Electronic Transactions on Numerical Analysis.
Volume 12, pp. 1-65, 2001.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

NUMERICAL EXPERIMENTS WITH ALGEBRAIC MULTILEVEL PRECONDITIONERS*

GÉRARD MEURANT†

Abstract. This paper numerically compares different algebraic multilevel preconditioners to solve symmetric positive definite linear systems with the preconditioned conjugate gradient algorithm on a set of examples arising mainly from discretization of second order partial differential equations. We compare several different smoothers, influence matrices and interpolation schemes.

Key words. multilevel preconditioner, conjugate gradient.

AMS subject classifications. 65F10, 65F50.

1. Introduction. In this paper we report on numerical experiments using some multilevel preconditioners for solving symmetric positive definite linear systems $Ax = b$ with the Preconditioned Conjugate Gradient (PCG) method.

The methods we would like to compare are purely algebraic algorithms that only use the matrix and the right hand side as inputs.

An important issue for solving very large problems on Tflops scale parallel computers is scalability. One would like to have the computer time constant when the problem size per processor is fixed and the number of processors increases, which means that the dimension of the problem is increasing. When using an iterative method like PCG, this implies that the number of iterations must be constant when the problem size is increased. But this is not enough since we also need to have a number of operations per iteration proportional to the problem size. We will see that most multilevel preconditioners considered in this paper lead to algorithms which are almost scalable for some problems arising from discretization of second order partial differential equations.

All these methods use the same design principles as the Algebraic Multigrid algorithm (AMG). The standard AMG is a multigrid–like method that has been firstly defined for M–matrices; see [8], [2]. After some smoothing steps, the equation for the error with the residual as the right hand side is solved recursively on a coarser grid, corresponding to a subset of the unknowns. In AMG the coarse meshes are defined by looking at the entries of the matrix A. This is based on the fact that for M–matrices the largest entries in the inverse (which are positive) are given by the structure of A. Moreover, there is a decrease of the entries of the inverse away from the structure of A; see [5]. The set of dependencies of an unknown (a node) is defined by (a part of) the neighbours of the given node. An influence set is defined for each unknown as the “transpose” of the set of dependencies. The fine and coarse nodes for each level are found on this basis. Then, knowing the fine and coarse nodes, interpolation weights are computed using the entries of A and the equations of the linear system. The restriction R is the transpose of the interpolation (prolongation) matrix P and the coarse matrix is generally defined as $A_c = RAP$. As we said before, the method also uses a smoothing operator. An iteration (V–cycle) of the recursive algorithm is the following:

1. Do ν iterations of smoothing.
2. Restrict the residual r to $r_c = Rr$.
3. Recursively solve $A_c e_c = r_c$.
4. Interpolate e_c to $e = Pe_c$.

*Received November 27, 2000. Accepted for publication January 16, 2001. Recommended by R. S. Varga.
†CEA/DIF, DCSA/ED, BP 12, 91680 Bruyères le Chatel, France. E-mail: gerard.meurant@cea.fr

ETNA
Kent State University
etna@mcs.kent.edu

5. Add the correction e to the current iterate.
6. Do ν iterations of smoothing.

More generally we can introduce a parameter γ and replace step 3 by doing γ iterations of the same algorithm with one level less. Choosing $\gamma = 1$ is the V–cycle just described and having $\gamma = 2$ is denoted as a W–cycle.

For the classical (geometric) multigrid method, typically for the Poisson equation in the unit square with finite differences, one uses Gauss–Seidel (or relaxed Jacobi) as a smoother, bilinear interpolation and a coarse mesh defined by taking every other node in each direction in a red–black fashion; see [2]. For discontinuous or anisotropic coefficients problems more sophisticated smoothers and/or interpolations (using the matrix entries) have to be used. This is what we are supposed to get automatically with AMG.

If everything is symmetric (which can be obtained by using symmetric Gauss–Seidel as a smoother) a preconditioner for PCG is given by running one iteration of the previous algorithm starting from $x^0 = 0$.

The multilevel preconditioners we are going to compare proceed in the same way as AMG using different definitions of the smoother, the coarsening algorithm and the interpolation.

In the following sections we describe the algorithms in more detail and some numerical results which have been obtained on several elliptic problems as well as more general linear systems.

2. The multilevel preconditioners.

We are going to look at the different components of the multilevel algorithm: the smoother, the influence matrix, the coarsening and interpolation algorithms.

2.1. The smoother.

2.1.1. Symmetric Gauss–Seidel.

When solving symmetric linear systems with PCG we need a symmetric positive definite preconditioner. One way to extend what is done in classical multigrid is to use a symmetric Gauss–Seidel iteration. The Gauss–Seidel algorithm is done with the given ordering of the unknowns and then another step is done using the reverse ordering. We will denote this smoother by 'gs' in the numerical experiments.

2.1.2. Incomplete Cholesky.

Another smoother which has been proposed is the Incomplete Cholesky (IC) decomposition $LD^{-1}L^T$ (where L is lower triangular and D diagonal) of the matrix. There are many different variants of this algorithm. The most popular one is to use a decomposition whose non zero structure of L is the same as the structure of the lower triangular part of A. However, one can also keep a part of the fill–in either by looking at the size of the entries or by using the levels of fill–in, see [6] for a review and the references therein. In the numerical experiments reported here we will only consider the variant with no fill although to be fair in comparison with the other approximate inverse smoothers we should have retained some fill in for the most difficult problems. This incomplete decomposition is used in a Richardson iteration

$$LD^{-1}L^T(x^{k+1} - x^k) = b - Ax^k,$$

when solving $Ax = b$. We will denote this smoother by 'ic'.

2.1.3. AINV approximate inverse.

Here the idea is to use an approximate inverse M from AINV as a smoother in a Richardson iteration defined as

$$x^{k+1} = x^k + M(b - Ax^k),$$

when solving $Ax = b$; see [1], [7]. This preconditioner computes an approximate factorization $M = ZD^{-1}Z^T$ of the inverse of A and involves a parameter τ used to define which elements are dropped during the factorization.

If we first consider a two–grid algorithm using one step of pre–smoothing and one step of post–smoothing starting from $x^0 = 0$, we can easily see from [7], that the preconditioner which we denote by $\tilde{M}_1$ is defined as

$$\tilde{M}_1 = M + M(I - AM) + (I - MA)(P(RAP)^{-1}R)(I - AM).$$

The matrix M from AINV is symmetric positive definite. Obviously $\tilde{M}_1$ is symmetric. It has been shown in [7] that $\tilde{M}_1$ is positive definite if we suppose that M is such that $\rho(I - AM) < 1$. In [7] it was also proven that under the same hypothesis 1 is a multiple eigenvalue of $\tilde{M}_1 A$. Moreover, all the eigenvalues of $\tilde{M}_1 A$ are smaller than or equal to 1.

This occurs whatever the choice of M, R and P as long as M and $\tilde{M}_1$ are symmetric and positive definite. Therefore the convergence rate of PCG using the two–grid preconditioner depends only on the smallest eigenvalue. Moreover, the same results apply if the coarse matrix is obtained by using the same algorithm recursively, that is in the multilevel case.

2.1.4. The approximate inverse of Tang and Wan. This smoother is an approximate inverse suggested by Tang and Wan in [9]. The approximate inverse M for a general matrix A is computed to minimize

$$\|I - MA\|_F,$$

the F norm being the Frobenius norm. This problem is equivalent to solving n l_2 minimization problems, n being the order of A,

$$\|A^T m_i - e_i\|, \tag{2.1}$$

where m_i^T is the ith row of M and e_i is the ith column of the identity matrix. Generally the difficult point in deriving approximate inverses of this type is to select the sparsity pattern to be imposed on M. There are sophisticated algorithms to do this adaptively. However, here we are only looking for a smoother. The proposal of Tang and Wan is to use a sparsity pattern corresponding to the neighbours of node i in the graph of A. More specifically, if we define the neighbours of i to be at level 0 and the same plus the neighbours of the neighbours to be at level 1, in the same way we can define the k–level neighbour set for any $k > 0$.

The smoother is defined by extracting from A^T the (k, l) submatrix $A^T_{k,l}$ corresponding to the rows in level k and the columns in level l and then solving the least squares problem (2.1) with the normal equations

$$A_{k,l} A^T_{k,l} m_i = A_{k,l} e_i.$$

This gives the ith row of M. Here we shall use $l = 1, k = 0$ as proposed by Tang and Wan, although this will not be enough for the most difficult problems. Usually this gives small linear systems to solve when the matrix A is sparse. However, the matrix M might not be symmetric, therefore we use $\frac{1}{2}(M + M^T)$ as the smoother. As with AINV, this is used in a Richardson iteration. We denote this smoother by 'tw'.

2.1.5. Conjugate gradient. A few iterations of the conjugate gradient algorithm have been suggested as a smoother. We will use $\nu = 3$ iterations of CG with a diagonal preconditioner and denote this smoother as 'gc'. We use 3 iterations because for most problems this gives the minimum number of operations.

2.1.6. Conjugate gradient with AINV preconditioner. We use 3 iterations of CG preconditioned with AINV with a parameter τ. This will be denoted as 'cg'.

2.1.7. A least squares polynomial. This smoother is simply to define M as a least squares polynomial preconditioner p_k. We wish to have the polynomial $\lambda p_k(\lambda)$ as close as possible to 1 in some sense on $[a, b]$, an interval enclosing the eigenvalues of A. One way of achieving this is to look for the polynomial p_k of degree $k \in \mathcal{Q}_k$, the set of polynomials of degree less than or equal to k that minimizes

$$\int_a^b (1 - \lambda q(\lambda))^2 w(\lambda)\, d\lambda, \quad q \in \mathcal{Q}_k,$$

where $w(\lambda)$ is a positive weight. Usually, one chooses the Jacobi weights,

$$w(\lambda) = (b - \lambda)^\alpha (\lambda - a)^\beta, \quad \alpha \geq \beta \geq -\frac{1}{2},$$

because we know the orthogonal polynomials associated with these weights. The solution of the minimization problem is explicitly known; see [6]. We shall use the Chebyshev weights $\alpha = \beta = -1/2$, and $a \geq 0$ and b are given by the Gerschgorin bounds for the eigenvalues of A. A stable algorithm for computing $z = P_k(A)r$ is the following (see [6] for details of the derivation):

$$s_0(0) = \frac{1}{\sqrt{\pi}}, \quad s_1(0) = \sqrt{\frac{2}{\pi}}\frac{a+b}{a-b}, \quad s_2(0) = \sqrt{\frac{2}{\pi}}\left[2\left(\frac{a+b}{a-b}\right)^2 - 1\right],$$

and

$$s_j(0) = 2\mu(0)s_{j-1}(0) - s_{j-2}(0), \quad j = 3, \ldots, k+1,$$

$$b_j = \frac{s_j(0)}{\sum_{i=0}^{k+1} s_i^2(0)}, \quad j = 1, \ldots, k+1.$$

Then,

$$z_{k+1} = b_{k+1} r, \quad z_k = b_k r + \frac{2}{b-a}(2A - (a+b)I)z_{k+1},$$

$$z_j = b_j r + \frac{2}{b-a}(2A - (a+b)I)z_{j+1} - z_{j+2}, \quad j = k-1, \ldots, 1,$$

and

$$u_{k+1} = \frac{4}{a-b} s_k(0) z_{k+1},$$

$$u_{j+1} = \frac{4}{a-b} s_j(0) z_{j+1} + u_{j+2}, \quad j = k-1, \ldots, 1.$$

Finally,

$$z = \sqrt{\frac{2}{\pi}}\frac{2}{a-b} z_1 + u_2.$$

This preconditioner is also used as a smoother in a Richardson iteration. We denote this smoother by 'po' and k will be the degree of the polynomial.

ETNA
Kent State University
etna@mcs.kent.edu

2.2. The influence matrix. An important part of the algorithm is to decide which unknowns correspond to the fine "nodes" or points and which to the coarse points. Hence, the set $\mathcal{N} = \{1, \dots, n\}$ of the unknown indices is split into two sets $\mathcal{N} = F \cup C$.

First of all, for each unknown (point or node) i we define the set of dependencies S_i and an influence matrix S whose rows are the S_i's padded with zeros. This can be done in many ways. The standard AMG algorithm (see [4], [8], [10], [2]) for an M–matrix defines

$$S_i = \{j | -a_{i,j} > \tau \max_{k \neq i}(-a_{i,k}), \quad \tau < 1\},$$

where τ is a parameter that defines which elements are strongly connected to i. The set of points that i influences is $S_i^T = \{j | \; i \in S_j\}$. This definition can be generalized to any matrix by

$$S_i^A = \{j \mid |a_{i,j}| > \tau \max_{k \neq i} |a_{i,k}|, \quad \tau < 1\}.$$

Since the previous definition uses the matrix A itself we will denote it by 'a' in the numerical experiments. We remark that this influence matrix is local as it is looking only at the neighbours of i in the graph of A.

Rather than using an influence matrix given by the entries of A, it seems natural to measure the influences of the points by the inverse of A since this describes how the unknowns are linked together. However, since we only have (eventually) at our disposal the approximate inverse M from AINV or the Tang and Wan approximate inverse, we can define (see [7])

$$S_i^M = \{j \in \mathcal{N}, j \neq i | \; m_{i,j} \neq 0\}.$$

This choice will be denoted as 'm'. Generally, we do not want to compute M when using AINV since it is only given in factored form and the solve at every iteration can be done with multiplications with Z and Z^T. Thus we will also define the influence matrix as

$$S_i^Z = \{j \in \mathcal{N}, j \neq i | \; n_{i,j} \neq 0\}.$$

Let Q be a diagonal matrix whose diagonal elements are the square roots of those of D^{-1} and $\tilde{Z} = ZQ$. Then, the matrix N is defined as $N = \tilde{Z} + \tilde{Z}^T - Q$. This choice will be denoted as 'z'.

We will see in the numerical experiments that when solving anisotropic problems with AINV the choices 'm' and 'z' can lead us to obtain too many couplings and coarse grids with very few points. Although this is right from the physics of the problem, it does not always allow us to compute the solution as fast as we would like and moreover we will not be able to use every interpolation scheme with these grids. A way to avoid this is to compute the coarse grid and then to check if every F node has at least one C node in its neighbours in the graph of A. If this is not the case, we can choose one of the neighbours and change its status to a C point. One can also filter the matrix S^Z by only keeping the largest elements in N.

Another way to obtain coarse grids with more nodes using AINV is to use two approximate inverses, one with a threshold τ_1 to compute the coarse grid and another one with a threshold $\tau_2 \leq \tau_1$ as a smoother. The first decomposition can be easily obtained from the second one. This would allow us to be able to smooth more on difficult problems without leading to grids which are too coarse.

2.3. The coarsening algorithm. Once S_i is fixed by any of the previous methods, there are different ways we can follow to decide which are the F and C points.

2.3.1. Algorithm C1. What we are going to denote as the "standard" ('st') coarsening algorithm is mainly based on two principles:

1. For each $i \in F$, each node $j \in S_i$ should either be in C or should depend on at least one point in C_i, which is the set of coarse points which are going to be used for the interpolation of i.
2. C should be (as much as possible) a maximal subset with the property that no C point depends on another C point.

The first criterion tends to increase the number of C points. The second one is used to limit the number of points in the coarse grid. The standard coarsening algorithm is defined by two passes. The first one uses weights w_i, which are the number of points that depend on i. One step of the algorithm is the following:

1. Choose the first point i with maximal weight as a C point.
2. Assign the points that i influences as F points.
3. Increment by 1 the weights of the points influencing these new F points.
4. Decrease by 1 the weights of the points that depend on i.

This first pass guarantees that each F point has at least one connection to a C point. This is needed for the standard interpolation. It tends sometimes to produce too many F points. A second pass (see [10]) could be added in which some F points are made into C points to enforce the first criterion and to minimize C–C connections. The idea is to test each F point to see if the first criterion is satisfied. The neighbours of i are split into the coarse (interpolatory) points C_i, the strongly connected non interpolatory points D_i^S (those which belong to S_i) and the weakly connected non interpolatory points D_i^W. If there is a point in D_i^S which is not connected to C_i, it is tentatively flagged as a C point. If the first criterion is verified with this new C point, it is definitely considered as a C point and testing on other F points continues.

For the problems we are going to consider this second pass has not much effect and since it is costly we are going to skip it.

2.3.2. Algorithm C2. There are many other ways to generate the F and C points. Let us look at an algorithm proposed by Cleary, Falgout, V.A. Henson, and Jones [3]. This algorithm was devised to be used on parallel computers. When used in parallel the algorithm first selects a set of independent points and then operates independently on the points of this set, which are the starting points. The weights to be used are the same as in 'st' (although [3] added random numbers to break ties). For our purpose, since we are not looking at the parallelism issue, we just select one point. One step of the algorithm is:

1. Choose a point i of maximal weight as a C point (but check if this does not introduce a C–C connection).
2. For all points j that influence i decrease the weight of j by 1 and remove edge i, j from the graph of S.
3. For all points j that depend on i, remove edge j, i from the graph and for each k that j influences if k depends on i decrease the weight of j by 1, remove the edge k, j from the graph.
4. When the weight of a point is 0 (or 1) flag it as an F point.

The original proposal [3] does not refuse C–C connections and flags nodes when their weight is 0. We will denote this algorithm by 'p' and the one with a check and a flag when the weight is 1 by 'p1'.

2.3.3. Algorithm C3. Another possibility is to use the following algorithm. As weights we use the max norms of the columns of M or N scaled to have a unit diagonal. We choose the point with maximal weight as a C point and flag the influences as F points. We find the points that influence the new F points and raise their weights by a percentage of the initial maximum weight. This is followed by the second pass of the standard algorithm. Finally, for interpolation purposes, we check that every F point has at least one C neighbour. This will be denoted as 'm2'.

Finally, we remark that the selection of the coarse grids depends on the matrices on the coarse levels and therefore also on the interpolation scheme which gives P and R and consequently the next coarse matrix. This implies that if we change the interpolation scheme, the number and location of coarse nodes also change.

2.4. The interpolation algorithm. The classical multigrid algorithm uses bilinear interpolation. However, it is well known that this is not satisfactory for general problems.

2.4.1. Algorithm I1. The standard AMG algorithm uses instead an interpolation based on the equations in the linear system. However, some approximations have to be done. Finally, for a point i in F, the interpolation weight with a coarse point j is

$$\omega_{i,j} = -\frac{a_{i,j} + \sum_{k \in D_i^S} \frac{a_{i,k} a_{k,j}}{\sum_{m \in C_i} a_{k,m}}}{a_{i,i} + \sum_{k \in D_i^W} a_{i,k}}.$$

This will be denoted as 'st' (standard interpolation). It is obtained by writing the equation for $Ae = 0$ and by doing some approximations; namely writing that $e_j \approx e_i$ for weak connections and using a weighted average for F connections. Note that the given F point needs to have at least one coarse point in its neighbourhood in the graph of A in order to be able to apply this interpolation scheme.

2.4.2. Algorithm I2. We can also use the approximate inverse to generate the interpolation weights; see [7]. Let C_i be the set of coarse nodes in S_i. For an F point i, the interpolation weights $w_{i,j}$ are defined as

$$w_{i,j} = \frac{n_{i,j}}{\sum_{l \in C_i} n_{i,l}}, \quad j \in C_i,$$

where $N = M$ or $\tilde{Z} + \tilde{Z}^T - Q$. In fact, in the most general case we use the absolute values of the coefficients. The rationale behind this choice being that the points which are more important for interpolation are the ones with the strongest connections. These choices will be denoted respectively as 'im' and 'iz'.

2.4.3. Algorithm I3. Another possibility is to use the approximate inverse in a different way. Suppose the matrix is permuted to

$$\begin{pmatrix} A_{f,f} & A_{f,c} \\ A_{f,c}^T & A_{c,c} \end{pmatrix},$$

and we have an approximate inverse M partitioned in the same way. This can be an approximate inverse of the permuted A or the permutation of an approximate inverse (note that this is not the same). Then, we can set the interpolation matrix as

$$P = \begin{pmatrix} -M_{f,f} A_{f,c} \\ I \end{pmatrix},$$

and the restriction is as usual $R = P^T$. The RAP matrix is

$$RAP = A_{c,c} - A_{f,c}^T M_{f,f} A_{f,c} - A_{f,c}^T M_{f,f}(I - A_{f,f} M_{f,f}) A_{f,c},$$

ETNA
Kent State University
etna@mcs.kent.edu

that is an approximate Schur complement plus a correction term which must be small if $M_{f,f}$ is a good approximation of the inverse of $A_{f,f}$. Strictly speaking, using the equations, we would have to apply a correction on the F unknowns after interpolation, but we neglect this as there was not a clear gain in the numerical experiments for large problems. We will refer to this as 'sc' interpolation.

2.4.4. Algorithm I4. We will also consider briefly other interpolation schemes. One is the energy minimization interpolation described by Wan, Chan, and Smith in the finite element framework; see [12]. Although this is formulated for finite elements, this interpolation can be used in a more general setting. The prolongation operator relates the coarse grid basis functions ϕ_i^H to the fine grid basis functions ϕ_i^h by

$$[\phi_1^H \cdots \phi_m^H] = [\phi_1^h \cdots \phi_n^h]P.$$

The coarse grid functions can be expressed in the fine grid basis

$$\phi_i^H = \sum_{j=1}^{n} \varphi_j^i \phi_j^h,$$

and we are looking for coefficients φ_j^i that minimize the A–norm of the coarse grid basis functions satisfying the fact that the interpolation of a constant value is exact. Let

$$\varphi^i = (\varphi_1^i \dots \varphi_n^i)^T, \quad \phi = (\varphi^1 \dots \varphi^m)^T.$$

Then, the minimization problem is $\min \frac{1}{2}\phi^T Q\phi$, $B^T\phi = 1$, where 1 is a vector of all ones, Q is a block diagonal matrix whose diagonal blocks Q_i are given by $(Q_i)_{k,l} = a_{k,l}$ if k and l are neighbours of i (in the graph of A) and $\delta_{k,l}$ otherwise. The constraints matrix B is given as $B^T = (I_1^T \cdots I_m^T)$, with $(I_i)_{k,l} = 1$ if $k = l$, i being a neighbour of k and 0 otherwise. The minimization problem is solved using a Lagrange multiplier Λ. This gives a linear system

$$\begin{pmatrix} Q & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \phi \\ \Lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

This is solved by eliminating ϕ to get $(B^T Q^{-1} B)\Lambda = -1$. Fortunately, Q^{-1} is not difficult to obtain, since Q is block diagonal with small blocks. Once we have ϕ and therefore the φ^js, we set $P = (\varphi^1 \cdots \varphi^m)$. This interpolation will be denoted as 'em'.

2.4.5. Algorithm I5. Following the ideas of Wagner [11] we can try to compute an interpolation that minimizes $\|(I - PR_{inj})S\|_F$, where R_{inj} is the injection and S is the smoothing operator. So, this interpolation does not only involve A but the smoothing operator. Wagner [11] also uses this to determine the fine and coarse points. Here, we assume they are given by the coarsening algorithm. Then, we just have to solve simple constrained quadratic minimization problems. Let q_i be vectors such that $(q_i)_k$ is 1 if $k = i$, $-p_{i,k}$ if $k \in \mathcal{P}_i$, the set of nodes used for the interpolation in $i \in F$, and 0 otherwise. The $p_{i,k}$s are the entries of P. Then, for all i we have to solve

$$\min \|S^T q_i\|_2, \quad q_i^T 1 = 0.$$

This interpolation is denoted by 'wi'.

ETNA
Kent State University
etna@mcs.kent.edu

2.4.6. Algorithm I6. A last possibility is to use a "local" solve to compute the interpolation weights. Suppose we are considering node $i \in F$. Then i has coarse and fine neighbours. We look only at the coarse neighbours of the fine neighbours (assuming that we will fix a value of zero for the fine neighbours of the fine neighbours). Then from A we extract the matrix corresponding to this set of nodes (i, the neighbours of i and the coarse neighbours of the fine neighbours of i) and we express the fine unknowns as a function of the coarse unknowns in this "small" system. The coefficients expressing i as a function of the coarse nodes give the interpolation weights when normalized for having a sum of 1. If the extracted matrix is

$$\begin{pmatrix} B_{f,f} & B_{f,c} \\ B_{f,c}^T & B_{c,c} \end{pmatrix},$$

then, we have $x_f = -B_{f,f}^{-1} B_{f,c} x_c$, and this gives the desired solution. This interpolation is denoted by 'wm'. It is a little bit similar to the Schur complement interpolation but operates only locally. Of course, it can be extended to a larger stencil by considering the neighbours of the neighbours of the neighbours and so on.

Numerically, we will only look at these different interpolations for one (discontinuous) problem we are going to describe, since it is almost impossible to test the possible combinations of influence, coarsening, interpolation, and smoother.

2.5. Other possibilities. Variations of the previous algorithms which use the approximate inverse AINV include using a parameter τ which varies with the level or using only the q largest elements on each column of Z. Another possibility is to use the regular AINV on the finest level and to truncate on the coarsest levels. This can sometimes save some floating point operations and still give the same number of iterations as the full algorithm. Many other smoothers have been suggested in the literature like block Jacobi or symmetric Gauss–Seidel with multicolor orderings.

3. Numerical experiments. Here we describe the test problems we use and we comment on the results which are given in tables in the next section. In all the experiments in this paper, we stop with a criterion

$$\|r^k\| \leq \varepsilon \|r^0\|,$$

where r^k is the residual at iteration k and $\varepsilon = 10^{-10}$. The right hand side b is the same random vector in all experiments of the same dimension and the initial vector is $x^0 = 0$.

3.1. The Poisson equation. We solve the Poisson equation with Dirichlet boundary conditions in the unit square with m discretization points in each direction (excluding boundaries) with a natural (left to right and bottom to top) ordering. This gives a matrix of order $n = m^2$. It is well known (see for instance [6]) that the condition number of the matrix is proportional to $1/h^2$ where $h = 1/(m+1)$. This can be computed analytically. Experimentally, we found that $\kappa(A) = 0.4\ h^2$. For the dimensions we used ($m = 10 : 10 : 60$) this gives $48.4, 178.1, 388.8, 680.6, 1053.4, 1506.2$; therefore this matrix can be considered as well conditioned.

Let us first look at the AMG algorithm (that is, as a stand alone algorithm not using PCG). We use a Gauss–Seidel smoother, the coarse grids are chosen using 'a', with $\tau = 0.06$ (but any value less than 0.25 will give the same results at least for the first coarse level), the standard interpolation 'st' and one step of pre and post smoothing ($\nu = 1$). We would like to note that these results are obtained with our own Matlab implementation of this algorithm and it might not be completely comparable to what was defined in [8]. So, we do not claim

ETNA
Kent State University
etna@mcs.kent.edu

that our results are representative of what could be obtained with the implementation of Ruge and Stuben. We note also that we could have used all the smoothers, coarsening algorithms and interpolation schemes we are considering in an AMG–like algorithm and not as a preconditioner for CG. AMG has the advantage over PCG of not requiring scalar products.

We give the number of iterations, the reduction factor ρ (which is computed on the last iterations), the number of floating point operations (excluding the initialization phase).

In Table 1 we look at the multigrid results for (our implementation of the) AMG with a maximum of 7 levels (which is enough for the problem sizes we are going to consider). We use a direct solver when the dimension of the matrix is smaller than 10. The coarsening algorithm is used without a second pass. We give the number of operations divided by n and on the last line the number of points in the different grids.

We can see from Table 1 that the number of iterations is constant (at least for these problem sizes) as well as the asymptotic reduction factor. Moreover, the number of operations is almost $1470\,n$. This means that this method is scalable for this particular problem. However, an iteration of this method cannot be used as a preconditioner for PCG since the corresponding matrix $\tilde{M}_1$ is not symmetric positive definite. As we said before we will have to use a symmetric Gauss–Seidel smoother.

We now start looking at the results for the multilevel preconditioners using PCG. The algorithms are denoted by a tuple: (smoother, influence, coarsening, interpolation). Moreover we can also use different values of ν the number of smoothing steps and/or of the cycle parameter γ. In addition to the number of iterations, we give the number of nodes on each grid as well as the total storage for the preconditioner under 'str'. The number of operations is also given (under 'op') as well as the number of operations divided by the problem size. This is used to assess the real (sequential) scalability of the algorithm. Finally, we give the condition number as given by the eigenvalues that can be computed from the PCG coefficients. This might not be very accurate when the number of iterations is too small.

We remark that we cannot test all the combinations of the smoother, the influence matrix, the coarsening and the interpolation. For instance, we cannot use the standard 'st' interpolation if we do not have at least one coarse point in the neighbours of a fine node in the graph of A.

Table 2 gives the results of an algorithm using ('gs', 'a', 'st', 'st'). This algorithm is scalable with an operation count about the same as AMG (with $\nu = 1$). The cost of the smoother is larger (since we use symmetric Gauss–Seidel) but the number of iterations is approximately half those of AMG. Of course, the storage is rather low as we do not have to store the smoother for the symmetric Gauss–Seidel method. However, we note that this algorithm is not parallel. To obtain a parallel algorithm the symmetric Gauss–Seidel smoother would have to be used with a multicolor ordering of the unknowns. Finally, we can see that it does not pay to smooth more by using $\nu > 1$ since the decrease in the number of iterations is not large enough to compensate the increase for the cost of one iteration. It is interesting to note that the coarse matrices are better and better conditioned when their dimension decreases. For instance, for $m = 40$ we have $\kappa(A_C) = 680.6, 170.6, 42.5, 10.6, 2.8$. The ratio between successive condition numbers is almost 4. This is also true for other problem sizes.

Table 3 shows the results with an Incomplete Cholesky smoother everything else being the same as in Table 2. The results of Table 3 are a little better than those in Table 2 using a symmetric Gauss–Seidel smoother regarding the number of operations. The storage is larger with the IC smoother since we have to store the preconditioner. The method is scalable but also not parallel although there are some ways to partially parallelize the Incomplete Cholesky factorization.

Table 4 gives results for the AINV smoother everything else being the same as in Table 2.

The results in Table 4 show that AINV is a smoother which is not as good (with this value of τ) as the symmetric Gauss–Seidel or the Incomplete Cholesky decomposition. The number of iterations is almost twice what it is for IC. However, it must be noted that we do not keep too much fill–in in AINV (using a smaller value of the parameter τ will decrease the number of iterations although at the expense of a larger storage). Moreover, AINV is a fully parallel smoother since it only involves matrix–vector multiplies. The number of iterations can be lowered by using a W–cycle ($\gamma = 2$). Then for $\nu = 1, 2, 5$ we obtain $9, 6, 4$ iterations but the number of operations is larger than with $\gamma = 1$.

The results for 'tw' are given in Table 5. From these results and comparing with those in Table 4, we see than 'tw' is a better smoother than 'ai'. The number of iterations and the number of operations are approximately the same as for the symmetric Gauss–Seidel smoother 'gs'. However, using 'tw' is fully parallel since the operation to apply the smoother is a matrix–vector product.

Table 6 gives the results for 3 iterations of the conjugate gradient with a diagonal preconditioner as a smoother. This is a good smoother but more expensive than IC. The number of iterations is smaller than for the AINV smoother but the cost is higher.

Table 7 shows that it is not interesting to use PCG with an AINV preconditioner as a smoother for this problem. The number of iterations is the same as with a diagonal preconditioner but, of course, the cost is larger. However, using this smoother is better than smoothing more when using the AINV smoother.

The results for the least squares polynomial are given in Table 8. We use a polynomial of order 1 since it is shown in Table 9 that, although increasing the degree of the polynomial decreases the number of iterations, there is no gain concerning the number of operations. This is because the decrease in the number of iterations is not fast enough to compensate for the larger number of matrix–vector products when we increase the degree.

The polynomial is computed using an interval $[a, b]$, which is usually taken as $0 \leq a \leq \lambda_{min}(A), b \geq \lambda_{max}(A)$. We have tried varying a since one can argue that we do not need to approximate all the eigenvalue spectrum for a smoother. For the Poisson problem, starting with $a = \lambda_{min}$ and increasing a towards b increases slightly the number of iterations by 1 or 2.

In Table 10 we look at the other coarsening algorithms when using 'a' to define the influence matrices and AINV as a smoother. The good news is that the number of iterations is almost insensitive to the coarsening algorithm. The bad news is that there can be very large differences in the number of operations depending on the choices of the coarse grids. For instance, 'p' (as used in this paper) tends to generate coarse grids with too many nodes and the cost is much larger. So, it is of interest to be able to generate coarse grids with as many few nodes as we could while preserving the number of iterations.

We now describe the results with the other interpolation schemes. Table 11 gives the results with the Schur interpolation 'sc'. The results using the Schur interpolation are clearly not scalable. There is a large increase in the number of iterations, the number of operations and the storage. Therefore, we won't use this algorithm anymore for the other problems. However, we note that this kind of interpolation has been used successfully in other papers but with a diagonal approximation to $A_{f,f}$.

Table 12 gives the results for the interpolation scheme using the entries of the factors of the approximate inverse. The results of Table 12 are (almost) scalable. Although the number of iterations is slightly larger than for other smoothers (which shows that AINV is not as good in this respect as IC) the number of operations is scalable. We note that there is a slight increase in the condition number which is about twice what it is for IC. The storage is comparable with IC. We remark that the cost is a little bit higher because, for this problem,

ETNA
Kent State University
etna@mcs.kent.edu

there are a few more nodes on the coarse grids than using 'a' but the differences are very small. This can possibly be corrected by using a smaller value of τ since we will have less coarse nodes. However, the main reason for the difference in the number of iterations is the smoother.

One way to lower the number of iterations is to keep more fill–in (by using a smaller value of τ) at the expense of more operations per iteration or by using a W–cycle ($\gamma = 2$) but this also gives a larger number of operations.

If we would have used the standard 'st' interpolation (which is feasible for this problem because every F node has a C node in its neighbours in the graph of A) we would have gotten the same results within one or two iterations. Therefore, our choice of algorithm could depend on our goals: a parallel algorithm, the smallest number of operations or the smallest storage. Unfortunately some of these goals are conflicting.

The conclusion for the Poisson equation is that all these preconditioners are given good results except the one using the Schur interpolation. They are all scalable and it appears that the most important component for this problem is the smoother. However, this equation has constant coefficients, so we must investigate more difficult problems before being able to make choices.

3.2. An anisotropic problem. We now would like to solve a diffusion problem with constant but anisotropic coefficients. The diffusion coefficient is 1 in the x–direction and 100 in the y–direction. This is a tough problem for approximate inverses since the decrease in the elements of the inverse of A is very slow in one direction and therefore approximations of the inverse with only a few non zero entries are not going to be accurate. In fact, the fill–in for the approximate factors in AINV is very sensitive to the threshold parameter.

On the contrary, this is a very nice problem for IC if the unknowns are properly numbered since the approximate factorization is close to the exact one since (with possibly some renumbering) the matrix is almost block diagonal with tridiagonal blocks. In fact a block diagonal smoother will probably do well on this problem.

We symmetrically scale the matrix A to have a unit diagonal (obtaining A_d). However, for this problem, the condition number of the scaled matrix is the same as for the original one. Moreover, the condition number of A is almost the same as for the Poisson problem: $0.4/h^2$. The scaling is more generally necessary to have a threshold parameter τ which is not varying too much with the problem. The condition numbers of A_C for $m = 40$ are $672.3, 171.6, 59.3, 27.3, 14.6, 8.6, 6.1$.

Table 13 gives the results of our AMG implementation. The number of iterations is constant and the number of operations is approximately $1670\ n$. This is almost the same as for the Poisson problem. Therefore we have a scalable algorithm.

Table 14 gives the results using the multilevel preconditioner with the symmetric Gauss–Seidel smoother. The coarse grids are generated by looking at the matrix entries. For this matrix, this automatically gives a semi coarsening with a very regular grid.

Table 15 gives the results for IC as a smoother. They are better than with symmetric Gauss–Seidel because IC is doing well for this problem. However, these two smoothers are not parallel.

Table 16 gives the results using AINV as a smoother. As we said before, this leads to more storage than, for instance using IC, because of the decay properties of the inverse. It does not exist a value of τ giving the same storage as for IC. The only way to get a smaller storage will be to truncate the approximate inverse. The number of iterations is constant but the number of operations is slightly increasing. The storage is much higher than with the two previous smoothers.

The results for ‘tw’ are given in Table 17. This shows again that this is a better smoother than ‘ai’.

Table 18 shows that the number of iterations using PCG with a diagonal preconditioner is almost the same as with the symmetric Gauss–Seidel or AINV. However, the cost is larger.

The number of iterations in Table 19 using PCG with AINV preconditioner is smaller but this smoother is more costly. Note that this differs from the Poisson equation.

Table 20 gives the results for the least squares polynomial smoother. Again, this is a good parallel smoother since the results are as good as for symmetric Gauss–Seidel and there is no storage for the smoother.

For the anisotropic problem the results in Table 21 using the ‘z’ scheme for the influence matrix are not fully scalable, even with the W–cycle although the increase in the number of iterations is quite small. However, we note that the cost is not higher than for the other methods and the storage is smaller (by a factor of 2) than what we get with ‘a’. We note that if we use $\nu = 2$ with the W–cycle the results are almost scalable even though it is more costly for small problems.

It is likely that these troubles arise because the problem is strongly coupled in one direction and consequently we find a very coarse grid with only one coarse point on every vertical line of the mesh. All the other points on the vertical line are strongly coupled to that point. We would like to generate more points in the coarse grids.

This can be done by using a couple of threshold parameters. Table 22 gives the results with AINV and $\tau = (1, 0.01)$. This leads to coarse grids with more nodes and approximate inverses with more fill–in since the second parameter is smaller. We see from the results that the number of iterations is almost constant. However, the storage and the number of operations grow with the problem dimension. Moreover, the number of operations is much larger than when using $\tau = 0.1$. We note that smoothing more is not enough to obtain a number of iterations independent of the problem dimension. Only putting more nodes (using $\tau = (1, 0.1)$) allows to obtain a smaller number of iterations.

There are other ways to obtain grids with more nodes. For instance, we can generate the grid using ‘z’ and then check if every fine node has a C neighbour in the graph of A. If not, we can add coarse nodes by choosing one of the neighbours. This allows also to have a constant number of iterations. But, as we have seen, it is usually not necessary to have so many coarse nodes.

3.3. A discontinuous problem. Here we are concerned with an isotropic diffusion problem with constant but discontinuous coefficients. The diffusion coefficient is 1 except in the strip $[0, 1] \times [1/4, 3/4]$ where its value is 100. In the discretization we were not really cautious about the average of the coefficients; we just took their pointwise values. Therefore, we will have sometimes to use problem dimensions different from those of the previous problems in order not to have to compute the coefficients on the discontinuities.

We symmetrically scale the matrix A to have a unit diagonal. The scaled matrix is an M–matrix but it is not diagonally dominant. The condition number of A_d is almost $0.8/h^2$.

Table 23 gives the results of AMG. We can see that the number of iterations is increasing although only slowly. It can be that other values of τ could give better results.

Table 24 gives the results using the multilevel preconditioner with the symmetric Gauss–Seidel smoother. The coarse grids are generated by looking at the matrix entries. The number of iterations is constant and the number of operations is proportional to n.

Table 25 gives the results for IC as a smoother. They are comparable to those of the Gauss–Seidel smoother.

Table 26 gives the results using AINV as a smoother. Contrary to the results for the anisotropic problem the number of iterations is about the same as for the other smoothers.

ETNA
Kent State University
etna@mcs.kent.edu

The storage and the number of operations are larger. This can be corrected by using other values of τ.

The results for 'tw' are given in Table 27. We remark that for this problem the number of iterations is larger than with AINV but this smoother is cheaper and the storage is much smaller (which probably explains the number of iterations).

Table 28 shows that PCG with the diagonal preconditioner gives also almost the same results but the number of operations is larger. This is the same with the AINV preconditioner in Table 29.

The results for the least squares polynomial are given in Table 30. The number of operations is a little larger than for 'gs' and 'ic' but the storage is quite low and this smoother is parallel. The number of operations is smaller with 'tw' but the storage is smaller with the polynomial.

For the discontinuous problem, the results in Table 31 using the approximate inverse for coarsening are not scalable for the V–cycle since the number of iterations is slightly increasing with the problem dimension. The increase for the number of iterations is really small for the W–cycle. We note that, nevertheless, for these small problem dimensions the numbers of operations are comparable to those in Table 26 and the storage is smaller by a factor of 2 because the grids are much coarser. The number of iterations can be reduced if we use coarse grids with more points with $\tau = (0.2, 0.06)$ like in Table 32. It can be further reduce if we smooth a little more using $\tau = (0.2, 0.01)$ at the expense of a larger storage.

We check that the results are independent from the jumps in the coefficients by doing some computations with larger jumps. Moreover, this is also true for other problems with discontinuous coefficients. For 'z', 'iz' the results are almost independent of the jumps.

Table 33 gives the results using ('ai', 'a','st',-) for different interpolation schemes. The results for the 'st' interpolation are found in Table 26. One can see that there are not that many differences in the numbers of iterations which are almost independent of the size of the problem. There are more differences on the numbers of operations as well as for the storage. We ran out of memory with 'wi' because of the way we coded the algorithm in Matlab. This is not intrinsic to the method. For this problem the best results are given by the standard interpolation 'st'. The other schemes are more costly.

3.4. A problem with rapidly varying coefficients. This problem is isotropic. The diffusion coefficient is $1 + 1000|x - y|$. It varies from 1000 on the boundary to 1 on the main diagonal of the mesh. We symmetrically scale the matrix A to have a unit diagonal (obtaining A_d). The condition number of A_d is $0.2/h^2$.

Table 34 gives the results for AMG. The number of iterations is slowly increasing and slightly larger than for the Poisson problem.

Tables 35 to 40 show that the results are always in the same range for all smoothers. For the 'z' influence matrix in Table 42 the number of iterations is slowly increasing for $\gamma = 1$ and almost constant for $\gamma = 2$. The number of iterations is better for $\tau = (1, 0.01)$ but the cost is much larger. Amongst the parallel methods 'tw' gives the best results. However, the polynomial smoother is not far behind and the storage is smaller.

3.5. A random Laplacian. To show that multilevel methods are not only working for M–matrices we consider matrices arising from the Poisson equation as in the first example but with the signs of the non zero non diagonal coefficients chosen at random. This implies that the matrix is an H–matrix.

Table 44 shows that the results of AMG for this problem are even a little better than for the Poisson equation.

This is true also in Tables 45 to 52 for the PCG preconditioners. This shows that all these methods are working nicely for H–matrices.

3.6. Other problems. All the preceding examples except the last one arise from two dimensional diffusion equations on a square domain discretized with a five point scheme. We would like to see how these methods behave on examples coming from other areas of scientific computing. We chose some symmetric matrices from the Harwell–Boeing collection or from the Boeing collection stored in the Tim Davis' collection (`http://www.cise.ufl.edu`). We had to normalize some of these matrices to be able to use the same values of τ as before. Of course, since the order of the matrices are given we cannot check if there is a dependence of the number of iterations on the size of the problem. Moreover, some of the problems are quite small and it can be that for instance AINV is faster than its multilevel counterparts. A solution with a direct method is also much faster. We use the following examples:

1. 1138-bus. An admittance matrix of order 1138 with 4054 non–zeros. This matrix has a small minimum eigenvalue and a condition number of about $8.5\ 10^6$. Our AMG implementation does not work on this matrix and the other methods do not give very good results although they are converging. To obtain a meaningful problem we add 0.01 to the diagonal elements of the normalized matrix. This gives a matrix with a condition number of 201.
2. bcsstk01. A stiffness matrix of order 48 with 400 non-zeros. It was normalized. This matrix is not diagonally dominant, nor an M–matrix, but nevertheless positive definite. The condition number of the normalized matrix is 1361.
3. gr3030. A matrix arising from a nine point approximation to the Laplacian on the unit square with a 30×30 mesh. It has order 900 and 7744 non–zeros. The condition number of the normalized matrix is 195.
4. bcsstk34. A stiffness matrix of order 588 with 21418 non-zeros. This matrix was normalized. Its condition number is 1.8.
5. bcsstk27. A matrix arising from the buckling analysis of an engine inlet of order 1224 with 56126 non-zeros. The normalized matrix has a condition number of 1024. This problem is quite difficult to solve.

3.6.1. 1138-bus modified. Table 53 gives the results for some of the methods we have studied so far. The smallest number of operations is given by the AINV smoother used with the 'z' influence matrix and 'iz' interpolation which gives a very low storage. The preconditioner 'tw' gives a large number of iterations since for this example the approximate inverse that is produced for the fine level is not positive definite. We note that for this problem the AINV smoother is working quite well since the number of iterations is smaller than for the other smoothers with a storage which is not much larger. The worst results are given by the PCG smoother with a diagonal preconditioner (which has no effect for this problem).

3.6.2. bcsstk01. Table 54 show that the AINV smoother is again working well although 'tw' is cheaper. We note that a degree 1 polynomial does not give good results. The best results are given using the 'a' influence matrix and the 'st' interpolation. However, the results using 'z' and 'iz' (using a different value of τ to be able to generate at least two grids) are quite close and the storage is smaller.

3.6.3. gr3030. Table 55 gives the results for the 9 point finite difference matrix. For this problem the best results are given by the IC smoother although the other methods are not too far away.

3.6.4. bcsstk34. Table 56 gives the results for all the methods. A value of $\tau = 0.01$ is used to be able to generate grids with enough nodes with 'a'. The best results are given by the IC smoother. Using the 'z' influence matrix gives coarse grids with more nodes and therefore a larger storage. It is likely that using another value of τ will be better for 'z'.

3.6.5. bcsstk27. We are not able to solve this problem using IC with no fill-in as well as with low order polynomials. It is likely that we should have to keep more fill-in for the Cholesky decomposition. We can solve the problem with polynomials of order larger than 20. However, the number of operations is much larger than for others methods. Similarly, we note that using ‘tw’ we got a very large number of iterations. This could have been fixed by extending the stencil. However, it is much easier just to change the value of τ in AINV. The problem is efficiently solved using the symmetric Gauss-Seidel smoother but this algorithm is not parallel. Using AINV requires a small value of τ to obtain a small number of iterations but then the storage is quite large.

4. Tables of results. In this section we group the results that were analyzed in the previous section. We show the results in this way since whatever we would have been doing using LaTeX, the comments would not have been on the same page as the corresponding results.

TABLE 1
AMG for Poisson equation, $\tau = 0.06$

$m = 10$	$m = 20$	$m = 30$
11	12	12
$\rho = 0.08$	$\rho = 0.12$	$\rho = 0.12$
op=122941, /n=1221	op=561953, /n=1405	op=1297936, /n=1442
100-50-14	400-200-51-14	900-450-119–32-13
$m = 40$	$m = 50$	$m = 60$
12	12	12
$\rho = 0.12$	$\rho = 0.12$	$\rho = 0.12$
op=2313029, /n=1446	op=3664272, /n=1466	op=5269951, /n=1464
1600-800-206- -53-15	2500-1250-324- -84-26-17-13	3600-1800-461- -119-34-13

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 2
PCG for Poisson equation, $\tau = 0.06$, multilevel, ('gs', 'a', 'st', 'st'), $\gamma = 1$

m	$\nu = 1$	$\nu = 2$	$\nu = 5$
10	6 op=107129, /n=1071 100-50-14 str=1297, /n=13 $\kappa = 1.05$	5 op=138717, /n=1387 $\kappa = 1.02$	4 op=233153, /n=2332 $\kappa = 1.005$
20	6 op=484856, /n=1212 400-200-51-14 str=5561, /n=13.9 $\kappa = 1.07$	5 op=636731, /n=1592 $\kappa = 1.03$	4 op=1084798, /n=2712 $\kappa = 1.01$
30	6 op=1138521, /n=1265 900-450-119-32-13 str=12995, /n=14.4 $\kappa = 1.06$	5 op=1499701, /n=1666 $\kappa = 1.03$	4 op=2562353, /n=2847 $\kappa = 1.01$
40	7 op=2330221, /n=1456 1600-800-206-53-15 str=23173, /n=14.5 $\kappa = 1.07$	5 op=2681901, /n=1676 $\kappa = 1.03$	5 op=5505957, /n=3441 $\kappa = 1.01$
50	6 op=3222342, /n=1289 2500-1250-324-84-26-17 str=36744, /n=14.7 $\kappa = 1.06$	5 op=4248607, /n=1699 $\kappa = 1.03$	5 op=8727417, /n=3491 $\kappa = 1.01$
60	7 op=5319529, /n=1477 3600-1800-461-119-34-18-13 str=52943, /n=14.7 $\kappa = 1.07$	5 op=6126171, /n=1702 $\kappa = 1.03$	5 op=12584007, /n=3496 $\kappa = 1.01$

TABLE 3
PCG for Poisson equation, $\tau = 0.06$ *, multilevel, ('ic', 'a', 'st', 'st'),* $\gamma = 1$

m	$\nu = 1$	$\nu = 2$	$\nu = 5$
10	5 op=85197, /n=852 100-50-14 str=2005, /n=20 $\kappa = 1.02$	4 op=114429, /n=1144 $\kappa = 1.007$	3 op=196072, /n=1961 $\kappa = 1.002$
20	5 op=381911, /n=954.8 400-200-51-14 str=8584, /n=21.5 $\kappa = 1.03$	4 op=521019, /n=1302 $\kappa = 1.01$	4 op=1132936, /n=2832 $\kappa = 1.003$
30	5 op=894202, /n=993.6 900-450-119-32-13 str=20009, /n=22.2 $\kappa = 1.02$	4 op=1223778, /n=1360 $\kappa = 1.01$	4 op=2667643, /n=2964 $\kappa = 1.003$
40	5 op=1598145, /n=998.8 1600-800-206-53-15 str=35667, /n=22.3 $\kappa = 1.03$	5 op=2631408, /n=1644 $\kappa = 1.01$	4 op=4771293, /n=2982 $\kappa = 1.004$
50	5 op=2528438, /n=1011 2500-1250-324-84-26-17 str=56497, /n=22.6 $\kappa = 1.02$	5 op=4165773, /n=1666 $\kappa = 1.01$	4 op=7557527, /n=3023 $\kappa = 1.004$
60	6 op=4265230, /n=1185 3600-1800-461-119-34-18-13 str=81388, /n=22.6 $\kappa = 1.03$	5 op=6008813, /n=1669 $\kappa = 1.01$	4 op=10901958, /n=3028 $\kappa = 1.005$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 4
PCG for Poisson equation, $\tau = 0.06$, multilevel, ('ai', 'a', 'st', 'st'), $\gamma = 1$

m	$\nu = 1$	$\nu = 2$	$\nu = 5$
10	11 op=167073, /n=1670 100-50-14 str=1892, /n=18.9 $\kappa = 1.74$	7 op=171069, /n=1711 $\kappa = 1.22$	4 op=219258, /n=2193 $\kappa = 1.02$
20	13 op=834163, /n=2085 400-200-51-14 str=8179, /n=20.4 $\kappa = 1.96$	9 op=946283, /n=2366 $\kappa = 1.32$	5 op=1199387, /n=2998 $\kappa = 1.03$
30	14 op=2077423, /n=2308 900-450-119-32-13 str=19044, /n=21.2 $\kappa = 2.01$	9 op=2214933, /n=2461 $\kappa = 1.34$	6 op=3299799, /n=3666 $\kappa = 1.04$
40	14 op=3723423, /n=2327 1600-800-206-53-15 str=34278, /n=21.4 $\kappa = 2.03$	10 op=4381433, /n=2738 $\kappa = 1.35$	6 op=5944087, /n=3715 $\kappa = 1.04$
50	14 op=5897948, /n=2359 2500-1250-324-84-26-17 str=54255, /n=21.7 $\kappa = 2.04$	10 op=6949048, /n=2780 $\kappa = 1.35$	6 op=9437844, /n=3775 $\kappa = 1.04$
60	14 op=8515738, /n=2365 3600-1800-461-119-34-18 str=78369, /n=21.8 $\kappa = 2.05$	10 op=10041954, /n=2789 $\kappa = 1.36$	6 op=13648658, /n=3791 $\kappa = 1.04$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 5
PCG for the Poisson equation, $\tau = 0.06$, multilevel, ('tw', 'a', 'st', 'st')

m	$\nu = 1$	$\nu = 2$	$\nu = 5$
10	7 op=102165, /n=1022 100-50-14 str=2221, /n=22.1 $\kappa = 1.13$	5 op=119733, /n=1197 $\kappa = 1.03$	4 op=208333, /n=2083 $\kappa = 1.01$
20	8 op=517790, /n=1294 400-200-51-14 str=9612, /n=24 $\kappa = 1.14$	6 op=641208, /n=1603 $\kappa = 1.05$	5 op=1165859, /n=2915 $\kappa = 1.02$
30	7 op=1077525, /n=1197 900-450-119-32-13 str=22481, /n=25 $\kappa = 1.13$	6 op=1509661, /n=1677 $\kappa = 1.04$	5 op=2754397, /n=3060 $\kappa = 1.02$
40	8 op=2172141, /n=1358 1600-800-206-53-15 str=40139, /n=25.1 $\kappa = 1.14$	6 op=2702569, /n=1689 $\kappa = 1.05$	5 op=4933860, /n=3084 $\kappa = 1.02$
50	8 op=3438494, /n=1375 2500-1250-324-84-26-17 str=63647, /n=25.5 $\kappa = 1.14$	6 op=4281980, /n=1713 $\kappa = 1.05$	5 op=7823195, /n=3129 $\kappa = 1.02$
60	8 op=4961032, /n=1378 3600-1800-461-119-34-18 str=91737, /n=25.5 $\kappa = 1.14$	6 op=6179210, /n=1716 $\kappa = 1.05$	5 op=11291359, /n=3136 $\kappa = 1.02$

TABLE 6
PCG for the Poisson equation, $\tau = 0.06$, $\nu = 3$, multilevel, ('gc', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=258997, /n=2590 100-50-14 str=1297, /n=13 $\kappa = 1.11$	3 op=346789, /n=3468 $\kappa = 1.0006$
20	7 op=1161445, /n=2904 400-200-51-14 str=5561, /n=13.9 $\kappa = 1.10$	3 op=1825181, /n=4563 $\kappa = 1.0007$
30	7 op=2713029, /n=3014 900-450-119-32-13 str=12995, /n=14.4 $\kappa = 1.09$	3 op=4684773, /n=5205 $\kappa = 1.0007$
40	7 op=4837789, /n=3024 1600-800-206-53-15 str=23173, /n=14.5 $\kappa = 1.09$	4 op=10394193, /n=6496 $\kappa = 1.0008$
50	7 op=7645725, /n=3058 2500-1250-324-84-26-17 str=36744, /n=14.7 $\kappa = 1.08$	3 op=13882757, /n=5553 $\kappa = 1.0005$
60	7 op=11017285, /n=3060 3600-1800-461-119-34-18 str=52943, /n=14.7 $\kappa = 1.07$	3 op=19903005, /n=5529 $\kappa = 1.0005$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 7
PCG for the Poisson equation, $\tau = 0.06$, $\nu = 3$, multilevel, ('cg', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	5 op=273117, /n=2731 100-50-14 str=1892, /n=18.9 $\kappa = 1.02$	3 op=486565, /n=4866 $\kappa = 1$
20	7 op=1657317, /n=4143 400-200-51-14 str=8179, /n=20.4 $\kappa = 1.12$	3 op=2578333, /n=6446 $\kappa = 1.0002$
30	7 op=3832301, /n=4258 900-450-119-32-13 str=19044, /n=21.2 $\kappa = 1.11$	3 op=6072229, /n=6747 $\kappa = 1.0003$
40	7 op=6992029, /n=4370 1600-800-206-53-15 str=34278, /n=21.4 $\kappa = 1.07$	3 op=11914653, /n=7447 $\kappa = 1.0003$
50	7 op=11029221, /n=4412 2500-1250-324-84-26-17 str=54255, /n=21.7 $\kappa = 1.08$	3 op=18004603, /n=7598 $\kappa = 1.0004$
60	7 op=15931997, /n=4426 3600-1800-461-119-34-18 str=78369, /n=21.8 $\kappa = 1.07$	3 op=27416989, /n=7616 $\kappa = 1.0003$

TABLE 8
PCG for the Poisson equation, $\tau = 0.06$, multilevel, ('po', 'a', 'st', 'st'), $k = 1$

m	$\nu = 1$	$\nu = 2$	$\nu = 5$
10	6 op=132273, /n=1323 100-50-14 str=1297, /n=13 $\kappa = 1.07$	5 op=193605, /n=1936 $\kappa = 1.03$	4 op=362233, /n=3622 $\kappa = 1.01$
20	7 op=681509, /n=1704 400-200-51-14 str=5561, /n=13.9 $\kappa = 1.09$	6 op=1029766, /n=2574 $\kappa = 1.05$	5 op=1998470, /n=4996 $\kappa = 1.02$
30	7 op=1596741, /n=1774 900-450-119-32-13 str=12995, /n=14.4 $\kappa = 1.09$	6 op=2418289, /n=2687 $\kappa = 1.04$	5 op=4701457, /n=5224 $\kappa = 1.02$
40	7 op=2852573, /n=1783 1600-800-206-53-15 str=23173, /n=14.5 $\kappa = 1.11$	6 op=4321221, /n=2701 $\kappa = 1.06$	5 op=8402409, /n=5251 $\kappa = 1.03$
50	7 op=4513885, /n=1806 2500-1250-324-84-26-17 str=36744, /n=14.7 $\kappa = 1.09$	6 op=6841180, /n=2737 $\kappa = 1.05$	5 op=13307185, /n=5323 $\kappa = 1.03$
60	7 op=6509253, /n=1808 3600-1800-461-119-34-18 str=52943, /n=14.7 $\kappa = 1.10$	6 op=9865746, /n=2740 $\kappa = 1.05$	5 op=19191079, /n=5331 $\kappa = 1.02$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 9
PCG for the Poisson problem, $m = 20$, $\tau = 0.06$, multilevel, ('po', 'a', 'st', 'st')

k	nb it	nb op
1	7	681509 /n=1704
2	6	890800 /n=2227
3	5	1015703 /n=2539
4	5	1268867 /n=3172
5	4	1267188 /n=3168
6	4	1478150 /n=3695
8	4	1900098 /n=4750
10	3	1856225 /n=4640
20	3	3543985 /n=8860
40	2	5187872 /n=12970

TABLE 10
PCG for Poisson equation, $\tau = 0.06$, multilevel, ('ai', 'a', '-', 'st'), $\gamma = 1$

m	'st'	'p'	'p1'	'm2'
10	11 op=167073 100-50-14	11 op=275085 100-50-38-26-16-10	11 op=167073 100-50-13	11 op=165441 100-50-12
20	13 op=834163 400-200-51-14	13 op=1558845 400-200-148-93-48-32-22	13 op=837691 400-200-54-13	13 op=845195 400-200-50-16
30	14 op=2077423 900-450-119-32-13	14 op=4853923 900-450-335-217-113-81-55	14 op=2071768 900-450-119-29-10	14 op=2066158 900-450-112-29-11
40	14 op=3723423 1600-800-206-53-15	14 op=10904208 1600-800-593-383-193-141-97	14 op=3731718 1600-800-209-53-20	14 op=3729153 1600-800-198-51-14
50	14 op=5897948 2500-1250-324-84-26-17	14 op=20402348 2500-1250-930-607-308-224-154	14 op=5892653 2500-1250-324-84-26-16	14 op=5884538 2500-1250-313-79-25-12

TABLE 11
PCG for Poisson equation, $\tau = 0.06$, multilevel, ('ai', 'z', 'st', 'sc'), $\gamma = 1$

m	$\nu = 1$	$\nu = 2$	$\nu = 5$
10	12 op=256824, /n=2568 100-50-25-12 str=3196, /n=31.9 $\kappa = 2.98$	8 op=276114, /n=2761 $\kappa = 1.87$	5 op=381183, /n=3812 $\kappa = 1.23$
20	20 op=2498084, /n=6245 400-200-100-51-23-12 str=19215, /n=48 $\kappa = 9.41$	14 op=2793278, /n=6983 $\kappa = 5.33$	9 op=3881723, /n=9704 $\kappa = 2.76$
30	28 op=9571569, /n=10635 900-450-225-110-60-30-15 str=54176, /n=60.2 $\kappa = 19.9$	21 op=11377105, /n=12641 $\kappa = 11.03$	13 op=15099181, /n=16777 $\kappa = 5.4$
40	38 op=26164947, /n=16353 1600-800-400-204-94-47-26 str=111615, /n=69.8 $\kappa = 34.5$	27 op=29422045, /n=18389 $\kappa = 18.9$	18 op=41625955, /n=26016 $\kappa = 9.04$
50	45 op=52278299, /n=20911 2500-1250-625-307-143-71-35 str=190351, /n=76.1 $\kappa = 53.1$	33 op=60515329, /n=24206 $\kappa = 29$	22 op=85339095, /n=34136 $\kappa = 13.7$
60	54 op=101004463, /n=28057 3600-1800-900-458-215-109-53 str=310963, /n=86.4 $\kappa = 75.7$	40 op=117903107, /n=32751 $\kappa = 41.3$	26 op=161832391, /n=44953 $\kappa = 19.4$

TABLE 12
PCG for Poisson equation, $\tau = 0.06$*, multilevel, ('ai', 'z', 'st', 'iz'),* $\gamma = 1$

m	$\nu = 1$	$\nu = 2$	$\nu = 5$
10	12 op=188340, /n=1883 100-50-19-10 str=2091, /n=20.9 $\kappa = 1.72$	8 op=207426, /n=2074 $\kappa = 1.22$	5 op=292389, /n=2928 $\kappa = 1.02$
20	14 op=987293, /n=2468 400-200-74-36-17 str=9433, /n=23.6 $\kappa = 1.96$	10 op=1161745, /n=2904 $\kappa = 1.35$	6 op=1575981, /n=3940 $\kappa = 1.11$
30	15 op=2448357, /n=2720 900-450-154-69-31-16 str=21701, /n=24.1 $\kappa = 2.02$	10 op=2709534, /n=3011 $\kappa = 1.48$	7 op=4216269, /n=4685 $\kappa = 1.19$
40	15 op=4513421, /n=2821 1600-800-267-130-61-29-13 str=39949, /n=24.9 $\kappa = 2.05$	11 op=5464389, /n=3415 $\kappa = 1.57$	8 op=8785665, /n=5491 $\kappa = 1.25$
50	16 op=7700375, /n=3080 2500-1250-431-208-97-50-21 str=64429, /n=25.7 $\kappa = 2.12$	11 op=8776877, /n=3518 $\kappa = 1.63$	8 op=14118983, /n=5647 $\kappa = 1.29$
60	16 op=11078581, /n=3078 3600-1800-607-283-136-65-27 str=92655, /n=25.7 $\kappa = 2.18$	12 op=13684058, /n=3801 $\kappa = 1.69$	8 op=20310541, /n=5642 $\kappa = 1.32$

TABLE 13
AMG for the anisotropic problem $\tau = 0.1$

$m = 10$	$m = 20$	$m = 30$
10	11	12
$\rho = 0.08$	$\rho = 0.11$	$\rho = 0.12$
op=106889, /n=1069	op=546900,/n=1367	op=1388527, /n=1543
100-50-20-10	400-200-100- -40-20-10	900-450-210- -90-45-15
$m = 40$	$m = 50$	$m = 60$
12	12	12
$\rho = 0.13$	$\rho = 0.14$	$\rho = 0.14$
op=2678135, /n=1674	op=4177997, /n=1671	op=6013795, /n=1670
1600-800-400- -200-100-40-30	2500-1250-600- -300-150-50-37	3600-1800-900- -420-210-60-20

TABLE 14
PCG for the anisotropic problem, $\tau = 0.1$, multilevel, ('gs', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=104749, /n=1047 100-50-20 str=1184, /n=11.8 $\kappa = 1.05$	4 op=197753, /n=1977 $\kappa = 1.002$
20	6 op=530923, /n=1327 400-200-100-40-20 str=5789, /n=14.5 $\kappa = 1.05$	4 op=1570273, /n=3926 $\kappa = 1.002$
30	6 op=1252516, /n=1392 900-450-210-90-45-15 str=13500, /n=15 $\kappa = 1.05$	4 op=4589393, /n=5099 $\kappa = 1.0002$
40	6 op=2537929, /n=1586 1600-800-400-200-120-80-60 str=28652, /n=17.9 $\kappa = 1.05$	4 op=17283553, /n=10802 $\kappa = 1.002$
50	6 op=3755860, /n=1502 2500-1250-600-300-150-75 str=41320, /n=16.5 $\kappa = 1.05$	4 op=16926673, /n=6771 $\kappa = 1.002$
60	6 op=5479546, /n=1522 3600-1800-900-420-210-90-75 str=60866, /n=16.9 $\kappa = 1.05$	4 op=25057153, /n=6960 $\kappa = 1.002$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 15
PCG for the anisotropic problem, $\tau = 0.1$, multilevel, ('ic', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	3 op=54933, /n=549 100-50-20 str=1907, /n=19.1 $\kappa = 1.0005$	2 op=108510, /n=1085 $\kappa = 1$
20	3 op=275557, /n=689 400-200-100-40-20 str=9258, /n=23.1 $\kappa = 1.001$	2 op=851096, /n=2128 $\kappa = 1$
30	3 op=647009, /n=719 900-450-210-90-45-15 str=21545, /n=23.9 $\kappa = 1.001$	2 op=2441542, /n=2713 $\kappa = 1$
40	3 op=1638579, /n=1024 1600-800-400-200-120-80-60 str=45037, /n=28.1 $\kappa = 1.001$	2 op=9142266, /n=5714 $\kappa = 1$
50	3 op=2432286, /n=973 2500-1250-600-300-150-75 str=65395, /n=26.2 $\kappa = 1.001$	2 op=9020462, /n=3608 $\kappa = 1$
60	3 op=3547688, /n=985 3600-1800-900-420-210-90-75 str=95952, /n=26.6 $\kappa = 1.001$	2 op=13364557, /n=3712 $\kappa = 1$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 16
PCG for the anisotropic problem, $\tau = 0.1$, multilevel, ('ai', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	5 op=88917, /n=889 100-50-20 str=2100, /n=21 $\kappa = 1.03$	3 op=147557, /n=1475 $\kappa = 1.0006$
20	6 op=634271, /n=1586 400-200-100-40-20 str=12232, /n=30.6 $\kappa = 1.04$	4 op=1722753, /n=4307 $\kappa = 1.002$
30	6 op=1673230, /n=1859 900-450-210-90-45-15 str=31839, /n=35.4 $\kappa = 1.05$	4 op=5304033, /n=5893 $\kappa = 1.002$
40	6 op=3564073, /n=2227 1600-800-400-200-120-80-60 str=68886, /n=43.05 $\kappa = 1.05$	4 op=18968193, /n=11855 $\kappa = 1.002$
50	6 op=5672348, /n=2268 2500-1250-600-300-150-75 str=107981, /n=43.2 $\kappa = 1.05$	4 op=22715713, /n=9086 $\kappa = 1.002$
60	6 op=8597374, /n=2388 3600-1800-900-420-210-90-75 str=163727, /n=45.5 $\kappa = 1.05$	4 op=35687666, /n=9913 $\kappa = 1.002$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 17
PCG for the anisotropic problem, $\tau = 0.1$, multilevel, ('tw', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=106173, /n=1062 100-50-20 str=2175, /n=21.7 $\kappa = 1.1$	4 op=206833, /n=2068 $\kappa = 1.005$
20	7 op=498941, /n=1247 400-200-100-40-20 str=10447, /n=26.1 $\kappa = 1.1$	4 op=1271233, /n=3178 $\kappa = 1.005$
30	7 op=1174365, /n=1305 900-450-210-90-45-15 str=24455, /n=27.2 $\kappa = 1.1$	4 op=3672033, /n=4080 $\kappa = 1.005$
40	7 op=2382829, /n=1489 1600-800-400-200-120-80-60 str=51642, /n=32.3 $\kappa = 1.1$	4 op=14170113, /n=8856 $\kappa = 1.005$
50	7 op=3534429, /n=1414 2500-1250-600-300-150-75 str=74845, /n=29.9 $\kappa = 1.1$	4 op=13844513, /n=5538 $\kappa = 1.005$
60	7 op=5160485, /n=1433 3600-1800-900-420-210-90-75 str=109978, /n=30.5 $\kappa = 1.1$	4 op=20565153, /n=5712 $\kappa = 1.005$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 18
PCG for the anisotropic problem, $\tau = 0.1$, $\nu = 3$, multilevel, ('gc,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=255893, /n=2559 100-50-20 str=1184, /n=11.8 $\kappa = 1.14$	3 op=343461, /n=3435 $\kappa = 1.0009$
20	7 op=1298749, /n=3247 400-200-100-40-20 str=5789, /n=14.5 $\kappa = 1.13$	3 op=2706781, /n=6767 $\kappa = 1.0007$
30	7 op=3038509, /n=3376 900-450-210-90-45-15 str=13500, /n=15 $\kappa = 1.12$	3 op=7705061, /n=8561 $\kappa = 1.0006$
40	7 op=6002285, /n=3751 1600-800-400-200-120-80-60 str=28652, /n=17.9 $\kappa = 1.11$	3 op=25570461, /n=15982 $\kappa = 1.0005$
50	7 op=89266621, /n=3571 2500-1250-600-300-150-75 str=41320, /n=16.5 $\kappa = 1.12$	3 op=26322181, /n=10529 $\kappa = 1.0004$
60	7 op=12989765, /n=3608 3600-1800-900-420-210-90-75 str=60866, /n=16.9 $\kappa = 1.12$	3 op=38772253, /n=10770 $\kappa = 1.0004$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 19
PCG for the anisotropic problem, $\tau = 0.1$, $\nu = 3$, multilevel, ('cg,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	2 op=160173, /n=1602 100-50-20 str=2100, /n=21 $\kappa = 1$	1 op=277117, /n=2771 $\kappa = 1$
20	3 op=1365989, /n=3415 400-200-100-40-20 str=12232, /n=30.6 $\kappa = 1.0001$	2 op=4002850, /n=10007 $\kappa = 1$
30	3 op=3634417, /n=4038 900-450-210-90-45-15 str=31839, /n=35.4 $\kappa = 1.0003$	2 op=12432212, /n=13814 $\kappa = 1$
40	3 op=7690141, /n=4806 1600-800-400-200-120-80-60 str=68886, /n=43 $\kappa = 1.0006$	2 op=40277673, /n=25174 $\kappa = 1$
50	4 op=15368018, /n=6147 2500-1250-600-300-150-75 str=107981, /n=43.2 $\kappa = 1.001$	2 op=50719673, /n=20288 $\kappa = 1$
60	4 op=23328865, /n=6480 3600-1800-900-420-210-90-75 str=163727, /n=45.5 $\kappa = 1.002$	2 op=79925475, /n=22202 $\kappa = 1$

TABLE 20
PCG for the anisotropic problem, $\tau = 0.1$, multilevel, ('po', 'a', 'st', 'st') $k = 1$

m	$\gamma = 1$	$\gamma = 2$
10	6 op=129781, /n=1298 100-50-20 str=1184, /n=11.8 $\kappa = 1.06$	4 op=247433, /n=2474 $\kappa = 1.002$
20	7 op=752509, /n=1881 400-200-100-40-20 str=5789, /n=14.5 $\kappa = 1.08$	4 op=1956033, /n=4890 $\kappa = 1.002$
30	7 op=1769517, /n=1966 900-450-210-90-45-15 str=13500, /n=15 $\kappa = 1.07$	4 op=5649793, /n=6277 $\kappa = 1.002$
40	8 op=4002993, /n=2502 1600-800-400-200-120-80-60 str=28652, /n=17.9 $\kappa = 1.15$	4 op=20382913, /n=12739 $\kappa = 1.002$
50	7 op=5272605, /n=2109 2500-1250-600-300-150-75 str=41320, /n=16.5 $\kappa = 1.08$	4 op=20345313, /n=8138 $\kappa = 1.002$
60	7 op=7691205, /n=2136 3600-1800-900-420-210-90-75 str=60866, /n=16.9 $\kappa = 1.08$	4 op=30159393, /n=8378 $\kappa = 1.002$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 21
PCG for the anisotropic problem, $\tau = 0.1$, multilevel, ('ai', 'z', 'st', 'iz')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=75272, /n=752 100-15 str=1400, /n=14 $\kappa = 1.05$	4 op=97633, /n=976 $\kappa = 1.002$
20	8 op=481448, /n=1204 400-20 str=6543, /n=16.4 $\kappa = 1.19$	5 op=592397, /n=1481 $\kappa = 1.03$
30	10 op=1582331, /n=1758 900-60-11 str=17618, /n=19.5 $\kappa = 1.38$	6 op=1949849, /n=2166 $\kappa = 1.08$
40	11 op=3365229, /n=2103 1600-120-14 str=34749, /n=21.8 $\kappa = 1.50$	7 op=4415917, /n=2759 $\kappa = 1.12$
50	13 op=6393915, /n=2558 2500-175-26-13 str=56497, /n=22.6 $\kappa = 1.81$	8 op=8188289, /n=3275 $\kappa = 1.25$
60	14 op=10046068, /n=2791 3600-210-32-16 str=82475, /n=22.9 $\kappa = 2.06$	9 op=13272893, /n=3687 $\kappa = 1.35$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 22
PCG for the anisotropic problem, $\tau = (1, 0.01)$, multilevel, ('ai', 'z', 'st', 'iz')

m	$\gamma = 1$	$\gamma = 2$
10	4 op=90038, /n=900 100-50-24-12 str=2497, /n=24.9 $\kappa = 1.008$	3 op=235557, /n=2356 $\kappa = 1.0001$
20	5 op=630197, /n=1575 400-200-66-33-14 str=13850, /n=34.6 $\kappa = 1.03$	3 op=1452157, /n=3630 $\kappa = 1.0008$
30	6 op=2305820, /n=2562 900-450-159-79-38-17 str=42850, /n=47.6 $\kappa = 1.05$	4 op=6502553, /n=7225 $\kappa = 1.002$
40	6 op=5028431, /n=3143 1600-800-251-126-58-29-15 str=92607, /n=57.9 $\kappa = 1.07$	4 op=14665673, /n=9166 $\kappa = 1.002$
50	7 op=10738437, /n=4295 2500-1250-414-207-81-34-15 str=172056, /n=68.8 $\kappa = 1.09$	4 op=27527546, /n=11011 $\kappa = 1.003$
60	7 op=18011516, /n=5003 3600-1800-616-308-126-63-31 str=288139, /n=80 $\kappa = 1.11$	4 op=49557238, /n=13766 $\kappa = 1.003$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 23
AMG for the discontinuous problem $\tau = 0.06$

$m = 10$	$m = 20$	$m = 30$
11	18	17
$\rho = 0.09$	$\rho = 0.25$	$\rho = 0.23$
op=135430, /n=1354	op=828773,/n=2072	op=1892192, /n=2102
100-50-16-11	400-200-46- -17-10	900-450-122- -30-21-17
$m = 39$	$m = 50$	$m = 59$
15	19	20
$\rho = 0.17$	$\rho = 0.27$	$\rho = 0.28$
op=2867393, /n=1835	op=5838352, /n=2335	op=8616620, /n=2475
1521-762-204- -57-31-21-16	2500-1250-324- -88-29-21-19	3481-1741-446- -122-41-22-17

TABLE 24
PCG for the discontinuous problem, $\tau = 0.06$, multilevel, ('gs', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=127378, /n=1274 100-50-16-11 str=1485, /n=14.8 $\kappa = 1.1$	4 op=302737, /n=3027 $\kappa = 1.002$
20	7 op=556267, /n=1391 400-200-46-17-10 str=5448, /n=13.6 $\kappa = 1.3$	4 op=1185685, /n=2964 $\kappa = 1.004$
30	8 op=1507599, /n=1675 900-450-122-30-21 str=13311, /n=14.8 $\kappa = 1.26$	4 op=2938191, /n=3265 $\kappa = 1.002$
39	7 op=2344515, /n=1541 1521-762-204-57-31-21-16 str=23115, /n=15.2 $\kappa = 1.21$	4 op=6702483, /n=4407 $\kappa = 1.002$
50	8 op=4197072, /n=1679 2500-1250-324-88-29-21 str=36892, /n=14.8 $\kappa = 1.35$	4 op=8669991, /n=3468 $\kappa = 1.005$
59	8 op=5931550, /n=1704 3481-1741-446-122-41-22-17 str=52033, /n=14.9 $\kappa = 1.35$	4 op=13191036, /n=3789 $\kappa = 1.004$

TABLE 25
PCG for the discontinuous problem, $\tau = 0.06$, multilevel, ('ic', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=114269, /n=1143 100-50-16-11 str=2306, /n=23.1 $\kappa = 1.03$	3 op=211467, /n=2115 $\kappa = 1.001$
20	7 op=510529, /n=1276 400-200-46-17-10 str=8491, /n=21.2 $\kappa = 1.2$	4 op=1061224, /n=2653 $\kappa = 1.003$
30	6 op=1062618, /n=1181 900-450-122-30-21 str=20515, /n=22.8 $\kappa = 1.15$	4 op=2599886, /n=2889 $\kappa = 1.001$
39	7 op=2126318, /n=1398 1521-762-204-57-31-21-16 str=35638, /n=23.4 $\kappa = 1.15$	4 op=5851962, /n=3847 $\kappa = 1.002$
50	7 op=3401485, /n=1361 2500-1250-324-88-29-21 str=56811, /n=22.7 $\kappa = 1.27$	4 op=7689215, /n=3076 $\kappa = 1.004$
59	7 op=4799520, /n=1379 3481-1741-446-122-41-22-17 str=80096, /n=23 $\kappa = 1.25$	4 op=11619184, /n=3338 $\kappa = 1.004$

TABLE 26
PCG for the discontinuous problem, $\tau = 0.06$, multilevel, ('ai', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=151474, /n=1515 100-50-16-11 str=3022, /n=30.2 $\kappa = 1.05$	4 op=319553, /n=3196 $\kappa = 1.002$
20	8 op=940565, /n=2351 400-200-46-17-10 str=13872, /n=34.7 $\kappa = 1.18$	5 op=1930373, /n=4826 $\kappa = 1.02$
30	7 op=2025525, /n=2251 900-450-122-30-21 str=33851, /n=37.6 $\kappa = 1.12$	4 op=4069393, /n=4521 $\kappa = 1.004$
39	7 op=3619203, /n=2379 1521-762-204-57-31-21-16 str=60219, /n=39.6 $\kappa = 1.13$	4 op=8407017, /n=5527 $\kappa = 1.004$
50	7 op=6042413, /n=2417 2500-1250-324-88-29-21 str=100118, /n=40 $\kappa = 1.26$	4 op=12772833, /n=5109 $\kappa = 1.006$
59	7 op=8611390, /n=2474 3481-1741-446-122-41-22-17 str=142531, /n=41 $\kappa = 1.23$	5 op=22832594, /n=6559 $\kappa = 1.005$

TABLE 27
PCG for the discontinuous problem, $\tau = 0.06$, multilevel, ('tw', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=117645, /n=1176 100-50-16-11 str=2593, /n=26 $\kappa = 1.12$	4 op=238513, /n=2385 $\kappa = 1.008$
20	8 op=515603, /n=1289 400-200-46-17-10 str=9515, /n=23.8 $\kappa = 1.33$	5 op=1140677, /n=2852 $\kappa = 1.01$
30	8 op=1235665, /n=1373 900-450-122-30-21 str=23150, /n=25.7 $\kappa = 1.29$	5 op=2820349, /n=3134 $\kappa = 1.01$
39	8 op=2161873, /n=1421 1521-762-204-57-31-21-16 str=40325, /n=26.5 $\kappa = 1.24$	5 op=6371455, /n=4189 $\kappa = 1.01$
50	9 op=3843983, /n=1537 2500-1250-324-88-29-21 str=64094, /n=25.6 $\kappa = 1.38$	5 op=8341373, /n=3336 $\kappa = 1.01$
59	9 op=5427707, /n=1559 3481-1741-446-122-41-22-17 str=90549, /n=26 $\kappa = 1.39$	5 op=12632543, /n=3629 $\kappa = 1.01$

TABLE 28
PCG for the discontinuous problem, $\tau = 0.06$, $\nu = 3$, multilevel, ('gc', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=298349, /n=2983 100-50-16-11 str=2773, /n=27.7 $\kappa = 1.09$	4 op=492485, /n=4925 $\kappa = 1.0004$
20	8 op=1317701, /n=3294 400-200-46-17-10 str=10188, /n=25.5 $\kappa = 1.19$	3 op=2012061, /n=5030 $\kappa = 1.001$
30	8 op=3095461, /n=3439 900-450-122-30-21 str=24673, /n=27.4 $\kappa = 1.17$	4 op=6017793, /n=6686 $\kappa = 1.001$
39	7 op=4799875, /n=3156 1521-762-204-57-31-21-16 str=42937, /n=28.2 $\kappa = 1.15$	4 op=13479577, /n=8862 $\kappa = 1.001$
50	9 op=9613203, /n=3845 2500-1250-324-88-29-21 str=68306, /n=27.3 $\kappa = 1.3$	4 op=17813393, /n=7125 $\kappa = 1.003$
59	8 op=12174245, /n=3497 3481-1741-446-122-41-22-17 str=96419, /n=27.7 $\kappa = 1.24$	4 op=26869257, /n=7719 $\kappa = 1.003$

TABLE 29
PCG for the discontinuous problem, $\tau = 0.06$, $\nu = 3$, multilevel, ('cg', 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	3 op=320353, /n=3203 100-50-16-11 str=3022, /n=30.2 $\kappa = 1.001$	2 op=729321, /n=7293 $\kappa = 1$
20	5 op=2361629, /n=5904 400-200-46-17-10 str=13872, /n=34.7 $\kappa = 1.02$	2 op=3753689, /n=9384 $\kappa = 1$
30	6 op=6654437, /n=7394 900-450-122-30-21 str=34851, /n=37.6 $\kappa = 1.05$	3 op=12546853, /n=13941 $\kappa = 1.0001$
39	5 op=10211191, /n=6713 1521-762-204-57-31-21-16 str=60219, /n=39.6 $\kappa = 1.03$	3 op=25861259, /n=17003 $\kappa = 1.0005$
50	6 op=19926938, /n=7971 2500-1250-324-88-29-21 str=100118, /n=40 $\kappa = 1.07$	3 op=39526405, /n=15811 $\kappa = 1.001$
59	7 op=32473523, /n=9329 3481-1741-446-122-41-22-17 str=142531, /n=40.9 $\kappa = 1.08$	4 op=73592777, /n=21141 $\kappa = 1.002$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 30
PCG for the discontinuous problem, $\tau = 0.06$, multilevel, ('po', 'a', 'st', 'st') $k = 1$

m	$\gamma = 1$	$\gamma = 2$
10	7 op=176045, /n=1760 100-50-16-11 str=1485, /n=14.8 $\kappa = 1.13$	4 op=364913, /n=3649 $\kappa = 1.002$
20	8 op=769349, /n=1923 400-200-46-17-10 str=5448, /n=13.6 $\kappa = 1.39$	4 op=1460913, /n=3652 $\kappa = 1.007$
30	8 op=1829485, /n=2033 900-450-122-30-21 str=13311, /n=14.8 $\kappa = 1.34$	4 op=3558513, /n=3954 $\kappa = 1.004$
39	8 op=3201265, /n=2105 1521-762-204-57-31-21-16 str=23115, /n=15.2 $\kappa = 1.26$	4 op=8053817, /n=5295 $\kappa = 1.004$
50	9 op=5683443, /n=2273 2500-1250-324-88-29-21 str=36892, /n=14.8 $\kappa = 1.41$	4 op=10528673, /n=4211 $\kappa = 1.008$
59	9 op=8020647, /n=2304 3481-1741-446-122-41-22-17 str=52033, /n=14.9 $\kappa = 1.45$	4 op=15955017, /n=4583 $\kappa = 1.007$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 31
PCG for the discontinuous problem, $\tau = 0.06$, multilevel, ('ai', 'z', 'st', 'iz')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=111261, /n=1126 100-15 str=1882, /n=18.8 $\kappa = 1.18$	5 op=154989, /n=1550 $\kappa = 1.03$
20	11 op=824933, /n=2062 400-44-13 str=8782, /n=21.9 $\kappa = 3.52$	8 op=1244657, /n=3111 $\kappa = 2.04$
30	12 op=2137984, /n=2375 900-100-28-13 str=21099, /n=23.4 $\kappa = 3.9$	9 op=3505293, /n=3895 $\kappa = 2.19$
39	14 op=4368457, /n=2872 1521-166-47-21-10 str=37356, /n=24.5 $\kappa = 4.99$	10 op=7164021, /n=4710 $\kappa = 2.68$
50	16 op=8321810, /n=3329 2500-271-51-22 str=62850, /n=25.1 $\kappa = 6.5$	11 op=12523013, /n=5009 $\kappa = 3.32$
59	17 op=12617568, /n=3625 3481-387-69-21-16 str=89907, /n=25.8 $\kappa = 7.9$	12 op=20087094, /n=5770 $\kappa = 3.8$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 32
PCG for the discontinuous problem, $\tau = (0.2, 0.06)$, multilevel, ('ai', 'z', 'st', 'iz')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=127051, /n=1270 100-40-15 str=2552, /n=25.5 $\kappa = 1.15$	4 op=218273, /n=2183 $\kappa = 1.01$
20	10 op=988121, /n=2470 400-147-53-23-10 str=11898, /n=29.7 $\kappa = 2.72$	6 op=1889889, /n=4725 $\kappa = 1.59$
30	10 op=2410906, /n=2679 900-304-117-53-24-12 str=28896, /n=32.2 $\kappa = 2.45$	6 op=6298817, /n=5887 $\kappa = 1.22$
39	11 op=4742431, /n=3118 1521-511-214-102-43-17 str=52274, /n=34.4 $\kappa = 2.9$	6 op=10171765, /n=6687 $\kappa = 1.31$
50	13 op=9484653, /n=3794 2500-837-356-174-67-31-17 str=89472, /n=35.8 $\kappa = 4.01$	7 op=21897205, /n=8759 $\kappa = 1.57$
59	13 op=13619971, /n=3913 3481-1163-510-248-88-40-17 str=128477, /n=36.9 $\kappa = 4.31$	8 op=36180434, /n=10390 $\kappa = 1.63$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 33
PCG for the discontinuous problem, $\tau = 0.06$, multilevel, ('ai', 'a', 'st', '-'), $\gamma = 1$

m	'iz'	'em'	'wi'	'wm'
10	7 op=212405, /n=2124 100-50-16 str=4098, /n=41	7 op=165429, /n=1654 100-50-14 str=2956, /n=29.6	7 op=161885, /n=1619 100-50-16-10 str=2795, /n=27.9	6 op=148681, /n=1487 100-50-16 str=3219, /n=32.2
20	10 op=1538429, /n=3846 400-200-45-20-12 str=20790, /n=52	9 op=987023, /n=2468 400-200-65-16 str=13329, /n=33.3	13 op=1276703, /n=3192 400-200-56-15 str=12198, /n=30.5	9 op=1058653, /n=2647 400-200-46-24-17 str=15034, /n=37.6
30	9 op=3663843, /n=4071 900-450-111-49-26-14 str=54708, /n=60.8	10 op=2665677, /n=2962 900-450-113-34-12 str=32528, /n=36.1	10 op=2552487, /n=2836 900-450-127-39-14 str=30958, /n=34.4	7 op=2539973, /n=2822 900-450-122-60-42-33-22 str=46329, /n=51.5
39	10 op=7407583, /n=4870 1521-762-189-70-36-21-14 str=100955, /n=66.4	12 op=5552781, /n=3651 1521-762-185-52-15 str=57033, /n=37.5	12 op=5294575, /n=3481 1521-762-210-52-17 str=54142, /n=35.6	8 op=4795489, /n=3153 1521-762-204-89-63 str=82054, /n=53.9
50	10 op=12459839, /n=4984 2500-1250-306-124-86 str=174490, /n=69.8	10 op=8210341, /n=3284 2500-1250-324-86-25 str=99116, /n=39.6	out of memory	8 op=8488763, /n=3395 2500-1250-324-120-80-58-37 str=136437, /n=54.6
59	11 op=19749155, /n=5673 3481-1741-421-166-105 str=254066, /n=72.9	13 op=14901069, /n=4281 3481-1741-433-116-30-10 str=140651, /n=40.4	out of memory	9 op=12671127, /n=3640 3481-1741-446-159-94-76 str=192518, /n=55.3

TABLE 34
AMG for the rapidly varying coefficient problem $\tau = 0.06$

$m = 10$	$m = 20$	$m = 30$
11	13	14
$\rho = 0.05$	$\rho = 0.13$	$\rho = 0.16$
op=124376, /n=1243	op=628323, /n=1571	op=1538752, /n=1710
100-50-14	400-200-52-16	900-450-119- -34-16-10

$m = 40$	$m = 50$	$m = 60$
14	16	17
$\rho = 0.17$	$\rho = 0.22$	$\rho = 0.23$
op=2745809, /n=1716	op=4835525, /n=1934	op=7471427, /n=2075
1600-800-207- -58-22-11	2500-1250-313- -78-23	3600-1800-450- -116-35-16-13

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 35
PCG for the rapidly varying coefficient problem, $\tau = 0.06$, multilevel, ('gs,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=111049, /n=1105 100-50-14 str=1285, /n=12.8 $\kappa = 1.05$	4 op=209633, /n=2096 $\kappa = 1.001$
20	6 op=509552, /n=1274 400-200-52-16 str=5653, /n=14.1 $\kappa = 1.08$	4 op=1158673, /n=2897 $\kappa = 1.002$
30	7 op=1393197, /n=1548 900-450-119-34-16-10 str=13243, /n=14.7 $\kappa = 1.11$	4 op=3388196, /n=3765 $\kappa = 1.001$
40	7 op=2453616, /n=1533 1600-800-207-58-22-11 str=23650, /n=14.8 $\kappa = 1.14$	4 op=5916142, /n=3698 $\kappa = 1.001$
50	7 op=3780647, /n=1512 2500-1250-313-78-23-15 str=36322, /n=14.5 $\kappa = 1.19$	4 op=8094016, /n=3238 $\kappa = 1.002$
60	8 op=6217905, /n=1727 3600-1800-450-116-35-16 str=52887, /n=14.7 $\kappa = 1.21$	4 op=12594385, /n=3498 $\kappa = 1.001$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 36
PCG for the rapidly varying coefficient problem, $\tau = 0.06$, multilevel, ('ic,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	5 op=85053, /n=850 100-50-14 str=1993, /n=19.9 $\kappa = 1.02$	3 op=147153, /n=1471 $\kappa = 1.0004$
20	5 op=385042, /n=963 400-200-52-16 str=8772, /n=21.8 $\kappa = 1.04$	3 op=795042, /n=1988 $\kappa = 1.0004$
30	6 op=1067074, /n=1191 900-450-119-34-16-10 str=20410, /n=22.7 $\kappa = 1.05$	3 op=2288596, /n=2543 $\kappa = 1.0005$
40	6 op=1905715, /n=1191 1600-800-207-58-22-11 str=36408, /n=22.7 $\kappa = 1.07$	3 op=4037610, /n=2524 $\kappa = 1.0005$
50	7 op=2251529, /n=1341 2500-1250-313-78-23-15 str=55886, /n=22.4 $\kappa = 1.11$	3 op=5551952, /n=2221 $\kappa = 1.0005$
60	7 op=4880849, /n=1356 3600-1800-450-116-35-16 str=81335, /n=22.6 $\kappa = 1.12$	3 op=8587888, /n=2385 $\kappa = 1.0004$

TABLE 37
PCG for the rapidly varying coefficient problem, $\tau = 0.06$, multilevel, ('ai,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=131937, /n=1319 100-50-14 str=2631, /n=26.3 $\kappa = 1.05$	4 op=237753, /n=2377 $\kappa = 1.002$
20	6 op=674612, /n=1686 400-200-52-16 str=13020, /n=32.5 $\kappa = 1.07$	4 op=1406753, /n=3517 $\kappa = 1.004$
30	7 op=1918237, /n=2131 900-450-119-34-16-10 str=32030, /n=35.6 $\kappa = 1.08$	4 op=3991393, /n=4435 $\kappa = 1.004$
40	7 op=3560860, /n=2225 1600-800-207-58-22-11 str=59273, /n=37 $\kappa = 1.09$	4 op=7464513, /n=4665 $\kappa = 1.004$
50	7 op=5669381, /n=2268 2500-1250-313-78-23-15 str=94133, /n=37.6 $\kappa = 1.15$	4 op=11390673, /n=4556 $\kappa = 1.005$
60	7 op=8365621, /n=2324 3600-1800-450-116-35-16 str=138741, /n=38.5 $\kappa = 1.17$	4 op=17532113, /n=4870 $\kappa = 1.004$

TABLE 38
PCG for the rapidly varying coefficient problem, $\tau = 0.06$, multilevel, ('tw,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=101973, /n=1020 100-50-14 str=2209, /n=22.1 $\kappa = 1.14$	4 op=164393, /n=1644 $\kappa = 1.01$
20	8 op=522398, /n=1306 400-200-52-16 str=9787, /n=24.5 $\kappa = 1.15$	5 op=1075205, /n=2688 $\kappa = 1.01$
30	8 op=1239904, /n=1378 900-450-119-34-16-10 str=22990, /n=25.5 $\kappa = 1.17$	5 op=3106333, /n=3451 $\kappa = 1.01$
40	8 op=2216574, /n=1385 1600-800-207-58-22-11 str=41072, /n=25.7 $\kappa = 1.19$	5 op=5491845, /n=3432 $\kappa = 1.01$
50	9 op=3786533, /n=1515 2500-1250-313-78-23-15 str=62958, /n=25.2 $\kappa = 1.25$	5 op=7525853, /n=3010 $\kappa = 1.01$
60	9 op=5517123, /n=1532 3600-1800-450-116-35-16 str=91732, /n=25.5 $\kappa = 1.28$	5 op=11657317, /n=3238 $\kappa = 1.01$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 39
PCG for the rapidly varying coefficient problem, $\tau = 0.06$, $\nu = 3$, multilevel, ('gc,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=258805, /n=1580 100-50-14 str=2373, /n=23.7 $\kappa = 1.1$	3 op=346405, /n=3461 $\kappa = 1.0005$
20	7 op=1169109, /n=2923 400-200-52-16 str=10455, /n=26.1 $\kappa = 1.10$	3 op=1856029, /n=4640 $\kappa = 1.0008$
30	7 op=2766621, /n=3074 900-450-119-34-16-10 str=24519, /n=27.2 $\kappa = 1.13$	3 op=5330853, /n=5923 $\kappa = 1.0007$
40	8 op=5547096, /n=3467 1600-800-207-58-22-11 str=43770, /n=27.3 $\kappa = 1.17$	3 op=9357341, /n=5848 $\kappa = 1.0007$
50	9 op=9467073, /n=3787 2500-1250-313-78-23-15 str=67172, /n=26.8 $\kappa = 1.32$	3 op=12924165, /n=5170 $\kappa = 1.0006$
60	9 op=13777663, /n=3827 3600-1800-450-116-35-16 str=97749, /n=27.1 $\kappa = 1.33$	3 op=19955869, /n=5543 $\kappa = 1.0006$

TABLE 40
PCG for the rapidly varying coefficient problem, $\tau = 0.06$, $\nu = 3$, multilevel, (‘cg,’ ‘a’, ‘st’, ‘st’)

m	$\gamma = 1$	$\gamma = 2$
10	3 op=275109, /n=2751 100-50-14 str=2631, /n=26.3 $\kappa = 1.0001$	2 op=541525, /n=5414 $\kappa = 1$
20	4 op=1793498, /n=4484 400-200-52-16 str=13020, /n=32.6 $\kappa = 1.004$	2 op=3240569, /n=8101 $\kappa = 1$
30	5 op=5382018, /n=5980 900-450-119-34-16-10 str=32030, /n=35.6 $\kappa = 1.01$	2 op=9227881, /n=10253 $\kappa = 1$
40	5 op=10007103, /n=6254 1600-800-207-58-22-11 str=59272, /n=37 $\kappa = 1.01$	2 op=17271369, /n=10795 $\kappa = 1$
50	5 op=15952841, /n=6381 2500-1250-313-78-23-15 str=94133, /n=37.6 $\kappa = 1.01$	2 op=26438009, /n=10575 $\kappa = 1$
60	5 op=23565355, /n=6546 3600-1800-450-116-35-16 str=138741, /n=38.5 $\kappa = 1.02$	2 op=40711657, /n=11309 $\kappa = 1$

TABLE 41
PCG for the rapidly varying coefficient problem, $\tau = 0.06$, multilevel, ('po', 'a', 'st', 'st') $k = 1$

m	$\gamma = 1$	$\gamma = 2$
10	6 op=132105, /n=1321 100-50-14 str=1285, /n=12.8 $\kappa = 1.08$	3 op=200037, /n=2004 $\kappa = 1.002$
20	7 op=687189, /n=1718 400-200-52-16 str=5653, /n=14.1 $\kappa = 1.13$	4 op=1359953, /n=3400 $\kappa = 1.002$
30	8 op=1837738, /n=2042 900-450-119-34-16-10 str=13243, /n=14.7 $\kappa = 1.17$	4 op=3941873, /n=4380 $\kappa = 1.002$
40	8 op=3279600, /n=2050 1600-800-207-58-22-11 str=23650, /n=14.8 $\kappa = 1.20$	4 op=6942913, /n=4339 $\kappa = 1.002$
50	8 op=5029811, /n=2012 2500-1250-313-78-23-15 str=36322, /n=14.5 $\kappa = 1.26$	4 op=9504033, /n=3802 $\kappa = 1.002$
60	9 op=8149743, /n=2264 3600-1800-450-116-35-16 str=52887, /n=14.7 $\kappa = 1.29$	4 op=14726433, /n=4091 $\kappa = 1.002$

TABLE 42
PCG for the rapidly varying coefficient problem, $\tau = 0.06$, multilevel, ('ai', 'z', 'st', 'iz')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=104509, /n=1045 100-13 str=1737, /n=17.4 $\kappa = 1.21$	4 op=120433, /n=1204 $\kappa = 1.03$
20	9 op=655513, /n=1649 400-46-15 str=8464, /n=21.2 $\kappa = 1.6$	6 op=929041, /n=2322 $\kappa = 1.16$
30	11 op=1928857, /n=2143 900-103-29-13 str=20670, /n=22.9 $\kappa = 1.8$	7 op=2764629, /n=3072 $\kappa = 1.23$
40	12 op=3849758, /n=2406 1600-180-37-15 str=38032, /n=23.8 $\kappa = 2.00$	8 op=5709825, /n=3568 $\kappa = 1.29$
50	13 op=6677681, /n=2671 2500-279-52-23-10 str=61230, /n=24.5 $\kappa = 2.15$	8 op=9535625, /n=3813 $\kappa = 1.34$
60	14 op=10475023, /n=2910 3600-402-66-28-12 str=89631, /n=24.9 $\kappa = 2.31$	9 op=15479973, /n=4300 $\kappa = 1.39$

TABLE 43
PCG for the rapidly varying coefficient problem, $\tau = (1, 0.01)$, multilevel, ('ai', 'z', 'st', 'iz')

m	$\gamma = 1$	$\gamma = 2$
10	4 op=175818, /n=1758 100-50-24-14 str=4726, /n=47.2 $\kappa = 1.008$	3 op=403493, /n=4035 $\kappa = 1.0001$
20	5 op=1631519, /n=4079 400-200-99-53-26-12 str=35029, /n=87 $\kappa = 1.05$	3 op=3848317, /n=9621 $\kappa = 1.002$
30	6 op=5849563, /n=6499 900-450-215-111-54-23-10 str=106604, /n=118 $\kappa = 1.11$	4 op=14671353, /n=16302 $\kappa = 1.009$
40	7 op=14237629, /n=8898 1600-800-385-199-104-45-22 str=226485, /n=141 $\kappa = 1.18$	4 op=31440793, /n=19650 $\kappa = 1.02$
50	8 op=27883151, /n=11153 2500-1250-605-314-157-72-37 str=393652, /n=157 $\kappa = 1.30$	5 op=65753549, /n=26301 $\kappa = 1.03$
60	8 op=43150516, /n=11986 3600-1800-875-454-233-108-53 str=608523, /n=169 $\kappa = 1.47$	5 op=102536912, /n=28482 $\kappa = 1.05$

TABLE 44
AMG for the random Laplacian problem $\tau = 0.06$

$m = 10$	$m = 20$	$m = 30$
9	10	10
$\rho = 0.06$	$\rho = 0.08$	$\rho = 0.08$
op=101687, /n=1017	op=719629, /n=1799	op=2717419, /n=3019
100-50	400-200	900-450
$m = 40$	$m = 50$	$m = 60$
10	11	
$\rho = 0.08$	$\rho = 0.14$	
op=6817639, /n=4261	op=8259634, /n=3304	
1600-800	2500-1250-382	

TABLE 45
PCG for the random Laplacian problem, $\tau = 0.06$, multilevel, ('gs,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=101851, /n=1018 100-50-19 str=1402, /n=14 $\kappa = 1.05$	4 op=185873, /n=1859 $\kappa = 1.002$
20	6 op=436591, /n=1091 400-200-67 str=5913, /n=14.8 $\kappa = 1.06$	4 op=813933, /n=2035 $\kappa = 1.002$
30	6 op=1021775, /n=1135 900-450-141 str=13514, /n=15 $\kappa = 1.06$	4 op=1934273, /n=2149 $\kappa = 1.002$
40	6 op=1875813, /n=1172 1600-800-254 str=24226, /n=15.1 $\kappa = 1.10$	4 op=3599433, /n=2250 $\kappa = 1.002$
50	6 op=3009093, /n=1204 2500-1250-383 str=37734, /n=15.1 $\kappa = 1.07$	4 op=5839113, /n=2336 $\kappa = 1.002$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 46
PCG for the random Laplacian problem, $\tau = 0.06$, multilevel, ('ic,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	5 op=81665, /n=816 100-50-19 str=2109, /n=21.1 $\kappa = 1.03$	3 op=138140, /n=1381 $\kappa = 1.001$
20	6 op=408899, /n=1022 400-200-67 str=8865, /n=22.2 $\kappa = 1.04$	3 op=604333, /n=1511 $\kappa = 1.001$
30	6 op=956647, /n=1063 900-450-141 str=20204, /n=22.4 $\kappa = 1.04$	3 op=1436677, /n=1596 $\kappa = 1.001$
40	6 op=1759473, /n=1100 1600-800-254 str=36234, /n=22.6 $\kappa = 1.04$	3 op=2682133, /n=1672 $\kappa = 1.001$
50	6 op=2823269, /n=1129 2500-1250-383 str=56510, /n=22.6 $\kappa = 1.04$	3 op=4354349, /n=1742 $\kappa = 1.001$

TABLE 47
PCG for the random Laplacian problem, $\tau = 0.06$, multilevel, ('ai,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	9 op=106723, /n=1067 100-50-19 str=1730, /n=17.3 $\kappa = 1.3$	6 op=135213, /n=1352 $\kappa = 1.06$
20	9 op=517643, /n=1294 400-200-67 str=7322, /n=18.3 $\kappa = 1.4$	6 op=666989, /n=1667 $\kappa = 1.08$
30	10 op=1489546, /n=1655 900-450-141 str=16694, /n=18.5 $\kappa = 1.39$	6 op=1763173, /n=1959 $\kappa = 1.08$
40	10 op=3003385, /n=1877 1600-800-254 str=29952, /n=18.7 $\kappa = 1.44$	7 op=4102413, /n=2564 $\kappa = 1.10$
50	10 op=4049613, /n=1620 2500-1250-383 str=51968, /n=20.7 $\kappa = 1.45$	7 op=7713653, /n=3085 $\kappa = 1.10$

TABLE 48
PCG for the random Laplacian problem, $\tau = 0.06$, multilevel, ('tw,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	7 op=97477, /n=975 100-50-19 str=2309, /n=23.1 $\kappa = 1.10$	4 op=153153, /n=1531 $\kappa = 1.007$
20	7 op=420317, /n=1051 400-200-67 str=9816, /n=24.5 $\kappa = 1.11$	4 op=677233, /n=1693 $\kappa = 1.008$
30	7 op=987589, /n=1097 900-450-141 str=22420, /n=24.9 $\kappa = 1.12$	4 op=1619793, /n=1800 $\kappa = 1.007$
40	7 op=1822893, /n=1139 1600-800-254 str=40282, /n=25.2 $\kappa = 1.18$	4 op=3039833, /n=1900 $\kappa = 1.008$
50	7 op=2932981, /n=1173 2500-1250-382 str=62888, /n=25.1 $\kappa = 1.13$	4 op=4954233, /n=1982 $\kappa = 1.008$

TABLE 49
PCG for the random Laplacian problem, $\tau = 0.06$, multilevel, ('gc,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	6 op=217603, /n=2176 100-50-19 str=1402, /n=14 $\kappa = 1.07$	3 op=326661, /n=3267 $\kappa = 1.0004$
20	6 op=906403, /n=2266 400-200-67 str=5913, /n=14.8 $\kappa = 1.08$	3 op=1375261, /n=3438 $\kappa = 1.0006$
30	6 op=2086895, /n=2319 900-450-141 str=13514, /n=15 $\kappa = 1.09$	3 op=3192357, /n=3547 $\kappa = 1.0007$
40	7 op=4313517, /n=2696 1600-800-254 str=24226, /n=15.1 $\kappa = 1.14$	3 op=5804669, /n=3628 $\kappa = 1.0007$
50	7 op=6842773, /n=2737 2500-1250-383 str=37734, /n=15.1 $\kappa = 1.14$	3 op=9266661, /n=3707 $\kappa = 1.0008$

TABLE 50
PCG for the random Laplacian problem, $\tau = 0.06$, $\nu = 3$, multilevel, ('cg,' 'a', 'st', 'st')

m	$\gamma = 1$	$\gamma = 2$
10	4 op=155828, /n=1558 100-50-19 str=1730, /n=17.3 $\kappa = 1.001$	2 op=180957, /n=1810 $\kappa = 1$
20	4 op=677228, /n=1695 400-200-67 str=7322, /n=18.3 $\kappa = 1.003$	2 op=788861, /n=1972 $\kappa = 1$
30	4 op=1626088, /n=1807 900-450-141 str=16694, /n=18.5 $\kappa = 1.002$	2 op=4895797, /n=2106 $\kappa = 1$
40	4 op=3058568, /n=1912 1600-800-254 str=29952, /n=18.7 $\kappa = 1.003$	2 op=3571317, /n=2232 $\kappa = 1$
50	4 op=5771853, /n=2309 2500-1250-383 str=51698, /n=20.7 $\kappa = 1.004$	2 op=9146465, /n=3659 $\kappa = 1$

TABLE 51
PCG for the random Laplacian problem, $\tau = 0.06$, multilevel, ('po', 'a', 'st', 'st') $k = 1$

m	$\gamma = 1$	$\gamma = 2$
10	6 op=125819, /n=1258 100-50-19 str=1402, /n=14 $\kappa = 1.07$	3 op=185669, /n=1857 $\kappa = 1.002$
20	6 op=532379, /n=1331 400-200-67 str=5913, /n=14.8 $\kappa = 1.10$	3 op=799069, /n=1998 $\kappa = 1.002$
30	7 op=1418277, /n=1576 900-450-141 str=13514, /n=15 $\kappa = 1.13$	3 op=1883749, /n=2093 $\kappa = 1.002$
40	7 op=2589037, /n=1618 1600-800-254 str=24226, /n=15.1 $\kappa = 1.24$	4 op=4352953, /n=2721 $\kappa = 1.006$
50	7 op=4138709, /n=1655 2500-1250-382 str=37734, /n=15.1 $\kappa = 1.16$	3 op=5611813, /n=2244 $\kappa = 1.003$

TABLE 52
PCG for the random Laplacian problem, $\tau = 0.06$, multilevel, ('ai,' 'z', 'st', 'iz')

m	$\gamma = 1$	$\gamma = 2$
10	9 op=146023, /n=1460 100-50-24-12 str=2134, /n=21.3 $\kappa = 1.4$	6 op=335049, /n=3350 $\kappa = 1.09$
20	10 op=731590, /n=1829 400-200-91-46-23-12 str=9489, /n=23.7 $\kappa = 1.4$	7 op=2729133, /n=6823 $\kappa = 1.1$
30	11 op=1850653, /n=2056 900-450-204-96-49-25-13 str=21533, /n=24.4 $\kappa = 1.6$	7 op=7510421, /n=8345 $\kappa = 1.15$
40	12 op=3624403, /n=2265 1600-800-381-176-90-40-20 str=39655, /n=24.8 $\kappa = 1.9$	8 op=15291297, /n=9557 $\kappa = 1.27$
50	12 op=5724579, /n=2290 2500-1250-595-288-129-65-36 str=62822, /n=25.1 $\kappa = 1.8$	8 op=24772913, /n=9909 $\kappa = 1.24$

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 53
1138-bus modified, $\tau = 0.06$

method	
AMG	104 it op=12227373 1138-472-178-80-61 $\rho = 0.81$ st=24922
(‘gs’, ‘a’, ‘st’, ‘st’)	22 it op=3880407 1138-472-178-80 st=12533
(‘ic’, ‘a’, ‘st’, ‘st’)	21 it op=3351650 1138-472-178-80 st=19993
(‘ai’, ‘a’, ‘st’, ‘st’)	14 it op=2808576 1138-472-178-80 st=24922
(‘tw’, ‘a’, ‘st’, ‘st’)	109 it op=14886591 1138-472-178-80 st=21849
(‘gc’, ‘a’, ‘st’, ‘st’)	63 it op=12546321 1138-472-178-80 st=23717
(‘cg’, ‘a’, ‘st’, ‘st’)	13 it op=5058207 1138-472-178-80 st=24922
(‘po’, ‘a’, ‘st’, ‘st’)	37 it op=7645767 1138-472-178-80 st=12533
(‘ai’, ‘z’, ‘st’, ‘iz’)	15 it op=2402769 1138-472-178-80 st=19597

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 54
bcsstk01, $\tau = 0.06$

method	
AMG	48 it op=431347 48-18 $\rho = 0.98$ st=1179
(‘gs’, ‘a’, ‘st’, ‘st’)	13 it op=110367 48-18 st=736
(‘ic’, ‘a’, ‘st’, ‘st’)	11 it op=84989 48-18 st=1144
(‘ai’, ‘a’, ‘st’, ‘st’)	13 it op=106783 48-18 st=1179
(‘tw’, ‘a’, ‘st’, ‘st’)	14 it op=98654 48-18 st=1354
(‘gc’, ‘a’, ‘st’, ‘st’)	48 it op=405284 48-18 st=1420
(‘cg’, ‘a’, ‘st’, ‘st’)	17 it op=237955 48-18 st=1179
(‘po’, ‘a’, ‘st’, ‘st’)	32 it op=301868 48-18 st=736
(‘ai’, ‘z’, ‘st’, ‘iz’) $\tau = 0.2$	14 it op=92294 48-10 st=867

TABLE 55
gr3030, $\tau = 0.06$

method	
AMG	10 it op=1182199 900-225-49 $\rho = 0.08$ st=26313
(‘gs’, ‘a’, ‘st’, ‘st’)	6 it op=1140515 900-225-49 st=12461
(‘ic’, ‘a’, ‘st’, ‘st’)	6 it op=1036929 900-225-49 st=19199
(‘ai’, ‘a’, ‘st’, ‘st’)	7 it op=1600109 900-225-49 st=26313
(‘tw’, ‘a’, ‘st’, ‘st’)	7 it op=1098301 900-225-49 st=22415
(‘gc’, ‘a’, ‘st’, ‘st’)	17 it op=3188309 900-225-49 st=23589
(‘cg’, ‘a’, ‘st’, ‘st’)	8 it op=3362081 900-225-49 st=26313
(‘po’, ‘a’, ‘st’, ‘st’)	8 it op=1767173 900-225-49 st=12461
(‘ai’, ‘z’, ‘st’, ‘iz’)	9 it op=1916455 900-117-33-14 st=24975

TABLE 56
bcsstk34, $\tau = 0.01$

method	
AMG	6 it op=1343689 588-66-12 $\rho = 0.008$ st=32921
('gs', 'a', 'st', 'st')	3 it op=1326425 588-66-12 st=25393
('ic', 'a', 'st', 'st')	3 it op=1146281 588-66-12 st=37855
('ai', 'a', 'st', 'st')	5 it op=1491137 588-66-12 st=32921
('tw', 'a', 'st', 'st')	4 it op=1412033 588-66-12 st=48319
('gc', 'a', 'st', 'st')	6 it op=2151621 588-66-12 st=23847
('cg', 'a', 'st', 'st')	5 it op=2490353 588-66-12 st=32921
('po', 'a', 'st', 'st')	6 it op=2717445 588-66-12 st=25393
('ai', 'z', 'st', 'iz')	4 it op=1982563 588-233-24 st=58317

TABLE 57
bcsstk27, $\tau = 0.06$

method	
('gs', 'a', 'st', 'st')	58 it op=61447065 1224-289-34 st=73840
('ai', 'a', 'st', 'st') $\tau = 0.06$	368 it op=312558975 1224-289-1234 st=110083
('ai', 'a', 'st', 'st') $\tau(= 0.06, 0.01)$	22 it op=45313063 1224-289-1234 st=251107
('ai', 'a', 'st', 'st') $\tau = (0.06, 0.005)$	12 it op=34343127 1224-289-1234 st=335578
('tw', 'a', 'st', 'st')	274 it op=231271769 1224-289-34 st=142813

5. Conclusion. In this paper we have compared several fully algebraic multilevel preconditioners for PCG. We used several different smoothers and ways to define the coarse grids as well as several interpolation schemes. It seems the most important point to obtain good results is the smoother. There is not much difference when changing the way to compute the coarse grids as well as the interpolation scheme as long as the interpolation of a constant is a constant. The symmetric Gauss-Seidel and the incomplete Cholesky decomposition are good smoothers but they are not parallel. The approximate inverses are fully parallel. The Wang and Tan proposal is a better smoother than AINV for most problems (at least when using the same amount of storage) but there is no guarantee to obtain a positive definite preconditioner.

There is no overall best algorithm for our set of examples. Everything depends on what we are looking for: smallest number of iterations, smallest computer time, smallest storage, best performance on a parallel computer, etc... For instance, AINV is not the best smoother but it is fully parallel; the 'z' influence matrix does not give the smallest number of operations but the storage is generally smaller than for the other methods. A very nice thing with AINV is that we just have to adjust one parameter to obtain a better smoother by reducing the value of τ although this increases the storage.

Almost all these methods give a constant number of iterations when the problem size is increasing for the partial differential equations we solved and some of them give a number of operations proportional to the problem size. It remains to test some of these methods on parallel computers with much larger problems to see if we still get the scalability we are looking for.

Acknowledgments. The author thanks Michele Benzi for interesting conversations about using AINV.

REFERENCES

[1] M. BENZI, C.D. MEYER AND M. TUMA, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135-1149.
[2] W.L. BRIGGS, V.E. HENSON AND S.F. MCCORMICK, *A multigrid tutorial, second edition*, SIAM, 2000.

[3] A. CLEARY, R. FALGOUT, V.E. HENSON AND J. JONES, *Coarse–grid selection for parallel algebraic multigrid*, to appear in LLNL report, 1999.

[4] V.E. HENSON, *An algebraic multigrid tutorial*, MGNET, 1999. Available online from http://www.mgnet.org.

[5] G. MEURANT, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 707-728.

[6] G. MEURANT, *Computer Solution of Large Linear Systems*, North–Holland, 1999.

[7] G. MEURANT, *A multilevel AINV preconditioner*, accepted in Numer. Algorithms (2001).

[8] J.W. RUGE AND K. STUBEN, *Algebraic multigrid*, in Multigrid methods, S.F. Mc Cormick, ed., pp. 73-130, SIAM, 1987.

[9] W–P. TANG AND W–L. WAN, *Sparse approximate inverse smoother for multigrid,* to appear.

[10] C. WAGNER, *Introduction to algebraic multigrid*, Course notes version 1.1, University of Heidelberg, 1998.

[11] C. WAGNER, *On the algebraic construction of multilevel transfer operators*, Heidelberg University, 1999. Available online from http://www.iwr.uni-heidelberg.de.

[12] W.L. WAN, T.F. CHAN & B. SMITH, *An energy minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.