coarsening factors, as one would expect. In addition, for block Jacobi, the minimum block size needed to yield good smoothing properties increases with increasing coarsening factor. It remains the same for hybrid GS as indicated by the theory.

The fact that block Jacobi is not a better smoother than hybrid GS in the above experiments has important implications for developing algorithms on future computer architectures. In particular, the ever-growing gap between local memory and global memory access speeds is leading researchers to consider algorithms that are more local in nature. One idea is to try to improve multigrid convergence by doing more

work locally in the smoother before exchanging data with other processors. From the above, it appears that this may not be straightforward.

5.3. Using Weights in Hybrid Smoothers. While we have shown that for many problems hybrid smoothers converge well, there are various situations where this is not the case, see e.g. Section 7.3. Convergence can be achieved by multiplying M_H with a weight ω as follows:

$$M_\omega = \omega M_H.$$

If M_H is SPD and $\omega = \lambda_{max}(M_H^{-1/2} A M_H^{-1/2})$, we immediately get that M_ω is convergent. In practice, ω can be obtained by the use of Lanczos or CG iterations. For further details on the use of relaxation weights in hybrid smoothers, see [38].

6. The ℓ_1 Smoother. While weighted hybrid smoothers are an attempt to fix hybrid smoothers by multiplying them with a suitable parameter, ℓ_1 *smoothers* do so by adding an appropriate diagonal matrix, which also leads to guaranteed convergence. They have the additional benefit of not requiring eigenvalue estimates. The ℓ_1 *smoother* is defined by

$$I - M_{\ell_1}^{-1} A; \quad M_{\ell_1} = M_H + D^{\ell_1} = \mathrm{diag}\{B_k + D_k^{\ell_1}\}, \tag{6.1}$$

where D^{ℓ_1} is a diagonal matrix with entries

$$d_{ii}^{\ell_1} = \sum_{j \in \Omega_o^{(i)}} |a_{ij}| \,.$$

Note that with this notation (5.3) is simply $D \geq \theta D^{\ell_1}$. Furthermore, D^{ℓ_1} has the important property that

$$\langle A\mathbf{v},\ \mathbf{v}\rangle \leq \sum_k \langle A_{kk}\mathbf{v}_k,\ \mathbf{v}_k\rangle + \langle D^{\ell_1}\mathbf{v},\ \mathbf{v}\rangle \,, \tag{6.2}$$

which follows from the Schwarz inequality $2|a_{ij}v_i v_j| \leq |a_{ij}|v_i^2 + |a_{ij}|v_j^2$.

We first show that M_{ℓ_1} is A-convergent, i.e., that $M_{\ell_1}^T + M_{\ell_1} - A$ is SPD. In the case where $B_k = A_{kk}$, we can actually show more, since (6.2) implies $\langle A\mathbf{v},\ \mathbf{v}\rangle \leq \langle M\mathbf{v},\ \mathbf{v}\rangle$. In general, if the block smoothers B_k are non-divergent in the A_{kk}-norm with at least one of them being convergent, then

$$\langle A_{kk}\mathbf{v}_k,\ \mathbf{v}_k\rangle \leq \langle (B_k^T + B_k)\mathbf{v}_k,\ \mathbf{v}_k\rangle$$

with strict inequality holding for at least one k. Hence, from (6.2),

$$\langle A\mathbf{v},\ \mathbf{v}\rangle < \sum_k \langle (B_k^T + B_k + D_k^{\ell_1})\mathbf{v}_k,\ \mathbf{v}_k\rangle \leq \langle (M_{\ell_1}^T + M_{\ell_1})\mathbf{v},\ \mathbf{v}\rangle.$$

REMARK 6.1. *The following scaled ℓ_1 smoother is also A-convergent:*

$$M_{\ell_1} = diag\{B_k + \frac{1}{2} D_k^{\ell_1}\}.$$

6.1. ℓ_1 Gauss-Seidel. Let $M_{\ell_1 GS} = M_{HGS} + D^{\ell_1}$ be the ℓ_1 GS smoother. This is the default smoother used in the AMS code [29]. From above, this smoother is always convergent, and we analyze its smoothing properties by directly computing the constant in Lemma 2.3. First, we state a Lemma needed to prove Theorem 6.2, but also of general interest.

LEMMA 6.1. *Suppose that A is SPD and B is arbitrary. Then*

$$\langle A\mathbf{x},\ \mathbf{x}\rangle \leq c\langle B\mathbf{x},\ \mathbf{x}\rangle \qquad \textit{implies} \qquad \langle B^{-1}\mathbf{x},\ \mathbf{x}\rangle \leq c\langle A^{-1}\mathbf{x},\ \mathbf{x}\rangle.$$

Proof. Note that by the given inequality B is invertible, and $B^{-1} + B^{-T}$ is SPD. Using Cauchy-Schwarz and the assumption above, we have

$$\begin{aligned}\langle B^{-1}\mathbf{x},\ \mathbf{x}\rangle^2 &= \langle A^{1/2}B^{-1}\mathbf{x},\ A^{-1/2}\mathbf{x}\rangle^2 \\ &\leq \langle AB^{-1}\mathbf{x},\ B^{-1}\mathbf{x}\rangle\langle A^{-1}\mathbf{x},\ \mathbf{x}\rangle \\ &\leq c\ \langle B^{-1}\mathbf{x},\ \mathbf{x}\rangle\langle A^{-1}\mathbf{x},\ \mathbf{x}\rangle.\end{aligned}$$

Dividing both sides by $\langle B^{-1}\mathbf{x},\ \mathbf{x}\rangle$ gives the desired result. ∎

THEOREM 6.2. *Without any restrictions, we have*

$$K(M_{\ell_1 GS}) \leq \left(1 + \frac{4}{\theta}\right)^2 K(M_{GS}).$$

In particular, ℓ_1 GS has multigrid smoothing properties comparable to full GS for any A, for which θ is bounded away from zero, independently of the number of blocks (processors) or the block sizes.

Proof. First, note that Lemma 6.1 implies

$$\langle M\mathbf{x},\ \mathbf{x}\rangle \leq 2\langle \widetilde{M}\mathbf{x},\ \mathbf{x}\rangle\,, \tag{6.3}$$

for any convergent smoother M, since from (2.3) and A SPD, we have

$$\langle \widetilde{M}^{-1}\mathbf{x},\ \mathbf{x}\rangle = \langle (M^{-1} + M^{-T} - M^{-1}AM^{-T})\mathbf{x},\ \mathbf{x}\rangle \leq 2\langle M^{-1}\mathbf{x},\ \mathbf{x}\rangle.$$

Now, observe that $\langle M_{\ell_1 GS}\mathbf{x},\ \mathbf{x}\rangle \geq \langle M_{GS}\mathbf{x},\ \mathbf{x}\rangle$, which implies

$$\langle D\mathbf{x},\ \mathbf{x}\rangle \leq \langle (M_{\ell_1 GS}^T + M_{\ell_1 GS} - A)\mathbf{x},\ \mathbf{x}\rangle.$$

Therefore,

$$\langle \widetilde{M_{\ell_1 GS}}\mathbf{x},\ \mathbf{x}\rangle = \langle (M_{\ell_1 GS}^T + M_{\ell_1 GS} - A)^{-1}M_{\ell_1 GS}\mathbf{x},\ M_{\ell_1 GS}\mathbf{x}\rangle \leq \|M_{\ell_1 GS}\mathbf{x}\|_{D^{-1}}^2.$$

By the triangle inequality in the D^{-1}-inner product,

$$\|M_{\ell_1 GS}\mathbf{x}\|_{D^{-1}} \leq \|M_{GS}\mathbf{x}\|_{D^{-1}} + \|(M_{\ell_1 GS} - M_{GS})\mathbf{x}\|_{D^{-1}}.$$

The first term above is simply $\langle \widetilde{M_{GS}}\mathbf{x},\ \mathbf{x}\rangle^{1/2}$, while the second one can be estimated using the Schwarz inequality and (6.3) as follows:

$$\begin{aligned}
\|(M_{\ell_1 GS} - M_{GS})\mathbf{x}\|^2_{D^{-1}} &= \sum_i \frac{1}{a_{ii}} \left(\sum_{j\in\Omega_o^{(i)}} |a_{ij}|x_i + \sum_{j<i\,,\, j\in\Omega_o^{(i)}} a_{ij}x_j \right)^2 \\
&\le \sum_i \frac{1}{a_{ii}} \left(2 \sum_{j\in\Omega_o^{(i)}} |a_{ij}| \right) \left(\sum_{j\in\Omega_o^{(i)}} |a_{ij}|x_i^2 + \sum_{j<i\,,\, j\in\Omega_o^{(i)}} |a_{ij}|x_j^2 \right) \\
&\le \frac{2}{\theta} \sum_i \left(\sum_{j\in\Omega_o^{(i)}} |a_{ij}| + \sum_{j>i\,,\, j\in\Omega_o^{(i)}} |a_{ij}| \right) x_i^2 \\
&\le \frac{4}{\theta^2}\langle D\mathbf{x},\ \mathbf{x}\rangle \le \frac{8}{\theta^2}\langle M_{GS}\mathbf{x},\ \mathbf{x}\rangle \le \frac{16}{\theta^2}\langle \widetilde{M_{GS}}\mathbf{x},\ \mathbf{x}\rangle.
\end{aligned}$$

The desired bound now follows by assembling the above estimates together. Note that in the last line we derived the inequality $\langle D\mathbf{x},\ \mathbf{x}\rangle \le 4\langle \widetilde{M_{GS}}\mathbf{x},\ \mathbf{x}\rangle$, which has appeared previously in [39], Lemma 3.3 and in [36], Proposition 6.12. □

REMARK 6.2. *The scaled variant of ℓ_1 GS based on Remark 6.1 is given by $M_{\ell_1 GS} = M_{HGS} + \frac{1}{2}D^{\ell_1}$. Theorem 6.2 also holds for this smoother.*

Another convergent option, which also takes advantage of the local estimation of θ in (5.3) is

$$M_{\ell_1 GS*} = M_{HGS} + D^{\ell_1 *}, \qquad \text{where} \qquad d_{ii}^{\ell_1 *} = \begin{cases} 0, & \text{if } a_{ii} \ge \eta d_{ii}^{\ell_1}; \\ d_{ii}^{\ell_1}/2, & \text{otherwise.} \end{cases} \tag{6.4}$$

and η is a fixed parameter satisfying $\eta > 1$. This smoother locally switches to ℓ_1 GS if hybrid GS is not appropriate, and we have found that the value $\eta = 1.5$ works well in practice. Some results with this smoother are shown in Section 7. Note that the smoother $M_{\ell_1 GS*}$ is identical to M_{HGS} when $\theta \ge \eta$, and reduces to the scaled version of $M_{\ell_1 GS}$ from Remark 6.2 when θ is uniformly small relative to η. Furthermore, the general conclusion of Theorem 6.2 still holds for $M_{\ell_1 GS*}$, since $\eta = 1.5$, for example, implies

$$\langle D\mathbf{x},\ \mathbf{x}\rangle \le 3\langle (M_{\ell_1 GS*}^T + M_{\ell_1 GS*} - A)\mathbf{x},\ \mathbf{x}\rangle.$$

6.2. ℓ_1 Jacobi. The ℓ_1 point Jacobi smoother is given by $M_{\ell_1 J} = D + D^{\ell_1}$ and is a special case of the ℓ_1 GS smoother when the blocks are of size one. Although the analysis in Theorem 6.2 holds for $M_{\ell_1 J}$, a slightly better constant is straightforward to derive [4].

7. Numerical Experiments. In this section, we discuss numerical experiments that complement the smoothing analysis and illustrate the effect of smoothers on AMG performance. First we describe our four test problems. Then we present results using a two-level AMG method. Next we discuss the potential impact of using threading in smoothers, which is particularly relevant for modern large-scale parallel computing. Finally, we present multilevel results for Conjugate Gradient (CG) preconditioned with AMG on up to 32,000 processors.

For the results in this section, we use a modification of the BoomerAMG code in the *hypre* software library [26] . We use HMIS coarsening [18] and extended+i(5) [17]

interpolation in all cases. For the coarse-grid solve, we use CG for the two-level results in Section 7.2, and Gaussian elimination for all multilevel results (coarse grids contain at most nine unknowns in this case). We use a relative convergence tolerance of 10^{-6} and report the number of AMG (or CG/AMG) iterations (as opposed to timings) because our focus here is on the effect of the smoother on AMG convergence. Finally, we note that the implementation of the Chebyshev polynomial smoother uses A scaled by its diagonal D, i.e. $p_\nu(D^{-1/2}AD^{-1/2})$, c.f. (4.1).

While we experimented with various smoothers, we present here only those results we deemed to be the most interesting. For polynomial smoothers, we chose the Chebyshev polynomial, since it is the optimal polynomial according to the theory. We only present results for 2nd degree polynomials. While the use of polynomials with larger degrees (3 or 4) led to better convergence, the gain in convergence was not sufficient to offset the increase in work required. We also don't report any results for weighted hybrid smoothers, since they performed similarly to ℓ_1 smoothers, which do not require eigenvalue estimates.

7.1. Problem Descriptions. We use four test problems for our numerical experiments. The first three problems are scalar diffusion problems of the form

$$-\nabla \cdot (a(x,y,z)\nabla u) = f,$$

and discretized on unstructured grids with the aFEM finite element package, which has been used previously in [28, 29, 14]. The resulting matrices have some positive off-diagonal entries and are not diagonally dominant. The fourth problem is a simple 3D Laplace problem on a structured grid and is an M-matrix.

2D square*:* This is a 2D problem posed on the unit square with $a = 1$ discretized with linear triangular elements as shown on the left in Figure 7.1. Boundary conditions are homogeneous Dirichlet on the right and left edge.

2D LLNL*:* This is a 2D problem on the unit square with four material subdomains resembling the Lawrence Livermore National Laboratory (LLNL) logo. We have $a(x,y) = 1$ in the inner three domains, $a(x,y) = 10^{-3}$ in the outer domain, and homogeneous Dirichlet boundary conditions. Linear triangular elements are used as shown in Figure 7.1.

3D sphere*:* This 3D diffusion problem is discretized on a domain approximating the unit ball with trilinear hexahedral finite elements. It contains 2 "random" material subdomains, with $a(x,y,z) = 1$ and $a(x,y,z) = 10^3$, see Figure 7.2. Boundary conditions are homogeneous Dirichlet.

7pt Laplacian*:* This is the 3D Laplace problem with $a = 1$, homogeneous Dirichlet boundary conditions, and discretized with a constant 7-point stencil on a structured meshing of the unit cube.

7.2. Two-level Results. In this section we provide results using a two-level AMG method to corroborate the smoothing analysis in the previous sections. We consider both strong and weak scaling.

For strong scaling, the problem size is fixed and the number of processors varies. Therefore, for a fixed problem size n, as the number of processors increases, the number of unknowns per processor decreases, and we affect the value of θ in (5.3). Strong scaling results for all four test problems are shown in Figures 7.3 and 7.4 for up to 4096 processors for four smoothers: hybrid GS, a C-F smoother using ℓ_1 Jacobi ('CF-L1-Jacobi'), ℓ_1 GS ('L1-GS'), and second order Chebyshev ('Cheby(2)'). The left y-axis indicates the number of (two-level) AMG iterations with each smoother.

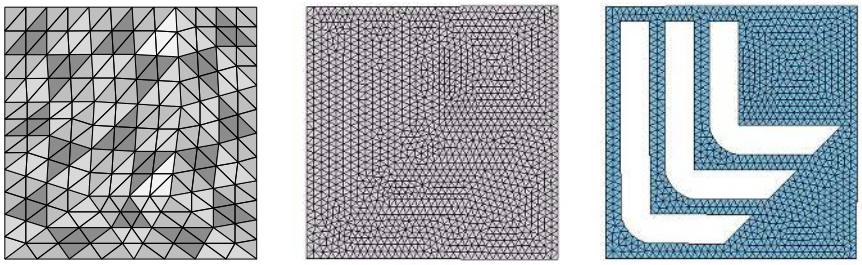

FIG. 7.1. *Coarse versions of the unstructured meshes used for the 2D square and 2D LLNL problems. The 2D square mesh is on the left, while the 2D LLNL mesh is shown in the center with the three material subdomains indicated on the right.*

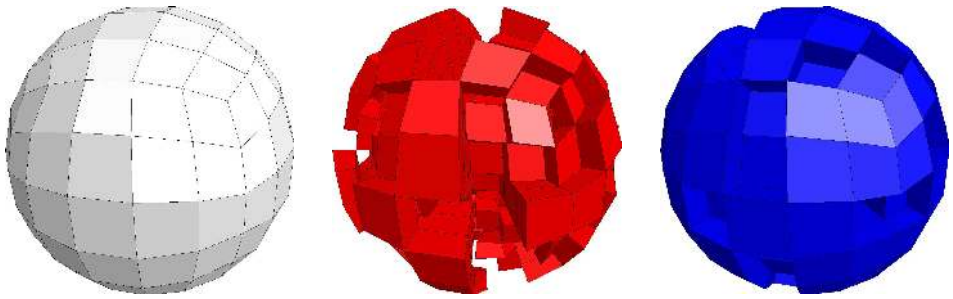

FIG. 7.2. *A coarse version of the unstructured mesh used for the 3D sphere problem, with the two material subdomains shown on the right.*

The dotted line in the figures corresponds to θ, and its value is indicated on the right y-axis. We chose the problem sizes n to be relatively small to better illustrate the effect of strong scaling on θ.

The 2D LLNL and 7pt Laplacian problems are particularly interesting as one can clearly see the relation between a decrease in θ and the deterioration of hybrid GS convergence. For the 3D sphere, AMG with hybrid GS does not converge except for the single processor case, and, correspondingly, θ is much less than one (note that θ is not defined in the single processor case, so the first data point in the figures for θ is at 32 processors). While 2D square and 2D LLNL problems also have θ less than one, the percentage of rows for which θ is less than one is much smaller than for the 3D sphere problem. For example, for the 2D square with 64 processors, θ is less than one for only a single row of the matrix. In practice, one could calculate θ during the AMG setup phase and use it to determine whether hybrid GS would be an appropriate smoother, or one should switch to ℓ_1 GS globally or locally as in (6.4). Note that the corresponding ℓ_1 row norms can be obtained in the process of calculating θ. For comparison, results for the ℓ_1 GS smoother from (6.4) are included for the 2D LLNL problem in Figure 7.3 (and later in Figure 7.8) and labeled 'L1-GS*'.

For weak scaling, the number of unknowns per processor remains constant as the number of processors increases. Therefore, the value of θ remains essentially constant. Weak scaling results are shown for the 7pt Laplacian problem in Figure 7.5 for 8 and 1000 unknowns per processor, both of which correspond to a $\theta = 2$, and, therefore, yield relatively flat iteration counts for hybrid GS.

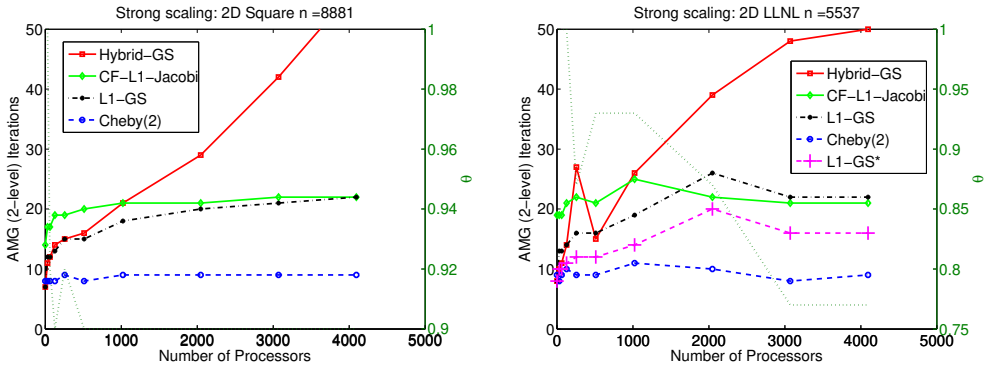

FIG. 7.3. *Two-level strong scaling results for the 2D square (left) and the 2D LLNL problem (right). (The global problem size is fixed and indicated by n.)*

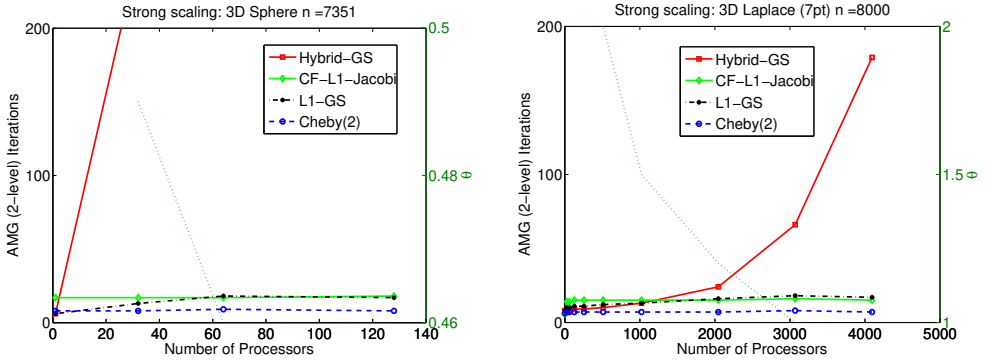

FIG. 7.4. *Two-level strong scaling results for the 3D sphere (left) and the 7pt Laplacian (right). (The global problem size is fixed and indicated by n.)*

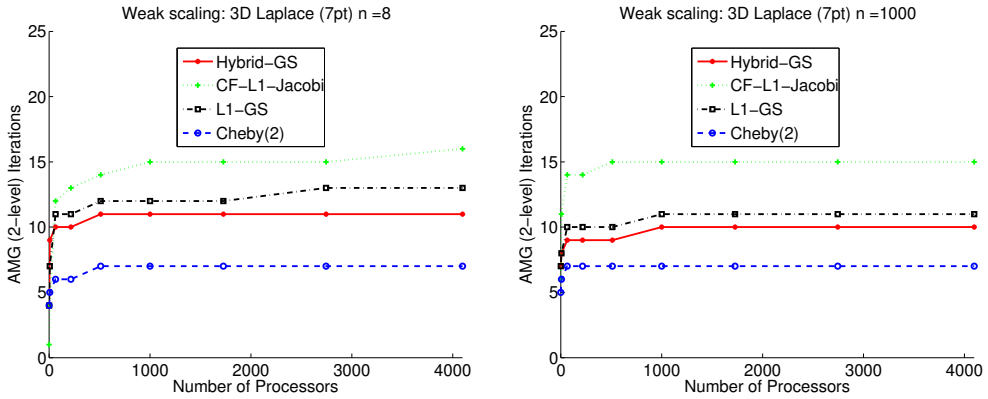

FIG. 7.5. *Two-level weak scaling for the 7pt Laplacian for $n = 8$ (left) and $n = 1000$ (right) unknowns per processor.*

7.3. Impact of Threading. Achieving scalability on modern multicore machines requires effective utilization of the new node architectures, which generally consist of multiple cores and sockets per node, cache sharing, multiple memory controllers, and non-uniform memory access times. An MPI-only programming model may not be well-suited to all multicore machine architectures (e.g., see [6]). Instead, a hybrid programming model, in which a subset of cores on a node (or all cores) use a shared memory programming model (like OpenMP) and MPI is used between nodes, can be more appropriate. Therefore, we are interested in smoothers that are amenable to a hybrid parallel programming model.

First we briefly describe the parallelism for the smoothers in the BoomerAMG code. To begin, the matrix A is distributed row-wise across p MPI processes, such that each process owns a contiguous block of rows that are stored in a parallel compressed sparse row (CSR) format (see [20] for details). This parallel storage format is similar to what many packages use. OpenMP is implemented at the loop level for the smoothers such that each thread operates on a subset of the process' rows. Therefore, for a polynomial smoother, the OpenMP implementation of the matrix-vector multiply is straightforward, and, similarly, Jacobi smoothers are easily threaded. However, for the hybrid GS smoother, we must modify the algorithm at the thread level in the same manner as the MPI implementation: we use GS within each thread and delayed updates (Jacobi) for those elements belonging to other threads.

To demonstrate the (negative) effect that threading may have on convergence, we show results obtained on the multicore cluster Hera at LLNL. Each node of Hera consists of four AMD Quadcore processors, each with its own memory controller that is attached to one quarter of the main memory on the node, and the nodes are connected by Infiniband. We use BoomerAMG as a preconditioner for CG ('AMG-CG'), and use Symmetric GS (SGS). The plots in Figure 7.6 give results for the unstructured 2D LLNL and 3D sphere problems on up to 256 nodes (4096 cores) of Hera. These are weak scaling results, so the problem size per core is approximately 87,000 and 57,000 for the 2D LLNL and 3D sphere problems, respectively. Note that the 2D square problem is not shown as its behavior is similar to that of the 2D LLNL problem, and the 7pt Laplacian is not shown as it is structured and unaffected by threading. The MPI-only version uses 16 MPI tasks per node, while the hybrid model we consider uses 16 threads and 1 MPI task per node (labeled 'OMP'). Figure 7.6 clearly illustrates that convergence degrades with the addition of threads for hybrid SGS; the convergence of the ℓ_1 SGS smoothers ('L1-SGS'), ℓ_1 Jacobi (not shown), and Chebyshev smoothers (not shown) are relatively unaffected. The 3D sphere problem is the most extreme example because AMG-CG with hybrid SGS no longer converges with the addition of threading.

To understand why convergence suffers with the addition of threads, we must consider the application's generation of the matrix. An AMG solver is typically part of a parallel software library, such as *hypre*, and the library receives the matrix A in a distributed manner: each process is given a subset of the rows of A. Note that aFEM utilizes the Metis library [27] to partition a refinement of a given coarse mesh into "nice" subdomains for each processor. These subdomains are further refined in parallel to obtain approximately the same size per processor for weak scaling. Unlike in the structured case, this process can lead to slight variations in the values of θ in weak scaling. Figure 7.7 gives an example of the 2D square problem subdivided for 16 processes. However, when BoomerAMG uses threads, then the rows given to each MPI task are further subdivided among the threads. Because BoomerAMG does not

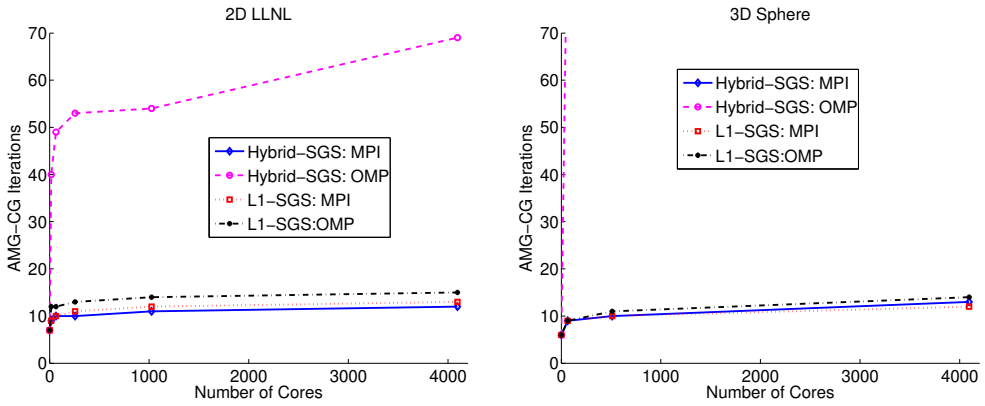

FIG. 7.6. *A comparison of the number of AMG-CG iterations to convergence for the 2D LLNL (left) and 3D sphere (right) problems with hybrid SGS smoothing and ℓ_1 SGS smoothing. 'MPI' = 16 tasks per node, 'OMP' = 16 threads per node.*

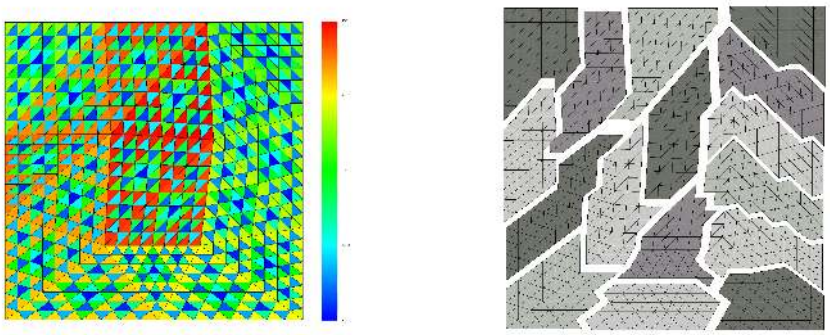

FIG. 7.7. *2D square problem partitioned into 16 subdomains for 16 MPI tasks (right). The coloring in the left figure indicates element numbering obtained through refinement. Note that the induced numbering of the degrees of freedom in each processor will be essentially random.*

know anything about the original elements, the thread subdomains are split unknown-wise (not element-wise), and aFEM's numbering of the elements and nodes becomes relevant. For the example in Figure 7.7, the coloring indicates the element numbering, which means that similarly-colored elements are given to the same thread. A domain partitioning such as this is detrimental for hybrid GS, as the smoother essentially becomes (unweighted) Jacobi. In other words, the blocks of the hybrid smoother will correspond to disconnected points, and the value of θ will deteriorate significantly. We stress that this example is not atypical for finite element applications: element numbering is often assigned as a coarser grid is refined, and requiring that the resulting unknowns are ordered "nicely" within processor subdomains imposes a non-trivial burden on the application. We note that if the FEM application is careful with the numbering within each MPI subdomain, then one can obtain similarly good results with both MPI and OpenMP. Alternatively, BoomerAMG could reorder unknowns, but reordering can be costly and memory consuming operation. The most logical option, then, is to prefer smoothers that do not depend on parallelism, such as the ℓ_1 and the polynomial smoothers.

7.4. Multilevel Weak Scaling. Now we consider the relative scalability of BoomerAMG on larger numbers of processors with the various smoothers. Experiments were run using MPI-only on the BG/L machine at LLNL for the 2D square, 2D LLNL, and 7pt Laplacian problems, and on Hera for the 3D sphere problem. We use BoomerAMG as a preconditioner for CG. Results for the 2D square and 2D LLNL problems are given in Figure 7.8 for up to 16384 processors. The iteration counts are fairly flat, and second order Chebyshev requires the fewest iterations. Hybrid SGS converges faster than ℓ_1 SGS, but this can be mitigated by the symmetric GS variant of (6.4) ('L1-SGS*') that is shown for the LLNL problem (the Hybrid-SGS and L1-SGS* lines are on top of each other). Results for the 3D sphere and 7pt Laplacian are given in Figure 7.9. The sphere could not be run on as many processors due to memory requirements for partitioning. Here also Cheby(2) exhibits the best convergence. While ℓ_1 SGS converges slower (or at least not faster) than Hybrid SGS for the smaller problem sizes as in the 2D case, this changes for the large problem sizes, where ℓ_1 SGS takes fewer iterations.

While the focus of our numerical experiments is not on timings, we do make a few comments in that regard. For the Chebyshev smoothers, eigenvalue estimates must be obtained during the AMG setup phase. As in [1], we use 10 iterations of CG and multiply λ_{max} by 1.1, and this strategy appears to be sufficient. The time required to calculate the eigenvalues is not a significant portion of the setup phase and is justifiable, at least for these test problems, given that the Chebyshev smoother required the fewest iterations. Furthermore, note that one can estimate the largest eigenvalue of A by using the maximum absolute row-sum of A (infinity norm), which, of course, is cheaper than performing 10 CG iterations. This strategy is equally effective for most problems, though often less so on the coarser grid levels. Finally, for the four smoothers shown here, the time per iteration is roughly similar because each essentially requires two sweeps over the coefficients of A. SGS requires a forward and backward sweep, second order Chebyshev requires two matrix-vector multiplies, and the C-F Jacobi must sweep through the C-points and the F-points.

Although the timings in our current implementation are roughly the same per multigrid iteration, it is worth making a few observations about the work done. Define a *work unit* to be the number of scalar operations required to do a matrix-vector multiply, and consider Figure 7.5. The Cheby(2) smoother does 4 work units of smoothing per multigrid iteration, while the Hybrid-GS and L1-GS smoothers do 2. The CF-L1-Jacobi only does 1.5 real work units of smoothing per multigrid iteration, because we used C-F before coarse-grid correction and F-C after. This means that C-relaxation is repeated twice between each multigrid iteration, with no additional benefit. The overall numbers of iterations in the figure somewhat reflect these work differences. A similar statement can be made about Figure 7.8 and Figure 7.9, except that Cheby(2), Hybrid-SGS, and L1-SGS all take 4 work units per multigrid iteration, while CF-L1-Jacobi again only does 1.5. Although CF-L1-Jacobi does much less real work, our current implementation does not reflect this in timings. One issue is that one sweep of C-F requires either two communications (our current implementation) or one communication with more data together with some replicated computations. Another issue involves ordering the computations in such a way that the data only flows through the cache once. These optimizations are difficult to achieve, but have already been demonstrated for red-black GS (see, e.g., [19]).

8. Concluding Remarks. In this paper we reviewed and analyzed a number of practical parallel multigrid smoothers, and evaluated their potential for scalability

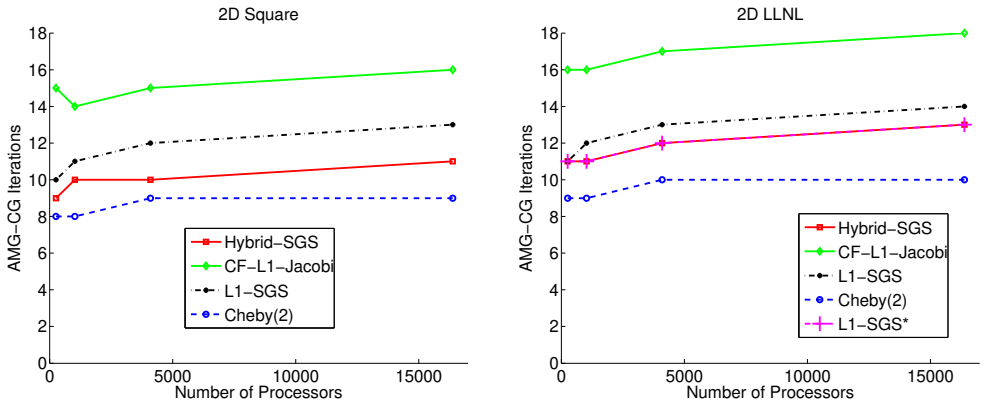

FIG. 7.8. *Weak scaling on up to 16384 processors for the 2D square (left) and 2D LLNL problem (right) with approximately 35,000 and 87,000 unknowns per processor, respectively.*

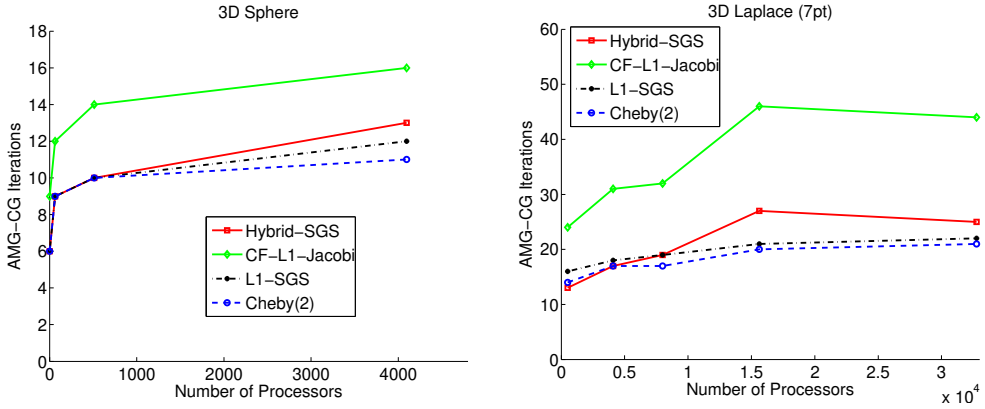

FIG. 7.9. *Weak scaling on up to 4096 processors for the 3D sphere with approximately 57,000 unknowns per processor (left) and for the 7pt Laplacian problem on up to 32768 processors with 125,000 unknowns per processor (right). Note that the upper limits for the y-axis are different for the two plots.*

on ultra-parallel computers with millions of processors.

Based on the framework from [22, 23] we proposed both a direct (Theorem 2.2) and a comparative (Lemma 2.3) approach for smoother analysis. Using these approaches, we showed that C-F smoothing is good if F-relaxation is fast to converge (Theorem 3.1), that Chebyshev is the optimal polynomial smoother (Theorem 4.1), and that hybrid Gauss-Seidel exhibits multigrid smoothing properties which are independent of the number of processors in many practical applications, e.g. if the matrix is diagonally dominant and the problem size per processor is large enough (Theorem 5.1). For the more robust ℓ_1 smoothers described in Section 6, we were able to prove processor-independent equivalence with full Gauss-Seidel with minimal restrictions on the matrix (Theorem 6.2).

The numerical results in Section 7 demonstrate that both Chebyshev and the ℓ_1 smoothers are robust with respect to weak scaling, strong scaling, and the use of mixed MPI / OpenMP parallelism. This is encouraging news for the scalability of the corresponding AMG algorithms on the next generation parallel machines. Polynomial

smoothers are especially appealing for their lack of dependence on the ordering of the unknowns and reliance on an often finely-tuned matrix-vector multiply kernel. Hybrid Gauss-Seidel is also a viable option, provided that the constant θ, which measures the relative size of the off-processor entries according to (5.3), is strictly larger than 1.

Acknowledgments. We thank Panayot Vassilevski for his helpful advice, and an anonymous referee whose comments led to a much improved paper.

REFERENCES

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro, *Parallel multigrid smoothing: Polynomial versus Gauss-Seidel*, J. Comput. Phys., 188 (2003), pp. 593–610. 1, 2, 8, 21

[2] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, 1996. 7

[3] O. Axelsson and P. Vassilevski, *Algebraic multilevel preconditioning methods I*, Numer. Math., 56 (1989), pp. 157–177. 8

[4] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang, *Multigrid smoothers for ultra-parallel computing: Additional theory and discussion*, Tech. Rep. LLNL-TR-xxxxxx, Lawrence Livermore National Laboratory, March 2011. 5, 10, 15

[5] A. H. Baker, T. Gamblin, M. Schulz, and U. M. Yang, *Challenges of scaling algebraic multigrid across modern multicore architectures*, in Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2011), 2011. To appear. Also available as LLNL Tech. Report LLNL-CONF-458074. 1

[6] A. H. Baker, M. Schulz, and U. M. Yang, *On the performance of an algebraic multigrid solver on multicore clusters*, in VECPAR 2010, J.M.L.M. Palma et al., ed., vol. 6449 of Lecture Notes in Computer Science, Springer-Verlag, 2011, pp. 102–115. http://vecpar.fe.up.pt/2010/papers/24.php. 19

[7] R. E. Bank and C. C. Douglas, *Sharp estimates for multigrid rates of convergence with general smoothing and acceleration*, SIAM J. Numer. Anal., 22 (1985), pp. 617–633. 2, 9

[8] F. A. Bornemann and P. Deuflhard, *The cascadic multigrid method for elliptic problems*, Numer. Math., 75 (1996), pp. 135–152. 8

[9] D. Braess, *The convergence rate of a multigrid method with Gauss–Seidel relaxation for the Poisson equation*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., vol. 960 of Lecture Notes in Mathematics, Berlin, 1982, Springer-Verlag, pp. 368–386. 3

[10] D. Braess and W. Hackbusch, *A new convergence proof for the multigrid method including the V cycle*, SIAM J. Numer. Anal., 20 (1983), pp. 967–975. 3

[11] J. H. Bramble, *Multigrid Methods*, vol. 294 of Pitman Research Notes in Mathematical Sciences, Longman Scientific & Technical, Essex, England, 1993. 3

[12] J. H. Bramble and X. Zhang, *The analysis of multigrid methods*, Handb. Numer. Anal., VII (2000), pp. 173–415. 3

[13] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390. 2, 3, 4

[14] J. J. Brannick and R. D. Falgout, *Compatible relaxation and coarsening in algebraic multigrid*, SIAM J. Sci. Comput., 32 (2010), pp. 1393–1416. LLNL-JRNL-417122. 6, 16

[15] M. Brezina, C. Heberton, J. Mandel, and P. Vaněk, *An iterative method with convergence rate chosen a priori*, Tech. Rep. UCD CCM Report 140, Center for Computational Mathematics, University of Colorado at Denver, February 1999. 8

[16] E. Chow, R. Falgout, J. Hu, R. Tuminaro, and U. Yang, *A survey of parallelization techniques for multigrid solvers*, in Parallel Processing for Scientific Computing, M. Heroux, P. Raghavan, and H. Simon, eds., SIAM Series on Software, Environments, and Tools, SIAM, 2006, ch. 10. 1

[17] H. De Sterck, R. D. Falgout, J. Nolting, and U. M. Yang, *Distance-two interpolation for parallel algebraic multigrid*, Num. Lin. Alg. Appl., 15 (2008), pp. 115–139. 15

[18] H. De Sterck, U. M. Yang, and J. Heys, *Reducing complexity in algebraic multigrid preconditioners*, SIMAX, 27 (2006), pp. 1019–1039. 15

[19] C. C. Douglas, J. Hu, M. Kowarschik, U. Rüde, and C. Weiss, *Cache optimization for structured and unstructured grid multigrid*, Elect. Trans. Numer. Anal., 10 (2000), pp. 21–40. 21

[20] R. Falgout, J. Jones, and U. M. Yang, *Pursuing scalability for hypre's conceptual interfaces*, ACM ToMS, 31 (2005), pp. 326–350. 19

[21] R. D. Falgout, *An introduction to algebraic multigrid*, Computing in Science and Engineering, 8 (2006), pp. 24–33. Special issue on Multigrid Computing. UCRL-JRNL-220851. 1

[22] R. D. Falgout and P. S. Vassilevski, *On generalizing the algebraic multigrid framework*, SIAM J. Numer. Anal., 42 (2004), pp. 1669–1693. UCRL-JC-150807. 1, 2, 3, 4, 5, 6, 22

[23] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov, *On two-grid convergence estimates*, Numer. Linear Algebra Appl., 12 (2005), pp. 471–494. UCRL-JRNL-203843. 1, 2, 5, 12, 22

[24] W. Hackbusch, *Multi-grid convergence theory*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., vol. 960 of Lecture Notes in Mathematics, Springer-Verlag, 1982, pp. 177–219. 2, 3

[25] V. Henson and U. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155–177. 1

[26] *hypre: High performance preconditioners*. http://www.llnl.gov/CASC/hypre/. 1, 15

[27] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392. 19

[28] T. Kolev and P. Vassilevski, *AMG by element agglomeration and constrained energy minimization interpolation*, Numer. Linear Algebra Appl., 13 (2006), pp. 771–788. UCRL-JC-219462. 16

[29] ———, *Parallel auxiliary space AMG for H(curl) problems*, J. Comput. Math., 27 (2009), pp. 604–623. Special issue on Adaptive and Multilevel Methods in Electromagnetics. UCRL-JRNL-237306. 1, 11, 14, 16

[30] J. Kraus, V. Pillwein, and L. Zikatanov, *Algebraic multilevel iteration methods and the best approximation to $1/x$ in the uniform norm*, Tech. Rep. 2009-17, Johann Radon Institute for Computational and Applied Mathematics (RICAM), 2010. http://arxiv.org/abs/1002.1859v1. 9

[31] J. Mandel, S. F. McCormick, and J. W. Ruge, *An algebraic theory for multigrid methods for variational problems*, SIAM J. Numer. Anal., 25 (1988), pp. 91–110. 3

[32] S. F. McCormick, *Multigrid methods for variational problems: further results*, SIAM J. Numer. Anal., 21 (1984), pp. 255–263. 3

[33] ———, *Multigrid methods for variational problems: general theory for the V–cycle*, SIAM J. Numer. Anal., 22 (1985), pp. 634–643. 3

[34] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1987, pp. 73–130. 3

[35] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., vol. 960 of Lecture Notes in Mathematics, Springer-Verlag, 1982, pp. 1–176. 2, 7

[36] P. S. Vassilevski, *Multilevel block factorization preconditioners: Matrix-based analysis and algorithms for solving finite element equations*, Springer, New York, 2008. 8, 15

[37] U. Yang, *Parallel algebraic multigrid methods - high performance preconditioners*, in Numerical Solution of Partial Differential Equations on Parallel Computers, A. Bruaset and A. Tveito, eds., vol. 51, Springer-Verlag, 2006, pp. 209–236. 1

[38] U. M. Yang, *On the use of relaxation parameters in hybrid smoothers*, Numer. Linear Algebra Appl., 11 (2004), pp. 155–172. UCRL-JC-151575. 1, 2, 13

[39] L. T. Zikatanov, *Two-sided bounds on the convergence rate of two-level methods*, Numer. Linear Algebra Appl., 15 (2008), pp. 439–454. 15