MULTI-SCALE LINEAR SOLVERS FOR VERY LARGE SYSTEMS DERIVED FROM PDES[*]

KLAUS LACKNER[†] AND RALPH MENIKOFF[‡]

Abstract. We present a novel linear solver that works well for large systems obtained from discretizing PDEs. It is robust and, for the examples we studied, the computational effort scales linearly with the number of equations. The algorithm is based on a wavelength decomposition that combines conjugate gradient, multi-scaling and iterative splitting methods into a single approach. On the surface, the algorithm is a simple preconditioned conjugate gradient with all the sophistication of the algorithm in the choice of the preconditioning matrix. The preconditioner is a very good approximate inverse of the linear operator. It is constructed from the inverse of the coarse grained linear operator and from smoothing operators that are based on an operator splitting on the fine grid. The coarse graining captures the long wavelength behavior of the inverse operator while the smoothing operator captures the short wavelength behavior. The conjugate gradient iteration accounts for the coupling between long and short wavelengths. The coarse grained operator corresponds to a lower resolution approximation to the PDEs. While the coarse grained inverse is not known explicitly, the algorithm only requires that the preconditioner can be a applied to a vector. The coarse inverse applied to a vector can be obtained as the solution of another preconditioned conjugate gradient solver that applies the same algorithm to the smaller problem. Thus, the method is naturally recursive. The recursion ends when the matrix is sufficiently small for a solution to be obtained efficiently with a standard solver. The local feedback provided by the conjugate gradient step at every level makes the algorithm very robust. In spite of the effort required for the coarse inverse, the algorithm is efficient because the increased quality of the approximate inverse greatly reduces the number of times the preconditioner needs to be evaluated. A feature of the algorithm is that the transition between coarse grids is determined dynamically by the accuracy requirement of the conjugate gradient solver at each level. Typically, later iterations on the finer scales need fewer iterations on the coarser scales and the computational effort is proportional to N rather than $N \log N$, where N is the number of equations. We have tested our solver on the porous flow equation. On a workstation we have solved problems on grids ranging in dimension over 3 orders of magnitude, from 10^3 to 10^6, and found that the linear scaling holds. The algorithm works well, even when the permeability tensor has spatial variations exceeding a factor of 10^9.

Key words. multigrid, conjugate gradient, linear solver, porous flow

AMS subject classifications. 65F10, 65N22, 65N55

1. Introduction.

We present a novel linear solver that works well for large systems obtained from discretizing PDEs. It is robust and for the examples we studied, the computational effort scales linearly with the number of equations. The algorithm is based on a wavelength decomposition which combines conjugate gradient, multi-scaling and iterative splitting methods into a single approach. On the surface, the algorithm is a simple preconditioned conjugate gradient with all the sophistication of the algorithm in the novel choice of the preconditioning matrix. The preconditioner is an approximate inverse of the linear operator constructed from the inverse of the coarse grained linear operator and a smoothing operator based on an iterative expansion of the operator on the fine grid. To transform the coarse grained operator to the fine grid, it is pre- and post-multiplied by restriction and prolongation operators. The coarse graining captures the long wavelength behavior of the inverse operator

[*]THIS WORK WAS PRESENTED AT THE FIFTH COPPER MOUNTAIN CONFERENCE ON ITERATIVE METHODS, APRIL, 1998.

[†]Theoretical Division, Mail Stop B-216, Los Alamos National Laboratory, Los Alamos, NM 87544 (`ksl@lanl.gov`).

[‡]Theoretical Division, Mail Stop B-214, Los Alamos National Laboratory, Los Alamos, NM 87544 (`rtm@lanl.gov`).

while the smoothing operator captures the short wavelength behavior. The conjugate gradient iteration accounts for the coupling between long and short wavelengths.

The coarse grained operator corresponds to a lower resolution approximation to the PDEs. While its inverse is not known explicitly, the conjugate gradient algorithm only requires that the preconditioner can be applied to a vector. The inverse is obtained as the solution of another preconditioned conjugate gradient solver that applies the same algorithm to the smaller problem. Thus, the method is naturally recursive. The recursion ends when the matrix is sufficiently small for a solution to be obtained efficiently with a standard solver.

Basing a preconditioner on the inverse of a simpler problem is akin to using an incomplete factorization. As with an incomplete factorization, our preconditioner only requires knowledge of sparse matrices. But rather than a simple forward and backward substitution used to evaluate the inverse of the product of a lower and upper triangular matrix, our preconditioner requires an iterative method. Despite the extra effort required for the coarse inverse, the algorithm is efficient because the increased quality of the approximate inverse greatly reduces the number of times the preconditioner needs to be evaluated. In the cases we studied the computational effort is dominated by the operations performed on the finest scale.

As with the multigrid method, by utilizing several scales we obtain rapid convergence of long wavelengths. In contrast to the way multigrid algorithms are typically applied, the recursive use of the conjugate gradient algorithm at each stage enforces an accurate transition from one scale to the next. In particular, the algorithm naturally accounts for the coupling between the long and short wavelengths introduced at every level of refinement. Thus, it avoids the accumulation of errors that stem from the interaction terms between the different levels. In addition, the transition between levels is determined dynamically rather than preprogrammed with a V-cycle or W-cycle. The algorithm is robust and works well without fine tuning even when the PDE is far from diagonal in Fourier space.

2. Algorithm.

Let the linear system be given by

$$\boldsymbol{A}x = b \,. \tag{2.1}$$

With a preconditioning matrix $\boldsymbol{M}^{-1}$, the system to be solved is

$$(\boldsymbol{M}^{-1}\boldsymbol{A})x = \boldsymbol{M}^{-1}b \,. \tag{2.2}$$

Our aim is to determine a good approximate inverse of $\boldsymbol{A}$ that can be used for $\boldsymbol{M}^{-1}$. This reduces the condition number of the system and hence enhances the convergence rate of any iterative solver. We use a conjugate gradient algorithm since it has the advantage that convergence can greatly be enhanced even if the preconditioner fails to account for a small number of modes.

To develop an approximate inverse, we start with a standard operator splitting of the form, see for example [2],

$$\boldsymbol{A} = \boldsymbol{P} - \boldsymbol{Q} \,. \tag{2.3}$$

We assume that $\boldsymbol{A}$ and $\boldsymbol{P}$ are symmetric and positive definite. The inverse is given formally by

$$\boldsymbol{A}^{-1} = (\boldsymbol{I} - \boldsymbol{P}^{-1}\boldsymbol{Q})^{-1}\boldsymbol{P}^{-1} = \sum_{n=0}^{\infty}(\boldsymbol{P}^{-1}\boldsymbol{Q})^n\boldsymbol{P}^{-1} \,. \tag{2.4}$$

The series converges when $||\boldsymbol{P}^{-1}\boldsymbol{Q}|| < 1$, where $||\cdot||$ denotes the L^2 matrix norm. Truncating the series after n terms defines an approximate inverse

$$\widetilde{\boldsymbol{A}}_n^{-1} = \sum_{k=0}^{n}(\boldsymbol{P}^{-1}\boldsymbol{Q})^k\boldsymbol{P}^{-1} . \tag{2.5}$$

Moreover, $\widetilde{\boldsymbol{A}}_n^{-1}$ is positive definite if $||\boldsymbol{P}^{-1}\boldsymbol{Q}|| < 1$. Frequently, the matrix arising from discretizing a PDE is diagonally dominated and it can be shown that standard splittings, such as Jacobi or Gauss-Seidel, do indeed satisfy $||\boldsymbol{P}^{-1}\boldsymbol{Q}|| < 1$. For these systems with the Jacobi splitting or the Gauss-Seidel splitting $\widetilde{\boldsymbol{A}}_n^{-1}$ is symmetric positive definite and can be used as a preconditioner for the conjugate gradient algorithm. However, $||\boldsymbol{P}^{-1}\boldsymbol{Q}||$ typically approaches 1 as the resolution of the discretization increases. Then, the convergence of the series for $\boldsymbol{A}^{-1}$ is very slow and $\widetilde{\boldsymbol{A}}_n^{-1}$ is not an effective preconditioner.

We can construct a better preconditioner with the aid of the following identity

$$\boldsymbol{A}^{-1} = (\boldsymbol{P}^{-1}\boldsymbol{Q})^m\boldsymbol{A}^{-1}(\boldsymbol{Q}\boldsymbol{P}^{-1})^m + \sum_{k=0}^{2m-1}(\boldsymbol{P}^{-1}\boldsymbol{Q})^k\boldsymbol{P}^{-1} \tag{2.6}$$

for any integer $m \geq 1$. This identity can be proved as follows. From the definition of the splitting, (2.3), we obtain

$$(\boldsymbol{Q}\boldsymbol{P}^{-1})\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{P}^{-1}\boldsymbol{Q}) . \tag{2.7}$$

Multiplying the first term on the right-hand side of (2.6) by $\boldsymbol{A}$ we obtain

$$(\boldsymbol{P}^{-1}\boldsymbol{Q})^m\boldsymbol{A}^{-1}(\boldsymbol{Q}\boldsymbol{P}^{-1})^m\boldsymbol{A} = (\boldsymbol{P}^{-1}\boldsymbol{Q})^m\boldsymbol{A}^{-1}\boldsymbol{A}(\boldsymbol{P}^{-1}\boldsymbol{Q})^m = (\boldsymbol{P}^{-1}\boldsymbol{Q})^{2m} .$$

Multiplying the second term on the right-hand side of (2.6) by $\boldsymbol{A}$ we obtain

$$\sum_{k=0}^{2m-1}(\boldsymbol{P}^{-1}\boldsymbol{Q})^k\boldsymbol{P}^{-1}(\boldsymbol{P}-\boldsymbol{Q}) = \sum_{k=0}^{2m-1}(\boldsymbol{P}^{-1}\boldsymbol{Q})^k - \sum_{k=1}^{2m}(\boldsymbol{P}^{-1}\boldsymbol{Q})^k = \boldsymbol{I} - (\boldsymbol{P}^{-1}\boldsymbol{Q})^{2m} .$$

The sum of these two equations yields the identity on the right-hand side. Hence the right-hand side of (2.6) is $\boldsymbol{A}^{-1}$.

Equation (2.6) forms the basis for combining two approximate inverses of $\boldsymbol{A}$ into one that is better than either. The second term on the right-hand side of (2.6) is the first approximate inverse, $\widetilde{\boldsymbol{A}}_{2m-1}^{-1}$. The first term on the right-hand side of (2.6) represents the error in using $\widetilde{\boldsymbol{A}}_{2m-1}^{-1}$ as an approximate inverse for $\boldsymbol{A}$. Substituting a second approximate inverse $\boldsymbol{W}$ for $\boldsymbol{A}^{-1}$ in the error term can be expected to yield an improved approximate inverse $\boldsymbol{M}^{-1}$,

$$\boldsymbol{M}^{-1} = (\boldsymbol{P}^{-1}\boldsymbol{Q})^m\boldsymbol{W}(\boldsymbol{Q}\boldsymbol{P}^{-1})^m + \sum_{k=0}^{2m-1}(\boldsymbol{P}^{-1}\boldsymbol{Q})^k\boldsymbol{P}^{-1} . \tag{2.8}$$

Under reasonable assumptions $\boldsymbol{M}^{-1}$ is a better approximate inverse than either $\boldsymbol{W}$ or $\widetilde{\boldsymbol{A}}_{2m-1}^{-1}$.

To measure how well $\boldsymbol{M}^{-1}$ approximates $\boldsymbol{A}^{-1}$ we use the quantity

$$\rho(\boldsymbol{M}^{-1}) = ||\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}||_{\boldsymbol{A}} .$$

If $\rho < 1$, an upper bound on the condition number of the preconditioned operator $\boldsymbol{M}^{-1}\boldsymbol{A}$ can be obtained in terms of ρ, see for example [10, Proposition 2.2]. The convergence rate of the conjugate gradient algorithm is related to the condition number. Consequently, ρ provides a bound on the convergence rate. Smaller values of ρ correspond to better convergence rates.

To compute $\rho(\boldsymbol{M}^{-1})$ we begin by subtracting (2.8) from (2.6) which yields

$$\boldsymbol{A}^{-1} - \boldsymbol{M}^{-1} = (\boldsymbol{P}^{-1}\boldsymbol{Q})^m(\boldsymbol{A}^{-1} - \boldsymbol{W})(\boldsymbol{Q}\boldsymbol{P}^{-1})^m \; . \tag{2.9}$$

Multiplying (2.9) on the right by $\boldsymbol{A}$ and applying (2.7) leads to a formula for the error of $\boldsymbol{M}^{-1}$ as an approximate inverse of $\boldsymbol{A}$,

$$\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A} = (\boldsymbol{P}^{-1}\boldsymbol{Q})^m \cdot (\boldsymbol{I} - \boldsymbol{W}\boldsymbol{A}) \cdot (\boldsymbol{P}^{-1}\boldsymbol{Q})^m \; . \tag{2.10}$$

We note that $||(\boldsymbol{P}^{-1}\boldsymbol{Q})^{2m}||_{\boldsymbol{A}} = ||(\boldsymbol{P}^{-1}\boldsymbol{Q})||^{2m}_{\boldsymbol{A}}$ since $\boldsymbol{P}^{-1}\boldsymbol{Q}$ is symmetric with respect to the inner product $(u,v)_{\boldsymbol{A}} = (u, \boldsymbol{A}v)$. Consequently, from (2.10) with $\boldsymbol{W} = 0$ we obtain $\rho(\widetilde{\boldsymbol{A}}^{-1}_{2m-1}) = ||(\boldsymbol{P}^{-1}\boldsymbol{Q})||^{2m}_{\boldsymbol{A}}$. Then from (2.10) we obtain the bound

$$\rho(\boldsymbol{M}^{-1}) \le \rho(\widetilde{\boldsymbol{A}}^{-1}_{2m-1}) \cdot \rho(\boldsymbol{W}) \; . \tag{2.11}$$

Hence, $\boldsymbol{M}^{-1}$ is a better approximate inverse than either of its components provided that each component is a reasonable approximate inverse satisfying $\rho(\widetilde{\boldsymbol{A}}^{-1}_{2m-1}) < 1$ and $\rho(\boldsymbol{W}) < 1$.

For large systems $||\boldsymbol{P}^{-1}\boldsymbol{Q}||$ is near 1 and for $\boldsymbol{M}^{-1}$ to be an effective preconditioner $\boldsymbol{W}$ and $\boldsymbol{P}^{-1}\boldsymbol{Q}$ need to be complementary in the sense that they approximate opposite ends of the spectrum of $\boldsymbol{A}$. To see this we note that the condition number is the ratio of the largest and smallest eigenvalues of $\boldsymbol{M}^{-1}\boldsymbol{A}$. If $\boldsymbol{P}^{-1}\boldsymbol{Q}$ acts to reduce the large eigenvalues (short wavelengths) and $\boldsymbol{W}$ acts to increase the small eigenvalues (long wavelengths) then the range of spectrum of $\boldsymbol{M}^{-1}\boldsymbol{A}$ is squeezed from both ends and the condition number with the combined approximate inverse can be much smaller than the condition number with either of its components. This heuristic can be made rigorous by applying the error analysis used in multigrid theory, see for example the review article by Xu [10].

Equation (2.8) forms the basis for a number of approximation schemes. Tatebe [9] showed that $\boldsymbol{M}^{-1}$ is symmetric positive definite if $\boldsymbol{W}$, $\boldsymbol{P}$ and $\boldsymbol{Q}$ are symmetric, $\boldsymbol{P}+\boldsymbol{Q}$ positive definite and $\boldsymbol{W}$ positive. Since $\boldsymbol{W}$ need be only positive and not positive definite, the coarse grained inverse can be used for $\boldsymbol{W}$ even though its null space is non-empty. Also note that $||\boldsymbol{P}^{-1}\boldsymbol{Q}|| < 1$ implies that $\boldsymbol{P} + \boldsymbol{Q}$ is positive definite.

2.1. Polynomial Preconditioner. The simplest choice is $\boldsymbol{W} = \boldsymbol{P}^{-1}$. In this case, $\boldsymbol{M}^{-1}$ corresponds to a conventional polynomial preconditioner; *i.e.,* $\boldsymbol{M}^{-1} = \widetilde{\boldsymbol{A}}^{-1}_{2m}$. Similarly, using $\boldsymbol{W} = \widetilde{\boldsymbol{A}}^{-1}_{2m-1}$ merely increases the number of terms in the series; *i.e.,* $\boldsymbol{M}^{-1} = \widetilde{\boldsymbol{A}}^{-1}_{4m-1}$. The disadvantage of this choice for $\boldsymbol{W}$ is that at high resolution, $||\boldsymbol{P}^{-1}\boldsymbol{Q}||$ is typically close to 1 and therefore $\boldsymbol{M}^{-1}$ is a poor preconditioner. The underlying reason is that with the standard splittings $\boldsymbol{P}$ connects only neighboring grid points and consequently $\boldsymbol{M}^{-1}$ provides a poor approximation of $\boldsymbol{A}^{-1}$ at long wavelengths. This is born out by experience showing that the number of iterations grows rapidly with the dimension of the system.

2.2. Multigrid Preconditioner. A better choice is that advocated by Tatebe [9] which aims to account for the long wavelengths by basing the preconditioner on a single step of a multigrid algorithm. Let G_k be a sequence of successively coarser grids, $\boldsymbol{R}_{k+1,k}$ and $\boldsymbol{E}_{k,k+1}$ be the restriction and prolongation operators connecting the grids G_k and G_{k+1}, $\boldsymbol{A}_k$ the coarsened operator on the grid G_k, and $\boldsymbol{P}_k$ and $\boldsymbol{Q}_k$ the splitting of $\boldsymbol{A}_k$. Here $k = 0$ corresponds to the finest mesh and $k_{\max}$ to the coarsest mesh. The preconditioner can be defined recursively as

$$(2.12)\ \boldsymbol{M}_k^{-1} = (\boldsymbol{P}_k^{-1}\boldsymbol{Q}_k)^m(\boldsymbol{E}_{k,k+1}\boldsymbol{M}_{k+1}^{-1}\boldsymbol{R}_{k+1,k})(\boldsymbol{Q}_k\boldsymbol{P}_k^{-1})^m + \boldsymbol{P}_k^{-1}\sum_{j=0}^{2m-1}(\boldsymbol{Q}_k\boldsymbol{P}_k^{-1})^j\ .$$

On the coarsest mesh, the problem can be solved exactly. Thus, the recursion ends with $\boldsymbol{M}_{k_{\max}}^{-1} = \boldsymbol{A}_{k_{\max}}^{-1}$. The standard multigrid algorithm uses $\boldsymbol{M}_0^{-1}$ as an approximate inverse in conjunction with a plain iterative solver.

If the restriction and prolongation operators are chosen to be adjoints of each other, $\boldsymbol{R}_{k+1,k}^T = \boldsymbol{E}_{k,k+1}$, the coarse grained approximate inverse at every level, $\boldsymbol{W}_k = \boldsymbol{E}_{k,k+1}\boldsymbol{M}_{k+1}^{-1}\boldsymbol{R}_{k+1,k}$ is symmetric. For many systems, Tatebe [9] has shown that with a Jacobi smoother or a Gauss-Seidel smoother the multigrid approximate inverse $\boldsymbol{M}_0^{-1}$ is both symmetric and positive definite. Thus, it can be used as a preconditioner for the conjugate gradient algorithm.

We note that (2.12) is the same approximate inverse as defined by Algorithm 3.8 in Xu's review[1] [10] when the multigrid smoother is identified with the approximate inverse $\widetilde{\boldsymbol{A}}_{m-1}^{-1}$. In addition, Xu pointed out [10, Proposition 2.2 and 2.3] that using $\boldsymbol{M}^{-1}$ as a preconditioner for a conjugate gradient algorithm results in a substantially faster convergence rate than what could be achieved with an optimum over-relaxed iterative scheme. The advantage of the conjugate gradient scheme over the plain iterative scheme, $u_{n+1} = u_n + \boldsymbol{M}^{-1}(f - \boldsymbol{A}u_n)$, standardly used in multigrid algorithms can be even more dramatic. The plain iteration scheme diverges if $\boldsymbol{M}^{-1}\boldsymbol{A}$ has an eigenvalue greater than 2, even when the condition number of $\boldsymbol{M}^{-1}\boldsymbol{A}$ is quite modest. In contrast the convergence of the conjugate gradient algorithm depends only on the condition number and hence makes for a more robust algorithm. A numerical example in subsection 4.3 illustrates this point.

In the multigrid terminology, (2.12) is a V-cycle. A W-cycle corresponds to replacing $\boldsymbol{M}_{k+1}^{-1}$ on the right-hand side of (2.12) with

$$\boldsymbol{M}_{k+1}^{-1} + (\boldsymbol{I} - \ M_{k+1}^{-1}\boldsymbol{A}_{k+1})\boldsymbol{M}_{k+1}^{-1}\ .$$

In effect, the first two terms in the expansion

$$\boldsymbol{A}^{-1} = \left[\boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\right]^{-1}\boldsymbol{M}^{-1}$$

are used to improve the approximation for the coarse inverse $\boldsymbol{A}_{k+1}^{-1}$. The improvement results from partially coupling the short and long wavelengths at every level. Tatebe discussed both V-cycle and W-cycle multigrid preconditioners in conjunction with the conjugate gradient algorithm. Our method, discussed below, makes the coupling between short and long wavelengths even stronger.

[1]There is a misprint in [10]. Step 3 of Algorithm 3.8 should be the same as Step 2 of Algorithm 3.6 with v^2 substituted for v^1.

For $\boldsymbol{M}_0^{-1}$ to be a good approximate inverse of $\boldsymbol{A}$, the splitting must be chosen such that $\boldsymbol{P}_k^{-1}\boldsymbol{Q}_k$ smoothes the shorter wavelengths to complement $\boldsymbol{W}_k$ which operates only on the longer wavelengths. The Jacobi splitting for the Laplace operator provides an example which illustrates this point. The standard Jacobi splitting does not damp the shortwave lengths associated with the checker board mode and does not work well in conjunction with multigrid algorithms. However, the Jacobi splitting with weight factor of $\frac{1}{2}$ does damp short wavelengths, including the checker board mode, and does work with the multigrid algorithm.

The underlying rationale for the multigrid algorithms is a wavelength decomposition. To put the multigrid preconditioner in perspective, we note that the plain multigrid algorithm is known to be super-convergent (scales proportional to N) for the Laplace operator. The Laplace operator has the property that it is diagonal in Fourier space and hence all the wavelengths are decoupled. In general, different wavelengths are coupled and the multigrid algorithm can be expected to scale only as $N \log N$. Stronger coupling between wavelengths increases the difficulty of the problem. For harder problems, the multigrid algorithm does not provide sufficient coupling between wavelengths and convergence may be slow or even fail if $\boldsymbol{M}^{-1}\boldsymbol{A}$ has an eigenvalue greater than 2. Using a single multigrid step as a preconditioner for a conjugate gradient algorithm extends the range of problems that can be solved [9, 1].

When the system is large enough to require many coarsening levels, the number of iterations needed to fully couple the long and short wavelengths can increase. This effect has been observed in [1, tables III, IV and V].

2.3. Recursive Multiscale Conjugate Gradient. The difficulty with the multigrid preconditioner can be overcome by defining the preconditioner in terms of the exact inverse of the coarsened operator $\boldsymbol{A}_c$ as follows

$$\boldsymbol{M}^{-1} = (\boldsymbol{P}^{-1}\boldsymbol{Q})^m(\boldsymbol{E}\boldsymbol{A}_c^{-1}\boldsymbol{R})(\boldsymbol{Q}\boldsymbol{P}^{-1})^m + \boldsymbol{P}^{-1}\sum_{j=0}^{2m-1}(\boldsymbol{Q}\boldsymbol{P}^{-1})^j \; , \tag{2.13}$$

In effect, $\boldsymbol{W} = \boldsymbol{E}\boldsymbol{A}_c^{-1}\boldsymbol{R}$ and the evaluation of $\boldsymbol{A}_c^{-1}$ on a vector is computed by applying the same algorithm to the coarse grained operator. Again the recursion ends by solving the problem exactly on the coarsest level. Using the exact inverse $\boldsymbol{A}_c^{-1}$ for $\boldsymbol{W}$ is the best preconditioner based on a single level of scaling. Instead of using $\boldsymbol{A}_c^{-1}$, Tatebe's scheme uses an approximation to the coarse inverse given by $\boldsymbol{M}_1^{-1}$ from (2.12). This approximation to the coarse inverse gets worse as the number of levels increases.

An important feature of our preconditioner is that the long and short wavelengths are coupled at each level by a conjugate gradient solver before proceeding to the next finer grid. This prevents truncation errors from the coarsening of the operator at each level from accumulating. In particular, if the coarsened operator does poorly on a few modes, which typically occurs when the coefficients of the underlying PDE are discontinuous, then these modes will be corrected by the conjugate gradient solver with only a small penalty.

The efficiency of the recursive multiscale algorithm is due to the high quality of the approximate inverse. By accounting for extremes in the spectrum, both short and long wavelengths, the preconditioned matrix $\boldsymbol{M}^{-1}\boldsymbol{A}$ has a low condition number and consequently only a small number of iterations is needed for the conjugate gradient solvers at every level. Our $\boldsymbol{M}^{-1}$ is a special case of a multigrid scheme. It is particularly simple in that it involves only 2 levels with an exact coarse inverse.

Consequently, multigrid theory provides an upper bound on the condition number of $\boldsymbol{M}^{-1}\boldsymbol{A}$. We expect the high quality of the preconditioner to result in a low number of iterations. Of course the reduction in the iterations must overcome any increase in the cost per iteration. The numerical examples below show that this is indeed the case.

The conjugate gradient algorithm relies on a simple recursion relation for the conjugate directions. The orthogonality of the conjugate directions is a consequence of the linearity of the preconditioner. A potential drawback of a preconditioner that depends on a linear solver is that the preconditioner is only a linear operator if the coarse grained inverse is solved accurately. In the numerical experiments described below we have found that the coarse grained inverse only need be solved to an accuracy comparable to that desired for the overall solution on the fine grid. This may be explained by following heuristic argument. Suppose the error from the inaccuracy in the coarse grained inverse is random. Then for a large problem, dimension $O(10^6)$, the component of the error along a particular vector is likely to be small. Thus the error in orthogonality for the first few iterations, $O(10)$, is small. If only a small number of iterations are required because of the quality of the approximate inverse then the effect of small non-linearities of the preconditioner, introduced by the recursive solvers, is negligible.

Other multiscale schemes that also use a conjugate gradient algorithm at every level have been developed, in particular, the cascadic conjugate gradient and the cascadic multigrid schemes [7, 4]. In contrast to our algorithm these are "one-way multigrid" methods in which the coarse grids are used to determine initial guesses for iterative algorithms on the next finer grid until a solution is obtained at the desired grid resolution. Though the approximate inverse in these methods is not strictly linear, convergence of the scheme has recently been proven for a large class of elliptic problems. The cascadic multiscale schemes are efficient when the short wavelengths have a minimal effect on the long wavelengths. Since our approach is to fully couple the short and long wavelengths before proceeding to the next level, we expect it to be more effective on difficult problems in which wavelengths are strongly coupled.

3. Implementation. To validate the algorithm we have implemented the solver in an object oriented C++ based code. The implementation is general enough to allow at every level for an arbitrary choice of solver and an arbitrary preconditioner. We have tested the code with the conjugate gradient solver using the three preconditioners described in the previous section: polynomial preconditioner, multigrid preconditioner and the recursive multiscale preconditioner. In addition, the implementation includes the standard multigrid algorithm. This enabled us to compare the algorithms while using the same prolongation, restriction, splitting and coarsening operators.

As a test case we used the porous flow equation

$$\nabla \cdot (\boldsymbol{K} \cdot \nabla P) = S \ . \tag{3.1}$$

Here, $\boldsymbol{K}$ is a permeability tensor, P is the pressure and S is a source. We considered only a diagonal tensor and used the linear system derived from the standard 5-point stencil for the finite difference operator on a two-dimensional regular grid. In the discretization P and S are cell centered fields whereas the components of K are face centered; from the cell centers K_{xx} is offset by a 1/2 cell in the x-direction and K_{yy} is offset by a 1/2 cell in the y-direction. The face centered components of the permeability field are obtained as the harmonic mean of the adjacent cell centered values. This discretization is a special case of support operator differencing [8] and results in a matrix that preserves the positivity and symmetry of the differential operator.

The discrete operator can be decomposed as $\boldsymbol{A} = \boldsymbol{D}+\boldsymbol{U}+\boldsymbol{L}$ where $\boldsymbol{D}$ is diagonal, $\boldsymbol{U}$ is strictly upper triangular and $\boldsymbol{L}$ is strictly lower triangular. In terms of these matrices, we have used two operator splitting $\boldsymbol{A} = \boldsymbol{P} - \boldsymbol{Q}$: the symmetric Gauss-Seidel splitting for which $\boldsymbol{P} = (\boldsymbol{D} + \boldsymbol{L})\boldsymbol{D}^{-1}(\boldsymbol{D} + \boldsymbol{U})$ and $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U}$, and the Jacobi splitting with weight factor of $\frac{1}{2}$, $\boldsymbol{P} = 2\boldsymbol{D}$ and $\boldsymbol{Q} = \boldsymbol{D} - \boldsymbol{U} - \boldsymbol{L}$. As is typical of multigrid algorithms, the Gauss-Seidel splitting works better and is used for the results presented below.

We allow for both Dirichlet and Neumann boundary conditions. Each point on the boundary can be independently chosen to satisfy one or the other condition. The type of boundary condition affects the discretized $\boldsymbol{K}$ field along the boundary. In addition, the value of the boundary condition enters as a source term in the boundary cells of the discretized equations. The boundary source terms are large and under coarsening scale differently than the source terms in the interior. To avoid complications from the boundary source terms, we transform to a problem with zero boundary conditions. This is accomplished by generating a smooth field P_{bf} that matches the boundary conditions. We then solve the problem for $\delta P = P - P_{\mathrm{bf}}$ with zero boundary conditions but an additional source term $-\nabla\cdot(\boldsymbol{K}\cdot\nabla P_{\mathrm{bf}})$.

The coarse grained permeability tensor is based on the transmissivities, $T_{xx} = \frac{\Delta y}{\Delta x}K_{xx}$ and $T_{yy} = \frac{\Delta x}{\Delta y}K_{yy}$ where Δx and Δy are the width and height of a grid cell. Physically these are extensive rather than intensive quantities. The transmissivity behaves like a conductance, in contrast to the permeability which behaves as a conductivity. The T_{xx} component is taken as

$$\frac{1}{T^c_{xx}} = \sum_i \frac{1}{\sum_j T_{xx}(i,j)} \tag{3.2}$$

where the sum is over fine cells contained within a coarse cell, and i and j are the indices corresponding to the x- and y-directions respectively. When a fine cell only partially overlaps a coarse cell the value of T_{xx} associated with the fine cell is multiplied by the fraction of the height in the overlap and divided by the fraction of the width in the overlap. This is the scaling for the conductance of a sub-cell when the conductivity is constant within the cell. A similar construction is used for T_{yy} with the sums over i and j interchanged. The coarsened permeability is an approximation analogous to treating resistances transverse to the flow direction in parallel and then these collective resistances in series. (Recall that resistances in parallel are computed like conductances in series and vice versa.)

The coarse grained operator remains diagonal. Despite its simplicity, it satisfies three properties that enable it to be quite useful. (i) Even if $\boldsymbol{K}$ on the finest grid is a scalar, the coarsened $\boldsymbol{K}$ is typically not scalar. (ii) The effect of the boundary conditions on the boundary K's are properly accounted for. (iii) The Laplace operator is preserved under coarsening.

It is noteworthy that the scale factor between grids is not limited to a factor of two, nor for that matter to an integer. Consequently, the number of grid points in a linear dimension does not have to be a power of the scale factor. Typically, we scaled between successive grid levels by a linear factor of 4 which reduces the dimension of the problem at each level by a factor of 16. The effective scale factor between adjacent levels tends to vary slightly in order to maintain an integer grid dimension.

The restriction operator is taken as the adjoint of the prolongation operator. For the prolongation operator we use the tensor product of 1-D linear interpolations. A piecewise constant interpolation works nearly as well. In contrast to the piecewise constant interpolation, a linear interpolation requires a boundary condition. For both Dirichlet and Neumann boundaries, we chose a zero slope for the boundary interpolation. While this correctly captures the Neumann condition, it introduces a small error for the Dirichlet case. However, the viability of the piecewise constant interpolation suggests that the error is small and can be neglected. Numerical experiments confirm this suggestion. For a Neumann boundary, since δP near the boundary can be large, a zero boundary condition is highly detrimental to the convergence of the algorithm. The fact that the zero slope interpolation condition is acceptable for both the Dirichlet and Neumann case greatly simplifies the implementation of boundary conditions that can vary between Dirichlet and Neumann from cell to cell.

The degree m of smoothing is related to the scale factor. Since the smoothing operator $\boldsymbol{P}^{-1}\boldsymbol{Q}$ typically connects only neighboring grid points we have chosen m to be the same as the scale factor. As a result the preconditioning matrix fully couples every fine grid cell. Smaller values of m would require a larger number of conjugate gradient iterations for problems in which the short wavelengths dominate the solution.

Our implementation of the conjugate gradient algorithm is conventional, as outlined in [3, §2.3.1, Figure 2.5]. We base the convergence criterion on the norm of the residual rather than the norm with respect to the preconditioner. This is because applying the preconditioner is the most expensive operation in the conjugate gradient step. It is performed at the beginning of the cycle and hence is out of date when the check for convergence is made at the end of the cycle. A convergence criterion of the same type is applied on every level. The criterion for convergence on the k^{th} level is

$$\frac{||\vec{r}||^2}{N_k} < f^k \times \epsilon^2 \tag{3.3}$$

where $\vec{r}$ is the residual, N_k is the number of grid points, ϵ is the desired root mean squared error of the residual on the finest level, and f is an adjustable parameter that allows us to tighten the error criterion on the coarser levels. The algorithm appears not to be very sensitive to the choice of f. In practice we found $f = 0.1$ works well. As f increases the number of iteration on the fine level gradually increases, while too small a value of f results in unnecessary iterations on the coarse levels.

4. Numerical Examples. We have tested our solver algorithm on several examples of the porous flow equations. The examples below use both Dirichlet and Neumann boundary conditions, typically, constant pressure $P = 1$ on the left and $P = 0$ on the right, and no flow on the top and bottom. We generated random log-normal permeability fields with either a Gaussian auto-correlation function or a power law auto-correlation function

$$\left(\frac{1}{1 + (\vec{r}_1 - \vec{r}_2)\boldsymbol{\Lambda}(\vec{r}_1 - \vec{r}_2)}\right)^{\frac{1}{4}}$$

where $\boldsymbol{\Lambda}$ is a positive definite matrix which defines cutoff lengths for the power law behavior. The variance of the permeability field is adjusted by scaling the log of the field. Similar problems have previously been used to test the conjugate gradient algorithm with a multigrid preconditioner [1].

Our test examples include meshes varying in size by a factor of 1000; from 2×10^3 to 2×10^6 grid points. We also have varied the difficulty of the problem by increasing

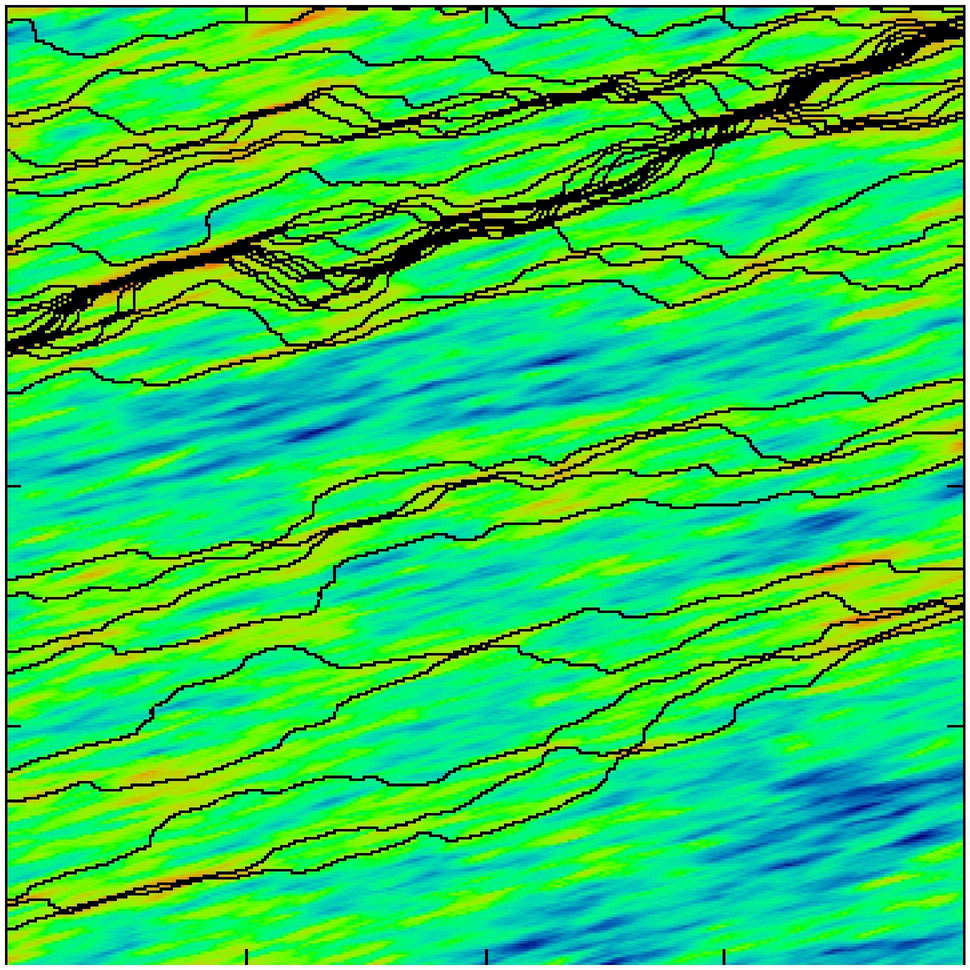

Fig. 4.1. *Flow lines (in black) superimposed on the log of permeability field for the base case. The permeability field is random log-normal. The log of the field has a zero mean, a variance of 2 and a power law auto-correlation. The field is discretized on a* 1000×1000 *grid and its log ranges from a low of -7.3 (blue) to a high of 7.9 (red). We note for this "fractal" permeability field the flow lines tend to follow narrow channels. This is in contrast to our experience with Gaussian auto-correlations where flow lines avoid obstacles but do not bunch into narrow channels.*

the variance of the permeability field to obtain a maximum to minimum permeability ratio up to 8×10^9. In addition, we have adjusted the error tolerance to vary the accuracy of the solution up to machine accuracy.

4.1. Base case. For a basic test case we used a power law field on a quarter of the unit square. We choose a variance of 2 with a zero mean and minimum correlation lengths of 0.016 and 0.002 oriented at 15 degrees with respect to the x-axis. The resulting permeability field contains a wide range of wavelengths. On a 1000×1000 grid the distribution of the log of the permeability field extends over 3.5 standard deviations. Consequently, the permeability field varies from 10^{-3} to 10^3 and the ratio of its maximum to minimum value is 10^6. The flow lines for the solution superimposed on the log of the permeability field are shown in figure 4.1. The conjugate gradient

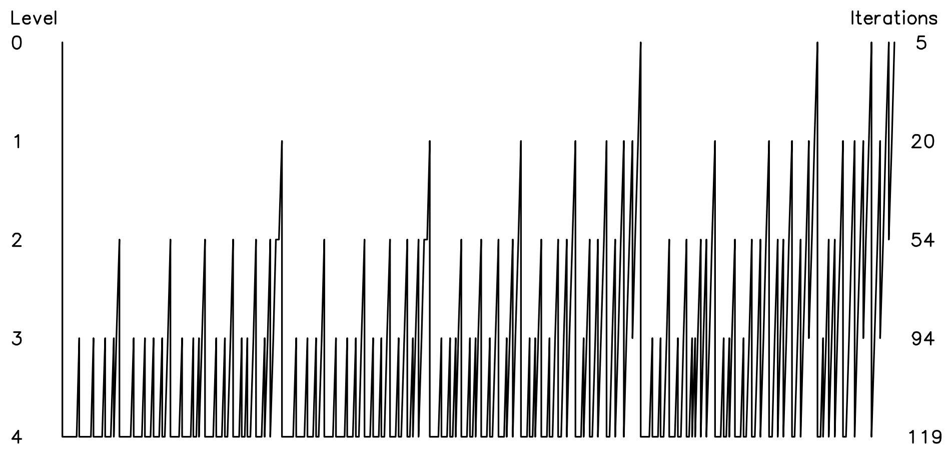

Fig. 4.2. *Level sequence for the base case. The x-coordinate is a time like variable. As the calculation precedes, the line indicates the grid level at which a conjugate gradient step is evaluated. Thus, the line represents the dynamical transitions between levels which replaces the preprogrammed V-cycle or W-cycle standardly used in multigrid algorithms. The level is given on the left. Level 0 is the finest grid and level 4 is the coarsest grid. The cumulative number of conjugate gradient iterations at each level is shown at the right.*

Table 4.1

Iteration count on each level for the base case. Computational effort on a level is proportional to the dimension of the level times the total number of iterations on that level.

Level	Grid	dimension	iterations	ratio	iterations× dimension	per cent total
4	$4 \times 4 =$	16	119		1904	0.03
				> 1.3		
3	$16 \times 16 =$	256	94		24064	0.37
				> 1.5		
2	$63 \times 63 =$	3869	63		214326	3.29
				> 3.2		
1	$252 \times 252 =$	63504	20		1270080	19.48
				> 4		
0	$1001 \times 1001 =$	1002001	5		5010005	76.84
					total 6520379	

solver on the finest level required 5 iterations to reduce the mean squared residual by a factor of 10^{10}. Later we show that this corresponds to 5 digits accuracy. Replacing the Neumann boundary conditions on the top and bottom with Dirichlet boundary conditions does not change the performance of the solver.

The algorithm dynamically determines its effort on every level. The sequence of transitions between levels (the replacement for the preprogrammed V-cycle or W-cycle standardly used in multigrid algorithms) is displayed in figure 4.2. It is seen that fewer and fewer conjugate gradient iterations for the coarse inverse are used for subsequent evaluations of the preconditioner for a conjugate gradient step on the fine level. This is an indication that the solution converges for long wavelengths before it converges for short wavelengths. The diminishing effort on the coarse levels is an important factor that allows the computational effort to scale with size proportional to N rather than $N \log N$. The total number of iterations is summarized in table 4.1. Even though the cumulative number of iterations is larger for the coarser grids, the ratio of iterations between levels goes down. The effort on each level is proportional to the number of iterations on that level times the dimension of the level. It is seen from the table that the total effort is dominated by the computation on the finest level.

To set the scale, the total computational effort is equivalent to about 1600 scalar products of vectors on the finest grid. On a workstation (SUN Ultra I, 170 Mhz) or a PC (PentiumPro, 200 Mhz) the time per point is 240 μs. This time is for the solver only and does not include initialization of the permeability fields on the coarse grids. Our implementation is memory efficient. The solver requires storage for a total of 10 vectors: 3 for the matrix, one each for the pressure field, the source field, the residual and the conjugate direction, and three temporaries to evaluate the preconditioner (2.13) on the residual.

Specializing the algorithm to the Laplace equation reduces the computational effort to 50 μs per point and decreases the required memory by 4 vectors. The difference in speed, as well as memory, can be attributed largely to the much more efficient coding for the simpler linear operator. It is a remarkable fact that the same preconditioned conjugate gradient algorithm works equally well for an operator with a rapidly varying permeability field as for the simple Laplace operator.

We choose this problem, which is fairly large for a workstation, as a base case because we felt it is necessary to have several levels of coarsening to assess the quality of the algorithm. With a scale factor of roughly 4, there are only 4 coarsenings for a 1000×1000 grid. The problem we have choosen is not trivial. As shown below the polynomial preconditioner takes a large number of iterations and the standard multigrid algorithm fails to converge. Our implementation of Tatebe's algorithm succeeds and will be discussed in more detail later.

Though the algorithm has not been optimized, it is robust and the run time is within a factor of 2 over a reasonable range for the parameters characterizing the algorithm. Figure 4.3 shows the effect of the scale factor between grids. For scale factors between 2 and 5, the time per point varies by only 30%. A minimum time of 180 μs per point occurs with a scale factor of 3.

In a recent modification to our code (using the multigrid procedure [10, Algorithm 3.8] for evaluating $\boldsymbol{M}^{-1}$ applied to a vector) without changing the preconditioner we achieved a reduction in run time by 40% at the cost of increasing the memory requirement by 10%, one additional vector. Thus, we expect that on a current top of the line PC (450 Mhz Pentium II) the runtime for our base case problem would be substantially less than a minute.

4.2. Scaling behavior. For high resolution, the scaling properties of the solver algorithm are critical. To test the scaling behavior we solved a series of problems on sucessively larger grids varying in dimension by 3 orders of magnitude. We found the effort per point to be independent of the size of the grid.

We started by generating a random permeability field on a unit square with a very fine grid, 2048×2048. Then the grid is truncated to a square subgrid by using only the subregion defined by the lower left corner and a specified upper right corner. The 1000×1000 field corresponds to our base case and is choosen to have a variance of 2 with a mean of zero. The cutoff lengths of the correlation function on this sequence of grids remains constant (32 by 4 cells) but the variance of the permeability field increases with the size of the grid. Consequently, the problem gets harder as the size increases.

Figure 4.4 shows the scaling behavior on grids ranging from 50×50 to 1600×1600. It can be seen that the number of iterations and the time per point are almost constant.

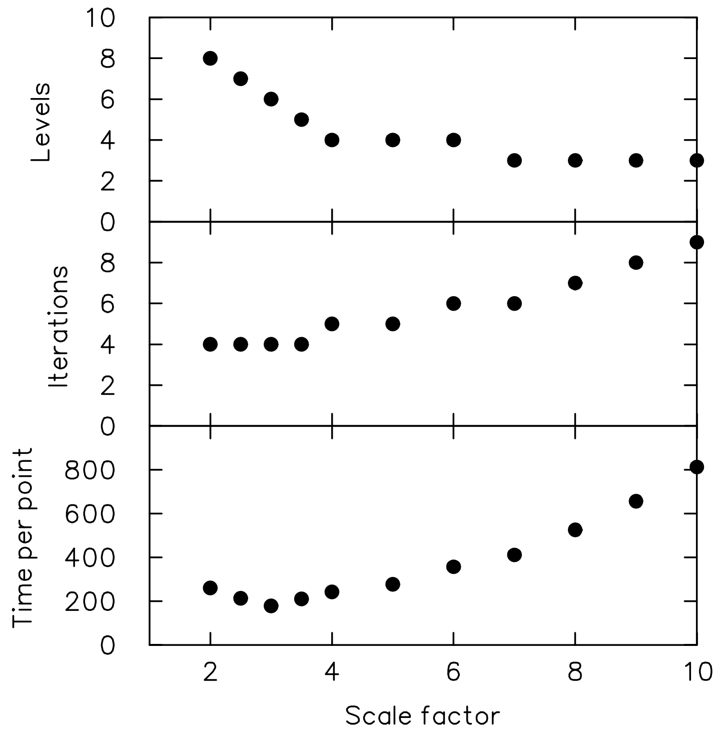

FIG. 4.3. *The effect of the scale factor between grids on the base case. Shown are the number of levels, iterations on the finest grid, and the time per point.*

This indicates that the computational effort scales linearly with the grid dimension. Despite the variable coefficients, this scaling is better than the $N \log N$ scaling for solving the Laplace equation with fast Fourier transforms.

We have run the same problems with Tatebe's multigrid preconditioning algorithm. As seen in figure 4.5 this algorithm requires more iterations. The overall trend is linear which indicates that the computational effort scales as $N \log N$. This is in line with the results of Ashby & Falgout [1]. A comparison with their results is necessarily imprecise because they concentrated on 3-D problems and a parallel implementation. The larger number of iteration in Tatebe's algorithm is partially offset by a lower cost per iteration. For the largest problems in this series, the time per point is a factor of two greater than for our algorithm. For the small problems the two algorithms are comparable in time.

In addition, we have tested the scaling behavior on a family of problems with the permeability field generated by interpolating from the very fine grid to coarser grids on the same physical domain and then rescaling to obtain the same variance. For this family of problems, the cutoff length to the auto correlation function in units of cells is proportional to the grid size. The decreasing smoothness of the discretized field, in terms of cell to cell variation, increases the difficulty of the problem for the smaller grids. Due to the changing correlation length, with this set of problems the time per point for our algorithm decreases slightly as the grid dimension increases while Tatebe's algorithm continues to scale as $N \log N$. For the largest grid our algorithm is again 50% faster than Tatebe's algorithm.

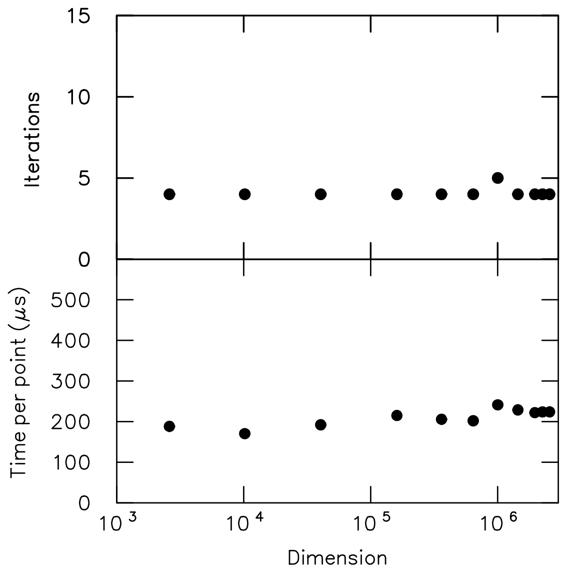

FIG. 4.4. *Number of iterations on the finest grid and time per point versus number of grid points for the recursive multiscale conjugate gradient algorithm.*

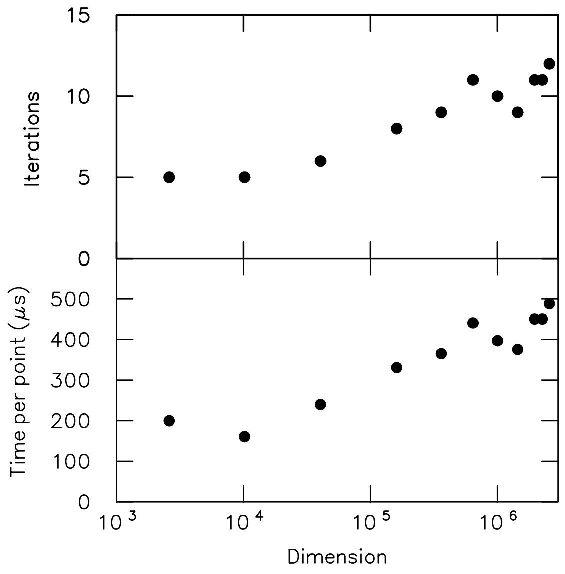

FIG. 4.5. *Number of iterations on the finest grid and time per point versus number of grid points for the multigrid preconditioned conjugate gradient (Tatebe's) algorithm.*

4.3. Accuracy. In order to obtain an estimate of the accuracy of our method, we first generated a test problem for which the exact solution is known. To this end we solved our base case approximately using another solver. We intentionally did not strive for high accuracy. By adding the small residual of the approximate solution to the source term of the base problem, we created a new test problem which by construction is solved exactly by the approximate pressure field of the base problem.

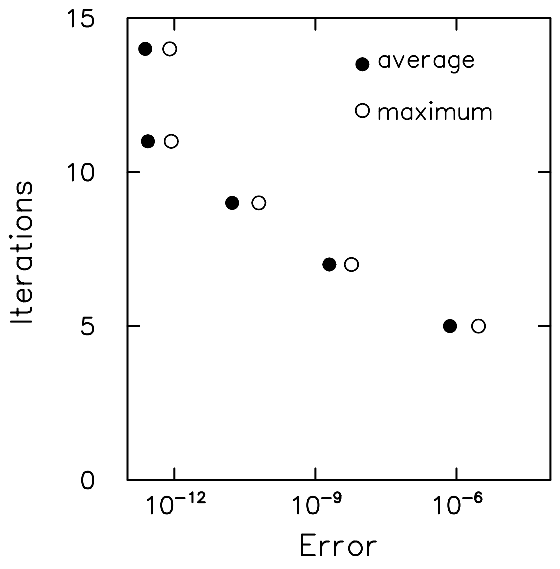

FIG. 4.6. *Iterations vs accuracy for the base case. The solution field is of order 1.*

By varying the error tolerance of our solver we generated a sequence of solutions. For these solutions, the iteration count as a function of the accuracy is shown in figure 4.6. Both the root mean squared error and the maximum error are used as measures of the accuracy. We observe that the iteration count increases linearly with the log of the accuracy until the improvement of the solution is limited by machine accuracy. The linear trend indicates that each iteration reduces the error by a factor of about 11. Consequently, only a small number of iterations are needed to achieve machine accuracy. The fact that the maximum error is within a factor of 3 of the root mean squared error indicates that the solution is uniformly accurate. This is another strong point of our algorithm and is a consequence of the preconditioner acting on all length scales.

To further test the robustness of our algorithm we increased the difficulty of the base problem by increasing the variance of the log of the permeability field. The iteration count as a function of the variance is shown in figure 4.7. A variance of 3 results in the values of the permeability field varying from minimum to maximum by a factor of 8×10^9. For larger variances, round-off errors in the finite difference approximation to $\nabla \cdot (\boldsymbol{K} \cdot \nabla P)$ limits the accuracy to which the solution can be computed. We observe that for our multiscale algorithm the iteration count is almost constant (4 or 5), whereas the iteration count for Tatebe's algorithm grows from 5 to 27. The plain multigrid algorithm works well when the variance is below 1.5 but the residual diverges when the variance is above 2. Ashby & Falgout [1, table V] observed the same trend.

For the iteration in the multigrid algorithm, the convergence rate is given by $||\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}||_{\boldsymbol{A}}$. Since $\boldsymbol{M}^{-1}\boldsymbol{A}$ is a positive operator, the trend with variance implies that the maximum eigenvalue of $\boldsymbol{M}^{-1}\boldsymbol{A}$ increases above 2 as the variance is increased. In contrast the convergence rate of the conjugate gradient algorithm depends only on the condition number. The trend of Tatebe's algorithm implies that the condition number of $\boldsymbol{M}^{-1}\boldsymbol{A}$ increases slowly with variance. As a test of this hypothesis on the

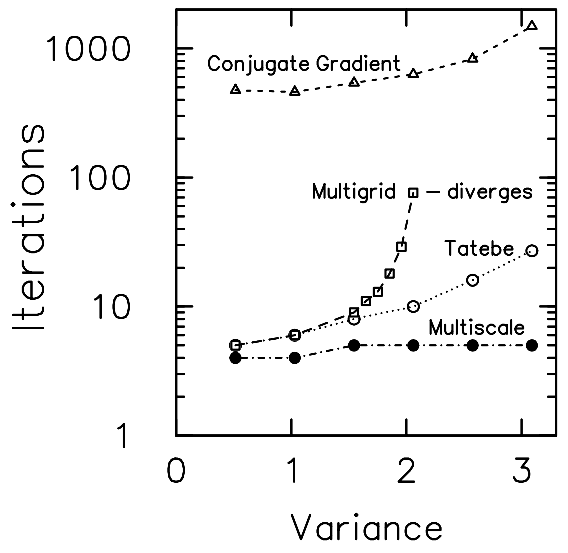

FIG. 4.7. *Iterations vs variance of the log of the permeability field. The variance is adjusted by scaling the log of the permeability field used in the base case. Four cases are shown; conjugate gradient with polynomial ($m = 8$) preconditioner (triangle), plain multigrid (square), multigrid preconditioned conjugate gradient or Tatebe's algorithm (open circle), and recursive multiscale conjugate gradient algorithm (solid sircle). The plain multigrid algorithm diverges for variance above 2.*

eigenvalues of $\boldsymbol{M}^{-1}\boldsymbol{A}$, we applied a relaxation to the plain iteration in the multigrid algorithm, *i.e.*, $u_{n+1} = u_n + \omega \boldsymbol{M}^{-1}(f - \boldsymbol{A}u_n)$ with $\omega = 0.5$. With this modification the multigrid algorithm does converge for the cases with larger variance, although as expected, at a slow rate. This example illustrates that a conjugate gradient iteration in place of a plain iteration increases the robustness of the multigrid scheme.

We also ran this set of problems using the conjugate gradient algorithm with a polynomial ($m = 8$) preconditioner. This simple algorithm does converge, but the iteration count is large; varying from 474 iterations for a variance of 0.5 to 1483 iterations for a variance of 3. Since the linear dimension is 1000 points, the large number of iterations does couple all wavelengths even though the preconditioner is only effective on short wavelengths. Though the multigrid preconditioner does not account for the coupling between short and long wavelengths at every level, the conjugate gradient iteration on the outer level (finest grid) provides sufficient coupling to greatly decrease the needed number of iterations. The multiscale preconditioner couples the short and long wavelengths at every level and consequently the number of iterations needed at the outer level is nearly the same. The cost for the preconditioner does increase with the difficulty of the problem because more inner iterations are used as the variance is increased. But at the highest variance, the algorithm with the multiscale preconditioner takes 1/3 the time as the algorithm with the multigrid preconditioner.

These tests shows that our algoithm is very robust. The preconditioner is effective on difficult problems. In fact, its advantage increases with the difficulty of the problem. With a small increase in the number of iterations the solution can be driven to near machine accuracy. It is remarkable, that despite the rather large spatial variation in the value of the permeamility field, a solution can still be obtained in only 5 iterations.

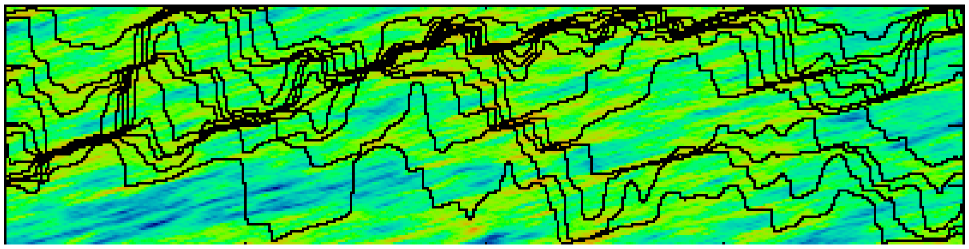

FIG. 4.8. *Flow lines (in black) superimposed on the log of permeability field for the flow in a narrow channel. The permeability field is random log-normal. The log of the field has a zero mean, a variance of 2 and a power law auto-correlation. The field is discretized on a* 2000×500 *grid and its log ranges from a low of -7.2 (blue) to a high of 7.9 (red). The cells have a 10 to 1 aspect ratio resulting in a domain with a 40 to 1 aspect ratio. To fit on the page, the x-direction has been compressed by a factor of 10. This is equivalent to a non-isotropic permeability field with K_{yy} a factor of 100 times K_{xx}.*

TABLE 4.2
Iteration count on each level for the flow in a narrow channel. Computational effort on a level is proportional to the dimension of the level times the total number of iterations on that level.

Level	Grid	dimension	iterations	ratio	iterations× dimension	per cent total
4	160 × 4 =	640	1548		990720	6.3
				> 8.1		
3	520 × 13 =	6760	191		1291160	8.2
				> 2.9		
2	1760 × 44 =	77440	66		5111040	32.3
				> 4.4		
1	2000 × 147 =	294000	15		4410000	27.9
				> 3.8		
0	2000 × 500 =	1000000	4		4000000	25.3
					total 15802920	

4.4. Non-isotropic permeability field. We performed preliminary tests to determine whether the algorithm can be applied to problems with large aspect ratios in either the domain or the individual grid cells. For the permeability field we used a strip of the fine field generated for the scaling study (full width and 25% of the height). The left half of the new field corresponds to the top half of the permeability field of the base case. The discretized grid of 2000×500 has the same overall dimension, 10^6, as the base case. In addition, as with the base case, the variance is set to 2.

We then reinterpreted the discretized field by assuming a cell aspect ratio of 10 to 1. While the initial field ranged from 0 to 1 in both the x- and y-directions, the new field ranges from 0 to 10 in the x-direction and from 0.25 to 0.5 in the y-direction. The rescaling results in a new permeability field in a channel with a length 40 times its width and in which the typical features (ratio of the correlation lengths) have an aspect ratio of 80 to 1.

The matrix for the problem with a 10 to 1 cell aspect ratio is equivalent to the matrix corresponding to square cells and a non-isotropic permeability field with K_{yy} a factor of 100 times K_{xx}. The flow lines superimposed on the log of the permeabilty field are shown in figure 4.8. The effect of the anisotropy is seen in the abrupt changes in direction of the flow lines in order to follow paths of high permeability.

We found that the solver performed better when the coarsening strategy aimed for cells with an aspect ratio of 1. Thus, despite the smaller dimension, the grid is first coarsened in the y-direction, and then uniformly in both directions. This is similar to the "semi-coarsening" strategy used for multigrid algorithms. The iteration count on each level is shown in table 4.2. The finest level still required only 4 iterations but the semi-coarsening does not reduce the grid dimension by as large a factor for the first two levels and the time per point of $768\,\mu$s is 2.4 times as large as for the base case. We conjecture that a better coarsening algorithm which accounts for off-diagonal components of the permeability tensor would not require the semi-coarsening and that with a uniform reduction in grid size the algorithm would be as efficient as for the base case.

5. Conclusion.

For a large system, any iterative algorithm requires an excellent preconditioner. For PDEs that can be reasonably described by discretization, a coarser discretization forms the basis for a good preconditioner. If our algorithm is well suited for the finest level of discretization, then it is reasonable to expect that it would be equally well suited to the next level. Thus, we are naturally led to a class of preconditioned algorithms which are recursive in nature. The preconditioner applies the very same algorithm on a coarser scale until the problem is either so coarsely resolved that further coarsening is detrimental or the problem is sufficiently small for direct solvers to be more efficient.

We have applied this philosophy to the conjugate gradient algorithm. However, it is very likely that the same ideas could be applied to other solvers. Indeed, the high quality of the preconditioner should be even more valuable for other Krylov space methods which need to store a set of conjugate directions. Memory requirements limit the number of conjugate directions that can be stored. The resulting restarts lower the efficiency of the algorithm. With a better preconditioner fewer iterations and hence fewer restarts can be expected. This should improve the efficiency. In fact for the examples we studied the number of iterations is small enough (about 5) that a restart would have been unnecessary. The small number of iterations would provide a substantial advantage of our method for non-symmetric positive definite operators.

This line of thought is not unique to our approach. Recently, a multigrid preconditioner has been successfully employed in the GMRES algorithm [6]. GMRES and other such more general Krylov space algorithms can be applied when the operator is not symmetric or not positive definite. They significantly broadening the number of problems that can be addressed. PDE's leading to non-symmetric problems are not uncommon.

Even if the problem is symmetric positive definite there may be advantages in using more general solvers. For example, in the porous flow problem addressed here optimum coasening techniques naturally lead to a nonsymmetric permeability tensor on the coarse grid even if the fine grid permeability tensor is symmetric. Our current limitation to symmetric operators did not allow for such coarsening. We believe that the reduced number of iteration from an optimum nonsymmetic coarsening may offset the added cost of an algorithm for a nonsymmetric operator.

In a broader context, our solver is another example demonstrating the advantages of hybrid schemes that combine multigrid and conjugate gradient algorithms. Because of their scaling behavior and robustness, hybrids algorithms are well suited to large problems. Our implementation allows us to define a different preconditioner on every level. The algorithm can be specialized to a particular problem by tailoring the solver on every level to the frequency distribution of scales. From this point of view Tatebe's algorithm and our algorithm are the end points of a large class of multiscale algorithms. Our algorithm has the advantage of a favorable scaling, but for some problems Tatebe's algorithm may be more suitable.

The ideas on which our approach is based are quite general and transcend the specific implementation. In many regards our implementation is a particularly simple example of the general approach. We expect in the future to see similar recursive algorithms that have equally favorable scaling behavior but can be applied to a large class of problems with more complex gridding and coarse graining schemes. In this paper we were not concerned with parallelizing. Tatebe's algorithm has been parallelized [1, 5]. We expect that a similar approach could be applied to parallelize our algorithm and that a significant increase in speed would be obtained for truely large problems.

REFERENCES

[1] S. F. Ashby and R. D. Falgout, *A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations*, Nuclear Science and Engineering, 124 (1996), pp. 143–159.

[2] O. Axelsson, *Iterative Solution Methods*, Cambridge Univ. Press, 1994.

[3] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*, SIAM, Philadelphia, Pa., 1994. `http://www.netlib.org/linalg/html_templates/Templates.html`.

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

[5] L. Brieger and G. Lecca, *Parallel multigrid preconditioning of the conjugate-gradient method for systems of subsurface hydrology*, J. Comput. Phys., 142 (1998), pp. 148–162.

[6] T. T. Feng, D. Peric, and D. R. J. Owen, *A multigrid enhanced GMRES algorithm for elasto-plastic problems*, Intern. J. for Numerical Methods in Engineering, 42 (1998), pp. 1441–1462.

[7] V. V. Shaidurov, *Some estimates of the rate of convergence for cascadic conjugate-gradient method*, Computers Math. Applic., 31 (1996), pp. 161–171.

[8] V. Shashkov and S. Steinberg, *Support-operator finite-difference algorithm for general elliptic problems*, J. Comput. Phys., 118 (1995), pp. 131–151.

[9] O. Tatebe, *The multigrid preconditioned conjugate gradient method*, in Proceedings of the Sixth Copper Mountain conference on Multigrid Methods, 1993, pp. 621–634.

[10] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Review, 34 (1992), pp. 581–613.