Compatible Coarsening in the Multigraph Algorithm

Randolph E. Bank*

October 13, 2005

Abstract

We present some heuristics incorporating the philosophy of compatible relaxation into an existing algebraic multigrid method, the so-called multigraph solver of Bank and Smith [1]. In particular, approximate left and right eigenvectors of the iteration matrix for the smoother are used in computing both the sparsity pattern and the numerical values of the transfer matrices that correspond to restriction and prolongation. Some numerical examples illustrate the effectiveness of the procedure.

1 Introduction

The modern era of research on algebraic multilevel methods was inspired by Brandt and co-workers [2, 3, 4, 5]. See Wagner [6] and Stüben [7] for recent surveys of the field. Although research on algebraic multilevel methods is now quite widespread, one of the largest current efforts is centered at Lawrence Livermore National Laboratory, in the Center for Applied Scientific Computing and their collaborators [8, 9, 10, 11, 12].

Multilevel algorithms typically consist of smoothing steps interlaced in some way with coarse grid correction steps. The coarse grid correction can be further decomposed in terms of restriction and prolongation operations, which define the mappings between two adjacent levels, and the coarse mesh operator itself. The algebraic multilevel solver described here evolved from the multigraph algorithm described in [1]. The work presented here is related to the construction of the restriction and prolongation operators (transfer matrices) in the coarse grid correction. This involves both the selection of the coarse graph points, and the determination of the restriction and interpolation coefficients (the numerical values of matrix elements in the transfer matrices).

*Department of Mathematics University of California, San Diego La Jolla, California 92093-0112. email:rbank@ucsd.edu. The work of this author was supported by the National Science Foundation under contract DMS-0208449. The UCSD Scicomp Beowulf cluster was built using funds provided by the National Science Foundation through SCREMS Grant 0112413, with matching funds from the University of California at San Diego.

The goal of *compatible relaxation*, or compatible coarsening as it is called here, is to construct a coarse grid correction that complements the smoother in the sense that it provides good error reduction for components of the error where the smoother is least effective. Such algorithms typically are heuristic, because of the need to keep the coarse grid correction inexpensive to compute, and also inexpensive to apply. The basic idea of compatible relaxation is not new; it has been widely studied, although not always under the name compatible relaxation, and there are many competing ideas. See, for example [13, 14, 15, 16, 17, 18, 19, 20, 8, 10] for some alternative approaches. Some unusual features of the algorithms described here include the use of local least squares to (approximately) interpolate certain test vectors, and the ability to specify the size of the coarse system a priori, through the use of a (user-specified) coarsening factor ρ.

The remainder of this paper is organized as follows. In Section 2, we provide a brief summary of the entire multigraph algorithm. In Section 3, we consider the issue of computing numerical values for the transfer matrices given their sparsity pattern. In Section 4, we describe our heuristic for computing the sparsity pattern. In both sections, left and right eigenvectors of the iteration matrix for the smoother are approximated by a few iterations of the power method. These approximate eigenvectors form the basis of our compatible coarsening heuristics. See [21, 22, 23] for some related ideas on frequency filtering. In Section 5, we provide some numerical illustrations illustrating the procedures. The multigraph solver is implemented in Fortran 77. The source code is available as part of the $PLTMG$ 9.0 software package [24], and also separately as a stand-alone solver for sparse linear systems of equations [25].

2 Matrix formulation

Let A be a large sparse, nonsingular $N \times N$ matrix. We assume that the sparsity pattern of A is symmetric, although the numerical values need not be. We will begin by describing the basic two-level method for solving

$$Ax = b. \tag{1}$$

Let B be an $N \times N$ nonsingular matrix, called the *smoother*, which gives rise to the basic iterative method used in the multilevel preconditioner. In our case, B is an approximate factorization of A, i.e.,

$$B = (L + D)D^{-1}(D + U) \approx P^t AP, \tag{2}$$

where L is (strict) lower triangular, U is (strict) upper triangular with the same sparsity pattern as L^t, D is diagonal, and P is a permutation matrix. The matrix B is computed via an ILU factorization of $P^t AP$, using a user specified drop tolerance to control the fill-in. The permutation matrix P is determined by applying the minimum degree algorithm to the graph of the matrix A, and the ILU itself is modeled on sparse Gaussian elimination techniques [26]. Additional details regarding the smoother may be found in [1]. Other approaches to ILU in multilevel algorithms can be found in [27, 28, 29, 30, 31]. We remark that our compatible coarsening algorithms apply to more general smoothers, and do not rely on the specific choice of smoother made here. For simplicity, henceforth in this section we assume $P = I$.

Given an initial guess x_0, m steps of the smoothing procedure produce iterates x_k, $1 \leq k \leq m$, given by

$$x_k = x_{k-1} + B^{-1}(b - Ax_{k-1}). \tag{3}$$

The second component of the two-level preconditioner is the *coarse grid correction.* We classify existing unknowns as either *fine* or *coarse.* This is represented algebraically by a permutation matrix $\hat{P}$ such that

$$\hat{P}A\hat{P}^t = \begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix}, \tag{4}$$

where the subscripts f and c denote fine and coarse, respectively. The $\hat{N} \times \hat{N}$ coarse grid matrix $\hat{A}$ is given by

$$\begin{aligned} \hat{A} &= \begin{pmatrix} V_{cf} & I_{cc} \end{pmatrix} \begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix} \begin{pmatrix} W_{fc} \\ I_{cc} \end{pmatrix} \\ &= V_{cf}A_{ff}W_{fc} + V_{cf}A_{fc} + A_{cf}W_{fc} + A_{cc}. \end{aligned} \tag{5}$$

The matrices V_{cf} and W_{fc}^t are $\hat{N} \times (N - \hat{N})$ matrices with identical sparsity patterns; thus $\hat{A}$ has a symmetric sparsity pattern. If $A^t = A$, we require $V_{cf} = W_{fc}^t$, so $\hat{A}^t = \hat{A}$.

Let

$$\hat{V} = \begin{pmatrix} V_{cf} & I_{cc} \end{pmatrix} \hat{P}, \qquad \hat{W} = \hat{P}^t \begin{pmatrix} W_{fc} \\ I_{cc} \end{pmatrix}. \tag{6}$$

In standard multigrid terminology, the matrices $\hat{V}$ and $\hat{W}$ are called *restriction* and *prolongation*, respectively. Given an approximate solution x_m to (1), an *exact* coarse grid correction for computing an iterate x_{m+1} is defined as follows.

$$\begin{aligned} \hat{r} &= \hat{V}(b - Ax_m), \\ \hat{A}\hat{\delta} &= \hat{r}, \\ x_{m+1} &= x_m + \hat{W}\hat{\delta}. \end{aligned} \tag{7}$$

A two level iteration consists of m pre-smoothing steps, followed by an exact coarse grid correction, followed by m post-smoothing steps. Since $\hat{A}\hat{\delta} = \hat{r}$ is of the same form as our original problem (1), the two-level iteration may be generalized to the multilevel case by applying recursion to the solution of the equation $\hat{A}\hat{\delta} = \hat{r}$ in (7). In this work (and in our code) we restrict attention to just the symmetric V-cycle with $m = 1$ pre-smoothing and post-smoothing iterations. To simplify notation we now use subscripts to denote level; for example, A_J now denotes to level J matrix and B_J the level J smoother. For the coarsest mesh solution ($J = 1$), our procedure is somewhat nonstandard; instead of a direct solution of (1), we compute an approximate solution using $m = 1$ smoothing iterations. We illustrate the practical consequences of this decision in Section 5. Here we summarize our recursive V-cycle procedure for approximately solving (1) on level J.

```
procedure V-cycle($J$, $x$, $b$)
    if $J = 1$, then
        $x \leftarrow x + B_J^{-1}(b - A_J x)$.
    else
        $x \leftarrow x + B_J^{-1}(b - A_J x)$.
        $\hat{b} \leftarrow \hat{V}_J(b - A_J x)$; $\hat{x} \leftarrow 0$.
        V-cycle($J - 1$, $\hat{x}$, $\hat{b}$).
        $x \leftarrow x + \hat{W}_J \hat{x}$.
        $x \leftarrow x + B_J^{-1}(b - A_J x)$.
    end if
end V-cycle
```

If A is symmetric, then so is the iteration matrix implicitly defined by our V-cycle, and thus it can be used as a preconditioner for a symmetric Krylov space method. If A is also positive definite, so is our preconditioner, and the standard conjugate gradient method could be used; otherwise the CSCG method [32], SYMLQ [33], or a similar method could be used. In the nonsymmetric case, the V-cycle preconditioner could be used in conjunction with the CSBCG method [32], GMRES [34], or a similar method.

In this work, we develop algebraic procedures for constructing the prolongation and restriction matrices $\hat{V}$ and $\hat{W}$ of (6). There are three major issues that must be addressed:

- First, one must determine the block structure of these matrices; this involves choosing which unknowns are *coarse* and which are *fine*. This reduces to determining the permutation matrix $\hat{P}$ of (4).
- Second, one must determine how coarse and fine unknowns are related, the so-called *parent-child relations* [6]. This involves computing the sparsity patterns for the matrices V_{cf} and W_{fc}.
- Third, one must compute the numerical values for these matrices, the so-called *interpolation coefficients* [13].

Computing the permutation matrix $\hat{P}$ is deferred until Section 4, while the other two issues are addressed in Section 3. Other recent approaches to coarsening can be found in [35, 13, 14, 15, 16, 17, 18, 19, 20, 8, 10].

3 Computing the transfer matrices

Here we consider the construction of the transfer matrices W_{cf} and V_{fc}, given the fine-coarse partition defined by the permutation matrix $\hat{P}$ of (4). For convenience and to simplify notation, in this section we assume that $\hat{P} = I$ and that the block structure given in (4) is known.

Our algorithm uses the vectors $\hat{w}$ and $\hat{v}$ defined as follows. Let w_0 and v_0 be given and let

$$\begin{aligned} w_k &= (I - B^{-1}A)w_{k-1}, \\ v_k &= (I - B^{-t}A^t)v_{k-1}, \end{aligned}$$

for $1 \leq k \leq \ell$, and then set

$$\begin{aligned} \hat{w} &= w_\ell / \|w_\ell\|, \\ \hat{v} &= v_m / \|v_m\|. \end{aligned}$$

When $A^t = A$, we compute only w_k, and in the practical implementation we normalize the v_k and w_k. Typically we take $w_0 = v_0 = e$, the vector of ones, and choose ℓ to be a small fixed integer such as $\ell = 3$. The vectors $\hat{w}$ and $\hat{v}$ are (possibly crude) approximations to eigenvectors corresponding to the spectral radii of the error propagation matrices for A and A^t generated by a few steps of the power method. Similar approaches have been employed by many authors; we mention in particular adaptive filtering techniques [21, 22, 23].

Intuitively, $\hat{w}$ and $\hat{v}$ are vectors that we want to approximate well in the coarse subspace. In particular, $\hat{w}$ should be in the column space of $\hat{W}$. Similarly $\hat{v}$ should be in the column space of $\hat{V}^t$. This implies that

$$\begin{aligned} \hat{w}_f &= W_{fc}\hat{w}_c, \\ \hat{v}_f &= V_{cf}^t\hat{v}_c, \end{aligned} \tag{8}$$

where $\hat{w}_f$ and $\hat{w}_c$ are the fine and coarse blocks of the vector $\hat{w}$. In other words, (8) indicates that W_{fc} should be chosen to *interpolate* $\hat{w}$ exactly. We note that if sparsity patterns of W_{fc} and V_{cf} are known, but the numerical values are unknown, then (8) can be interpreted as simple linear constraints on coefficients for each row of the matrix.

In our procedure, we usually take the sparsity pattern of W_{fc} to be the same as A_{fc}. We initially choose $\tilde{V}_{cf}$ and $\tilde{W}_{fc}$ according to the formulae

$$\begin{aligned} \tilde{W}_{fc} &= -R_{ff}D_{ff}^{-1}A_{fc}, \\ \tilde{V}_{cf} &= -A_{cf}D_{ff}^{-1}\tilde{R}_{ff}. \end{aligned} \tag{9}$$

Here D_{ff} is a diagonal matrix with diagonal entries equal to those of A_{ff}. In this sense, the nonzero entries in $\tilde{V}_{cf}$ and $\tilde{W}_{fc}$ are chosen as multipliers in Gaussian elimination. The nonnegative diagonal matrices R_{ff} and $\tilde{R}_{ff}$ are chosen such that nonzero rows of W_{fc} and columns of V_{cf}, respectively, have unit norms in ℓ_1.

This initial choice is then modified via least squares to satisfy (8). Here we will describe our procedure for computing a single row of W_{fc}. We suppose that this row has $K \geq 2$ nonzeros. Let $\tilde{r}^t$ be the compressed row vector of size K containing only the nonzero entries. Let z be the corresponding vector of coefficients in $\hat{w}_c$, and let u be a vector with entries ± 1 with signs chosen such that $\tilde{r}^t u = 1$, reflecting (9).

The final vector $r = \tilde{r} + \delta$ is chosen to satisfy

$$\begin{aligned} r^t u &= 1, \\ r^t z &= \beta, \end{aligned} \tag{10}$$

where β is the appropriate entry from $\hat{w}_f$. Equations (10) represent two linear constrains on the coefficients of δ. The first constraint (weakly) controls both the signs and the magnitudes of the entries, while the second constraint imposes the condition that W_{fc} should be chosen to *interpolate* $\hat{w}$ exactly. Thus (10) can be interpreted as a system of two linear equations for the $K \geq 2$ coefficients of δ; in the typical case, this system will be under determined. The classical least squares (generalized inverse) solution for this under determined system is formally given by

$$r = \tilde{r} + \begin{pmatrix} u & z \end{pmatrix} \begin{pmatrix} u^t u & u^t z \\ u^t z & z^t z \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \beta - \tilde{r}^t z \end{pmatrix},$$

which is easily computed by solving a 2×2 linear system. In practice, we must check to be sure the matrix is sufficiently well conditioned, and if not, take $\delta = 0$.

4 Choosing the fine-coarse partition

We now consider the selection of coarse unknowns. In our procedure, we compute a target value N_{trgt} for the size of the coarse set $\hat{N}$ as $N_{trgt} = \lfloor N/\rho \rfloor + 1$, and try to achieve a coarse set with $\hat{N} \leq N_{trgt}$. The parameter $\rho > 1$ is user specified; a typical value would be $\rho = 4$ for discretizations of partial differential equations in two space dimensions. An initial guess for the coarse set is provided by the coarsening algorithm used in the previous version of the multigraph code, based on the well-known reverse Cuthill–McKee algorithm, a sparse matrix ordering technique that tends to yield matrices with minimal bandwidth [36]. Once the vertices have been ordered in this fashion, we can make a breadth-first search to compute a maximal independent set (in the graph theory sense), which becomes our initial guess for the coarse set. A more detailed description of this part of our coarsening procedure can be found in [1].

Once a reasonable independent set has been computed as our initial guess, it is iteratively updated based on the approximate eigenvectors $\hat{v}$ and $\hat{w}$ and the constrained approximate eigenvectors $\bar{v}$ and $\bar{w}$ computed by

$$\begin{aligned}\tilde{w}_k &= E_c(I - B^{-1}A)E_c\tilde{w}_{k-1},\\ \tilde{v}_k &= E_c(I - B^{-t}A^t)E_c\tilde{v}_{k-1},\end{aligned}$$

for $1 \le k \le \ell$, with

$$\begin{aligned}\bar{w} &= \tilde{w}_\ell/\|w_\ell\|,\\ \bar{v} &= \tilde{v}_\ell/\|v_\ell\|.\end{aligned}$$

Here E_c is a diagonal matrix with zero diagonal entries for coarse unknowns and ones otherwise. The initial guess is taken as E_ce, where as before, e is the vectors of ones. Note also that $\bar{w}$ and $\bar{v}$ are normalized consistently with $\hat{v}$ and $\hat{w}$. Overall, this calculation provides an impression of the behavior of the smoother on the complement of the current coarse space. The calculation of $\bar{v}$ and $\bar{w}$ is similar to, and inspired by, the compatible relaxation approaches used by the AMG group at Lawrence Livermore National Laboratory [9, 11].

Based on these approximate eigenvectors, we compute

$$\begin{aligned}\hat{s}_i &= \max_{j\in\{i\}\cup adj(i)}\{|\hat{v}_j| + |\hat{w}_j|\},\\ \bar{s}_i &= |\bar{v}_i| + |\bar{w}_i|.\end{aligned}$$

Using $\hat{s}_i$ and $\bar{s}_i$ we define some heuristics for changing the status of a given vertex from fine to coarse or coarse to fine. If a given fine vertex is changed to coarse, we assume that the effect is to reduce the current value of $\bar{s}_i$ to zero. On the other hand, if a current coarse vertex is changed to fine, we assume the effect is to increase zero to $\hat{s}_i$.

Our iterative procedure uses a threshold parameter $\theta = \theta_0 \max_k\{|\hat{v}_k|+|\hat{w}_k|\}$, with $0 < \theta_0 < 1$, and a bias parameter $\eta > 1$; typical values are $\theta_0 = .1$ and $\eta = 10$. Coarse vertices are ordered according to their values of $\hat{s}_i$ and fine vertices are ordered according to their values of $\bar{s}_i$. Given an existing fine-coarse partition, we update the partition depending on the value of $\hat{N}$.

Case 1: $\hat{N} > N_{trgt}$. In this case, we reduce the number of coarse unknowns to N_{trgt}, choosing the least harmful based on the ordering of the $\hat{s}_i$.

Case 2: $\hat{N} < N_{trgt}$. In this case, we first try to reduce the number of coarse unknowns, changing to fine any coarse unknown that satisfies $\hat{s}_i\eta < \theta$. We then change from fine to coarse as many fine unknowns as possible that satisfy $\bar{s}_i > \theta$. For this, we use the ordering based on $\bar{s}_i$, and stop when $\hat{N} = N_{trgt}$ or $\bar{s}_i \le \theta$.

Case 3: $\hat{N} = N_{trgt}$. In this case we try to simultaneously change fine unknowns to coarse, and coarse unknowns to fine, maintaining $\hat{N} = N_{trgt}$. The fine unknown with largest $\bar{s}_i$ is compared with the coarse unknown with the smallest $\hat{s}_j$. If $\hat{s}_j\eta < \bar{s}_i$ an exchange is made. The algorithm continues through the ordered lists, exchanging pairs of coarse and fine unknowns until $\hat{s}_j\eta \ge \bar{s}_i$.

We apply this update procedure to the independent set partition provided by the reverse Cuthill-McKee ordering. For the first iteration, we typically have Case 1 or Case 2. We then make a second iteration, using the updated coarse–fine partition and new approximate eigenvectors $\bar{v}$ and $\bar{w}$. For this second iteration, we typically have Case 3, although Case 2 is also possible.

Finally, the coarsened matrix $\hat{A}$ of (5) is computed and possibly *sparsified* using the user specified drop tolerance to remove small off-diagonal elements. Empirically, applying a drop tolerance to $\hat{A}$ at the end of the coarsening procedure has proved more efficient, and more effective, than trying to independently sparsify its constituent matrices.

5 Numerical experiments

In this section, we present a few numerical illustrations. Some of these experiments reprise those given in [1], which document the behavior of the original multigraph program, and [28], which describes an earlier algebraic hierarchical basis algorithm. These experiments were run on a Linux dual Xeon 3.06GHz workstation, using double precision arithmetic and the g77 compiler. The multigraph solver, subroutines to generate the structured matrix examples, and data files containing the unstructured matrix examples can be found at http://cam.ucsd.edu/~reb.

In our first sequence of experiments, we consider several matrices loosely based on the classical case of 5-point centered finite difference approximations to $-\Delta u$ on a uniform square mesh. Dirichlet boundary conditions are imposed. This leads to the $n \times n$ block tridiagonal system

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix}$$

with T the $n \times n$ tridiagonal matrix

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}.$$

This is a simple test problem easily solved by standard multigrid methods. In contrast to this example we also consider the block tridiagonal system

$$\bar{A} = 8I - A.$$

Both A and $\bar{A}$ have the same eigenvectors and the same eigenvalues, although the association of eigenvectors and eigenvalues are reversed in the case of $\bar{A}$.

That is, the so-called *smooth* eigenvectors are associated with large eigenvalues, while *rough* eigenvectors are associated with smaller eigenvalues. Although $\bar{A}$ does not arise naturally in the context of numerical discretizations of partial differential equations, it is of interest because it defies much of the conventional wisdom for multigrid methods.

Third, we consider block 3×3 systems of the form

$$S = \begin{pmatrix} A & 0 & C_x \\ 0 & A & C_y \\ C_x^t & C_y^t & -D \end{pmatrix},$$

where A is the discrete Laplacian and D is a symmetric positive definite "stabilization" matrix with a sparsity pattern similar to A. However, the nonzeros in D are of size $O(h^2)$, compared to size $O(1)$ nonzero elements in A. C_x and C_y also have sparsity patterns similar to that of A, but these matrices are nonsymmetric and their nonzero entries are of size $O(h)$. Such matrices arise in stabilized discretizations of the Stokes equations. About one third of the eigenvalues of S are negative, so S is quite indefinite.

In Table 1, *Levels* refers to the number of levels used in the calculation. The drop tolerance was set to $\epsilon = 10^{-2}$ for all matrices. The coarsening parameter was set to $\rho = 4$. The initial guess for all problems was $x_0 = 0$.

In Table 1, the parameter *Digits* refers to

$$Digits = -\log \frac{\|r_k\|}{\|r_0\|}. \tag{11}$$

In these experiments, we asked for six digits of accuracy. The column labeled *Cycles* indicates the number of multigrid cycles (accelerated by CSCG) that were used to achieve the indicated number of digits. Finally, the last two columns, labeled *Init.* and *Solve*, record the CPU time, measured in seconds, for the initialization and solution phases of the algorithm, respectively. Initialization includes all the orderings, incomplete factorizations, and computation of transfer matrices used in the multigraph preconditioner. Solution includes the time to solve (1) to at least six digits given the preconditioner.

In analyzing these results, it is clear that our procedure does reasonably well on all three classes of matrices. Although it appears that the rate of convergence is not independent of N, it seems apparent that the work is growing no faster than logarithmically. CPU times for larger values of N are affected by cache performance as well as by the larger number of cycles. Times for some smaller values of N were below the resolution of the Linux utility *etime* and were reported as zero.

For the highly indefinite Stokes matrices S, it is important to also note the robustness, that the procedure solved all of the problems. With more nonzeros per row on average, the incomplete factorization was more expensive to compute than for the other cases. This is reflected in relatively larger initialization and solve times.

In our next experiment, we illustrate the effect of the drop tolerance ϵ and the number of levels. For the 5-point matrix A with $N = 160000$, we solved

Table 1: Performance comparison.

n	N	Levels	Digits	Cycles	Init.	Solve
Discrete Laplacian A, $\epsilon = 10^{-2}$						
10	100	3	6.1	2	0.0e 0	0.0e 0
20	400	4	6.9	3	1.0e-2	0.0e 0
40	1600	5	7.1	4	3.0e-2	0.0e 0
80	6400	7	7.2	5	1.1e-1	2.0e-2
160	25600	8	7.4	6	4.7e-1	1.4e-1
320	102400	9	6.1	5	2.3e 0	6.6e-1
$\bar{A} = 8I - A$, $\epsilon = 10^{-2}$						
10	100	3	6.9	2	0.0e 0	0.0e 0
20	400	4	8.3	3	1.0e-2	0.0e 0
40	1600	5	7.0	3	2.0e-2	1.0e-2
80	6400	7	6.9	3	1.0e-1	1.0e-2
160	25600	8	6.3	3	4.5e-1	7.0e-2
320	102400	9	6.1	3	2.4e 0	3.8e-1
Stokes matrix S, $\epsilon = 10^{-2}$						
10	300	4	6.8	2	1.0e-2	0.0e 0
20	1200	5	8.4	3	4.0e-2	0.0e 0
40	4800	6	7.8	3	2.6e-1	4.0e-2
80	19200	7	6.8	6	1.4e 0	2.6e-1
160	76800	8	6.3	8	7.1e 0	1.9e 0

the problem for $\epsilon = 10^{-k}$, $1 \le k \le 3$, and $1 \le levels \le 7$. We terminated the iteration when the solution had six digits, as measured by (11). We also provide the total non zeros for the system matrices on all levels ($\sum |A|$) and the corresponding approximate LDU factorizations ($\sum |U|$), measured in thousands of entries.

Here we see that our method behaves in a very predictable way. In general, decreasing the drop tolerance or increasing the number of levels improves the convergence behavior of the method. We note that, unlike the classical multigrid method, where the coarsest matrix is solved exactly, in our code we have chosen to approximately solve the coarsest system using just one smoothing iteration using the incomplete factorization. When the maximum number of levels are used, as in Table 1, the smallest system is typically 1×1 or 2×2, and this is an irrelevant remark. However, in the case of Table 2, the fact that the smallest system is not solved exactly significantly influences the overall rate of convergence. This is why, unlike methods where the coarsest system is solved

Table 2: Dependence of convergence on ϵ and levels: discrete Laplacian A, $N = 160000$, $\rho = 4$.

ϵ	levels	Digits	Cycles	Init.	Solve	$\sum \lvert A \rvert$	$\sum \lvert U \rvert$
	1	6.0	362	0.9	23.4	479	643
	2	6.0	116	2.3	15.2	678	802
	3	6.1	52	2.7	7.4	727	850
10^{-1}	4	6.1	30	2.8	4.3	740	861
	5	6.2	15	2.8	2.2	743	864
	6	6.6	10	2.8	1.6	743	864
	7	6.1	9	2.8	1.5	743	864
	1	6.0	110	1.2	8.0	479	1236
	2	6.1	43	3.0	6.5	678	1662
	3	6.1	23	3.6	3.9	730	1769
10^{-2}	4	6.0	12	3.6	2.2	744	1794
	5	6.2	7	3.7	1.5	748	1800
	6	6.8	6	3.7	1.3	749	1801
	7	7.0	6	3.7	1.3	749	1801
	1	6.0	38	1.7	3.2	479	1999
	2	6.2	17	4.2	3.1	678	2818
	3	6.3	9	4.8	2.1	729	3019
10^{-3}	4	7.2	5	4.9	1.3	743	3065
	5	7.3	5	4.9	1.1	747	3075
	6	7.4	5	4.9	1.2	748	3076
	7	7.4	5	4.9	1.2	748	3077
≈ 0	1	11.2	1	8.1	0.2	479	5626

exactly, increasing the number of levels often improves the rate of convergence.

We also include in Table 2 the case $\epsilon = 0$, sparse Gaussian elimination. (In fact, our code uses $\mu\|A\|$ as the drop tolerance (μ is the machine epsilon) when the user specifies $\epsilon = 0$ to avoid dividing by zero.) Here we see that Gaussian elimination is reasonably competitive on this problem. However, we generally expect the initialization cost for $\epsilon = 0$ to grow like $O(N^{3/2})$. For *level* $= 1$ and $\epsilon > 0$, we expect the solution times to grow like $O(N^p)$, $p > 1$. For the best multilevel choices, we expect both initialization and solution times to behave like $O(N) - O(N \log N)$.

In our next example, we consider the effect of the coarsening factor ρ for the 7-point discrete Laplacian on the unit cube. With a $50 \times 50 \times 50$ mesh, we have $N = 125000$. In Table 3 we present data rates for $\rho = 2, 4, 8, 16, 32, 64$

Table 3: Dependence of convergence on ρ: three dimensional discrete Laplacian A, $N = 125000$, $\epsilon = 10^{-2}$.

ρ	levels	Digits	Cycles	Init.	Solve	$\sum \lvert A \rvert$	$\sum \lvert U \rvert$
	2	6.1	17	3.9	2.3	507	1234
64	3	6.1	17	3.9	2.3	507	1234
	4	6.1	17	3.9	2.3	507	1234
	2	6.3	17	4.0	2.3	521	1257
32	3	6.3	17	4.0	2.3	522	1258
	4	6.3	17	4.0	2.3	522	1258
	2	6.0	15	4.4	2.2	551	1307
16	3	6.5	16	4.5	2.4	554	1313
	4	6.5	16	4.5	2.4	555	1313
	2	6.0	12	5.4	1.8	614	1407
	3	6.2	12	5.6	1.9	639	1431
8	4	6.2	12	5.6	1.9	643	1433
	5	6.2	12	5.6	1.9	643	1433
	6	6.2	12	5.6	1.9	643	1434
	2	6.3	11	6.7	1.9	744	1635
	3	6.9	9	7.3	1.9	834	1739
	4	6.9	9	7.6	1.9	857	1765
4	5	6.8	9	7.6	1.9	863	1771
	6	6.8	9	7.6	1.9	864	1772
	7	6.8	9	7.6	1.9	865	1772
	8	6.8	9	7.6	1.9	865	1772
	2	6.2	14	6.6	3.8	1092	2020
	3	6.0	11	9.2	3.3	1502	2453
	4	6.2	9	10.9	3.1	1741	2672
	5	6.5	8	11.6	2.9	1872	2782
	6	6.3	7	12.1	2.7	1946	2838
2	7	6.6	7	12.4	2.7	1990	2866
	8	6.8	7	12.8	2.7	2014	2880
	9	6.9	7	13.0	2.7	2028	2888
	10	6.0	6	13.3	2.4	2036	2892
	11	6.0	6	13.5	2.4	2041	2894
	12	6.0	6	13.5	2.4	2043	2896
ϵ	levels	Digits	Cycles	Init.	Solve	$\sum \lvert A \rvert$	$\sum \lvert U \rvert$
10^{-2}	1	6.1	26	2.8	2.0	493	1213
10^{-6}	1	7.0	3	131.0	1.4	493	28573

and several values of levels. For a three dimensional problem, $\rho = 8$ roughly corresponds to the case of uniform refinement in the case of classical multigrid methods. For comparative purposes, we also present data for the case of 1 level and $\epsilon = 10^{-2}, 10^{-6}$; we were unable to compute the case $\epsilon \approx 0$ due to the very large storage requirements for the triangular factors. Here we see behavior similar to the two dimensional case, although generally there is much more fill-in for an equivalent choice of ϵ.

In our final series of tests, we study the convergence of the method for a suite of test problems generated from the finite element code $PLTMG$ [24]. These example problems were presented in our earlier work [28], where a more complete description of the problems, as well as numerical results for our hierarchical basis multigraph method and the classical AMG algorithm of Ruge and Stüben [4], can be found. As a group, the problems feature highly nonuniform, adaptively generated meshes, relatively complicated geometry, and a variety of differential operators. For each test case, both the sparse matrix and the right-hand side were saved in a file to serve as input for the iterative solvers. A short description of each test problem is given below.

Problem Superior. This problem is a simple Poisson equation

$$-\Delta u = 1$$

with homogeneous Dirichlet boundary conditions on a domain in the shape of Lake Superior. This is the classical problem on a fairly complicated domain. The solution is generally very smooth but has some boundary singularities.

Problem Hole. This problem features discontinuous, anisotropic coefficients. The overall domain is the region between two concentric circles, but this domain is divided into three subregions. On the inner region, the problem is

$$-\delta \Delta u = 0$$

with $\delta = 10^{-2}$. In the middle region, the equation is

$$-\Delta u = 1,$$

and in the outer region the equation is

$$-u_{xx} - \delta u_{yy} = 1.$$

Homogeneous Dirichlet boundary conditions are imposed on the inner (hole) boundary, homogeneous Neumann conditions on the outer boundary, and the natural continuity conditions on the internal interfaces. While the solution is also relatively smooth, singularities exist at the internal interfaces.

Problem Texas. This is an indefinite Helmholtz equation

$$-\Delta u - 2u = 1$$

posed in a region shaped like the state of Texas. Homogeneous Dirichlet boundary conditions are imposed. The length scales of this domain are roughly 16×16, so this problem is fairly indefinite.

Problem UCSD. This is a simple constant coefficient convection-diffusion equation

$$-\nabla \cdot (\nabla u + \beta u) = 1,$$

$\beta = (0, 10^5)^T$ posed on a domain in the shape of the UCSD logo. Homogeneous Dirichlet boundary conditions are imposed. Boundary layers are formed at the bottom of the region and the top of various obstacles.

Problems Jcn 0 *and Jcn* 180. The next two problems are solutions of the current continuity equation taken from semiconductor device modeling. This equation is a convection-diffusion equation of the form

$$-\nabla \cdot (\nabla u + \beta u) = 0,$$

$\beta = 0$ in most of the rectangular domain. However, in a curved band in the interior of the domain, $|\beta| \approx 10^4$ and is directed radially. Dirichlet boundary conditions $u = 10^{-5}$ and $u = 10^{10}$ are imposed along the bottom boundary and along a short segment on the upper left boundary, respectively. Homogeneous Neumann boundary conditions are specified elsewhere. The solutions vary exponentially across the domain which is typical of semiconductor problems.

In the first problem, Jcn 0, the convective term is chosen so the device is *forward biased.* In this case, a sharp internal layer develops along the top interface boundary. In the second problem, Jcn 180, the sign of the convective term is reversed, resulting in two sharp internal layers along both interface boundaries.

We summarize the results in Table 4. As before, perhaps the most important point is that the method solved all of the problems. While convergence rates are not independent of h, once again the growth appears to be at worst logarithmic. We did not attempt to optimize the rate of convergence by varying the parameters. We took $\rho = 4$ for all problems and chose the drop tolerance as in [1] to allow the tables to be comparable. In [1], ϵ was chosen to obtain similar numbers of cycles for all problems, and this seems to also be true in the present case.

Acknowledgment

Much of this work was inspired by discussions with the algebraic multigrid group in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory. I am especially indebted to Rob Falgout, Charles Tong and Panayot Vassilevski.

References

[1] Randolph E. Bank and R. Kent Smith. An algebraic multilevel multigraph algorithm. *SIAM J. on Scientific Computing*, 25:1572–1592, 2002.

Table 4: Performance comparison. $\rho = 4$ for all cases.

N	Levels	Digits	Cycles	Init.	Solve
Superior, $\epsilon = 10^{-3}$					
5k	6	7.4	3	8.0e-2	1.0e-2
20k	7	7.3	5	5.1e-1	1.1e-1
80k	8	6.4	7	3.5e 0	1.2e 0
Hole, $\epsilon = 10^{-4}$					
5k	6	6.4	3	1.3e-1	2.0e-2
20k	7	8.2	4	7.7e-1	1.5e-1
80k	8	7.2	6	4.9e 0	1.6e 0
Texas, $\epsilon = 10^{-5}$					
5k	6	6.3	1	1.2e-1	1.0e-2
20k	7	7.8	2	8.4e-1	9.0e-1
80k	8	6.8	4	6.2e 0	1.1e 0
UCSD, $\epsilon = 10^{-3}$					
5k	6	11.9	2	7.0e-2	2.0e-2
20k	7	11.3	2	4.5e-1	1.0e-1
80k	6	10.4	2	2.9e 0	6.7e 0
Jcn 0, $\epsilon = 10^{-4}$					
5k	6	11.9	2	1.6e-2	3.0e-2
20k	7	11.8	2	8.2e-1	1.6e-1
80k	8	9.1	2	5.2e 0	1.1e 0
Jcn 180, $\epsilon = 10^{-5}$					
5k	5	11.9	2	1.7e-1	3.0e-2
20k	6	7.7	2	9.1e-1	1.8e-1
80k	8	7.2	3	5.9e 0	1.5e 0

[2] Achi Brandt, Steve McCormick, and Juhn Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Technical report, Institute for Computational Studies, Colorado State University, Fort Collins CO, 1982.

[3] Achi Brandt, Steve McCormick, and Juhn Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In *Sparsity and Its Applications (D. J. Evans, ed.)*. Cambridge University Press, Cambridge, UK, 1984.

[4] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers Applied Math.*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[5] J. E. Dendy. Black box multigrid. *J. Comput. Phys.*, 48:366–386, 1982.

[6] Christian Wagner. Introduction to algebraic multigrid. Technical report, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen,Universität Heidelberg, 1999.

[7] K. Stüben. A review of algebraic multigrid. *J. Comput. Appl. Math.*, 128(1-2):281–309, 2001.

[8] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. Algebraic multigrid based on element interpolation (AMGe). *SIAM J. Sci. Comput.*, 22(5):1570–1592 (electronic), 2000.

[9] Andrew J. Cleary, Robert D. Falgout, Van Emden Henson, and Jim E. Jones. Coarse-grid selection for parallel algebraic multigrid. In *Solving irregularly structured problems in parallel (Berkeley, CA, 1998)*, volume 1457 of *Lecture Notes in Comput. Sci.*, pages 104–115. Springer, Berlin, 1998.

[10] Andrew J. Cleary, Robert D. Falgout, Van Emden Henson, Jim E. Jones, Thomas A. Manteuffel, Stephen F. McCormick, Gerald N. Miranda, and John W. Ruge. Robustness and scalability of algebraic multigrid. *SIAM J. Sci. Comput.*, 21(5):1886–1908 (electronic), 2000.

[11] Robert D. Falgout and Panayot S. Vassilevski. On generalizing the algebraic multigrid framework. *SIAM J. Numer. Anal.*, 42(4):1669–1693 (electronic), 2004.

[12] Van Emden Henson and Ulrike Meier Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.*, 41(1):155–177, 2002.

[13] W. L. Wan, Tony F. Chan, and Barry Smith. An energy-minimizing interpolation for robust multigrid methods. *SIAM J. Sci Comput.*, 21:1632–1649, 1999.

[14] Tony F. Chan and Petr Vanek. Detection of strong coupling in algebraic multigrid solvers. In *Multigrid methods, VI (Gent, 1999)*, volume 14 of *Lect. Notes Comput. Sci. Eng.*, pages 11–23. Springer, Berlin, 2000.

[15] J. Mandel, M. Brezina, and P. Vaněk. Energy optimization of algebraic multigrid bases. *Computing*, 62(3):205–228, 1999.

[16] Petr Vaněk, Marian Brezina, and Jan Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numer. Math.*, 88(3):559–579, 2001.

[17] A. Krechel and K. Stüben. Operator dependent interpolation in algebraic multigrid. In *Multigrid methods V (Stuttgart, 1996)*, volume 3 of *Lect. Notes Comput. Sci. Eng.*, pages 189–211. Springer, Berlin, 1998.

[18] Jan Mandel. Local approximation estimators for algebraic multigrid. *Electron. Trans. Numer. Anal.*, 15:56–65 (electronic), 2003.

[19] Jinchao Xu and Ludmil Zikatanov. On an energy minimizing basis for algebraic multigrid methods. *Comput. Vis. Sci.*, 7(3-4):121–127, 2004.

[20] G. Haase, U. Langer, S. Reitzinger, and J. Schöberl. Algebraic multigrid methods based on element preconditioning. *Int. J. Comput. Math.*, 78(4):575–598, 2001.

[21] Christian Wagner and Gabriel Wittum. Adaptive filtering. *Numer. Math.*, 78(2):305–328, 1997.

[22] Christian Wagner and Gabriel Wittum. Filtering decompositions with respect to adaptive test vectors. In *Multigrid methods V (Stuttgart, 1996)*, volume 3 of *Lect. Notes Comput. Sci. Eng.*, pages 320–334. Springer, Berlin, 1998.

[23] Gabriel Wittum. *Filternde Zerlegungen.* Teubner Skripten zur Numerik. [Teubner Scripts on Numerical Mathematics]. B. G. Teubner, Stuttgart, 1992.

[24] Randolph E. Bank. PLTMG: A software package for solving elliptic partial differential equations, users' guide 9.0. Technical report, Department of Mathematics, University of California at San Diego, 2004.

[25] Randolph E. Bank. Multigraph users' guide - version 1.0. Technical report, Department of Mathematics, University of California at San Diego, 2001.

[26] S. C. Eisenstat, M. C. Gursky, M.H. Schultz, and A.H. Sherman. Algorithms and data structures for sparse symmetric Gaussian elimination. *SIAM J. Sci. Statist. Comput.*, 2:225–237, 1982.

[27] Randolph E. Bank and Christian Wagner. Multilevel ILU decomposition. *Numerische Mathematik*, 82:543–576, 1999.

[28] Randolph E. Bank and R. Kent Smith. The incomplete factorization multigraph algorithm. *SIAM J. on Scientific Computing*, 20:1349–1364, 1999.

[29] Randolph E. Bank and Jinchao Xu. The hierarchical basis multigrid method and incomplete LU decomposition. In *Seventh International Symposium on Domain Decomposition Methods for Partial Differential Equations (D. Keyes and J. Xu, eds.)*, pages 163–173. AMS, Providence, RI, 1994.

[30] A. Reusken. A multigrid method based on incomplete Gaussian elimination. *J. Numer. Linear Algebra Appl.*, 3:369–390, 1996.

[31] G. Wittum. On the robustness of ILU smoothing. *SIAM J. Sci. Comput.*, 10:699–717, 1989.

[32] Randolph E. Bank and Tony F. Chan. An analysis of the composite step biconjugate gradient method. *Numerische Mathematik*, 66:295–319, 1993.

[33] Chris C. Paige and Michael A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[34] Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20:345–357, 1983.

[35] Randolph E. Bank and Jinchao Xu. An algorithm for coarsening unstructured meshes. *Numer. Math.*, 73:1–36, 1996.

[36] A. George and J.W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.