# Multigrid Preconditioned Conjugate Gradients for the Numerical Simulation of Groundwater Flow on the Cray T3D

Steven F. Ashby
Lawrence Livermore Nat'l Lab
P.O. Box 808, L-316
Livermore, CA 94551
sfashby@llnl.gov

Robert D. Falgout
Lawrence Livermore Nat'l Lab
P.O. Box 808, L-316
Livermore, CA 94551
rfalgout@llnl.gov

Steven G. Smith
Lawrence Livermore Nat'l Lab
P.O. Box 808, L-316
Livermore, CA 94551
smith84@llnl.gov

Thomas W. Fogwell
IT Corporation
4585 Pacheo Blvd.
Martinez, CA 94553
510-372-5210

## Abstract

This paper discusses the numerical simulation of groundwater flow through heterogeneous porous media. The focus is on the performance of a parallel multigrid preconditioner for accelerating convergence of conjugate gradients, which is used to compute the hydraulic pressure head. The numerical investigation considers the effects of enlarging the domain, increasing the grid resolution, and varying the geostatistical parameters used to define the subsurface realization. The results were obtained using the ParFlow groundwater flow simulator on the Cray T3D massively parallel computer.

## Introduction

Effective environmental remediation and prudent natural resource management are important national priorities attracting increased attention from government and industry alike. In the area of environmental remediation, the U.S. Department of Energy is spending billions of dollars (nearly one third of its budget) on the cleanup of its contaminated sites, as well as on the development of promising new technologies such as dynamic steam stripping and biofilters. Companies also are spending large amounts of money on the cleanup of contaminated facilities. In the area of water resource management, the State of California and various municipalities are keenly interested in managing limited water supplies more wisely. Finally, the cost-effective management of oil reservoirs is more important than ever as oil companies seek to wring additional profits from maturing fields.

The numerical simulation of subsurface fluid flow and chemical migration plays an increasingly important role in these applications. Sophisticated simulations have been used for decades in the petroleum industry with considerable success. Computational environmental remediation is less widely used, but it is gaining in popularity as sites becomes larger and more complex. It is particularly attractive for the design, evaluation, and management of engineered remediation procedures [1]. Simulations can be used, for instance, to choose the best cleanup strategy for a given site, and then, once a scheme is chosen, to manage it in the most cost effective fashion. They also can be used to perform more realistic risk assessment in support of key decision-making, as well as an aid in demonstrating regulatory compliance.

## The Need for High Performance Computing Technology

Many of the computer codes in use today make unrealistic assumptions about the nature of the subsurface medium and the associated flow behavior. For example, many codes assume that the subsurface is homogeneous in composition and spatial distribution. In reality, the subsurface is three-dimensional and heterogeneous. These heterogeneities result in *preferential flow channels*, which can have a dramatic impact on flow and contaminant transport [2]. Consequently, the heterogeneous nature of the subsurface must be taken into account if one is to draw reliable conclusions from the numerical simulations.

The size of the site to be modeled (typically several square kilometers) and the need to resolve these heterogeneities adequately (on the order of meters), leads to computational domains with upwards of one billion spatial zones. The use of adaptive gridding and local refinement can reduce the total number of zones needed by one or two orders of magnitude, but one is still left with huge problems that quickly overwhelm all but the largest of conventional supercomputers. The problem is all the more acute when one wishes to run large, time-dependent simulations repeatedly, as when one undertakes detailed parameter studies or risk assessments. In light of these considerations, it is necessary to employ massively parallel processing power, and toward this end, we are building a parallel flow simulator called ParFlow. It is designed to be portable and scalable across a variety of distributed memory MIMD machines with message passing, ranging from workstation clusters to large MPPs. See [3, 4] for a brief discussion of the parallel implementation and performance of ParFlow.

Massively parallel computing platforms may be necessary for detailed simulations, but they are not sufficient. One also needs to employ high performance *algorithms*, that is, accurate and fast numerical techniques that can be implemented efficiently on these machines. As we will see, simply changing the linear solver can result in nearly two orders of magnitude reduction in CPU time. This is especially important in time-dependent simulations, where the right numerical method can mean the difference between a 30-*hour* run and a 30-*minute* run on an MPP.

Of course, one never has enough data (i.e., direct measurements) to characterize a given site completely. To develop the detailed subsurface characterization needed for the type of simulation described above, hydrogeologists typically employ geostatistical techniques such as *turning bands* (see below) to create statistically accurate *realizations* of key subsurface properties, particularly the hydraulic conductivity [5]. Monte Carlo and optimization techniques can be used to quantify the inherent uncertainty and enable site managers to perform more realistic risk assessments. These techniques also can be used to evaluate various remediation strategies, say, determining the optimal pumping configuration in a pump-and-treat scheme.

## Numerical Simulation of Groundwater Flow

In this paper, we consider steady state saturated (i.e., single phase) flow, which is of practical interest because contaminant transport is most rapid in this region. It is essential that we be able to solve such problems quickly and accurately because they will constitute the main computational cost of the multiphase, time-dependent simulations in which we are ultimately interested. In particular, we need an efficient and scalable elliptic solver, which we have found in the multigrid preconditioned conjugate gradient algorithm described below.

Our mathematical model of groundwater flow is based on Darcy's Law and conservation of mass in a porous medium, which may be combined and rewritten as

$$- \nabla \cdot (K \nabla (h + z)) - Q = 0 \tag{1}$$

where $h$ is the pressure head (from which the flow velocity field is determined), $K$ is the hydraulic conductivity (i.e., subsurface realization), and $Q$ is a source term (used to represent pumping wells, for example). At present, the problem domain is assumed to be a parallelepiped; the boundary conditions may be Dirichlet, Neumann, or mixed. In the experiments below, we enforce no flow (Neumann) conditions on the top and the bottom, and constant head (Dirichlet) conditions on the vertical faces.

The hydraulic conductivity realization is central to the problem definition; it is embodied in the $K$ function in equation (1). We are using Tompson's turning bands algorithm [5] to generate $K$. This is a technique for computing a spectral random field with given statistical properties (mean $\mu$, variance $\sigma^2$, and correlation lengths $\lambda_x$, $\lambda_y$, and $\lambda_z$). The algorithm has two phases: the generation of lines and the projection of grid points onto those lines. In the first phase, a number of rays through the origin (typically 50-100) are created, and one-dimensional random fields are generated along these. In the second phase, these 1D fields are combined to produce a 3D random field. Specifically, each point is projected onto the lines, and these values are combined to determine the value of the 3D field at the given point. See [4] for a discussion of the parallel implementation and performance of turning bands.

### *Discrete solution approach*

We employ a standard 7-point finite volume spatial discretization on a uniform mesh (but we plan to incorporate local mesh refinement and adaptive gridding). After discretization, we obtain a large system of linear equations, $Ah = f$. The coefficient matrix $A$ is symmetric positive definite and has the usual seven stripe pattern. The matrix has order $N = n_x \times n_y \times n_z$, where the $n_i$ are the number of grid points in the $x$, $y$, and $z$ directions, respectively. For problems of interest, $N$ is in the millions; the large number is dictated by the size of the physical site and the need to resolve heterogeneities adequately. Once the pressure head is computed, the velocity field can be calculated easily using a simple differencing scheme. This field is then passed to a transport code to simulate contaminant migration.

The solution of the large linear system is computationally intensive and must be done efficiently and accurately. Since we are interested in detailed simulations (i.e., high resolution), we must use an iterative scheme. Within the hydrology community, the most commonly used methods are SIP and SSOR. Although adequate for many problems, these methods suffer in comparison to the conjugate gradient method [6]. In this method, the energy norm of the error is minimized over a Krylov subspace at each step of the iteration, thereby yielding an optimal iterative method. When combined with a good preconditioner, the resulting preconditioned conjugate gradient (PCG) method [7] is an extraordinarily powerful tool. For example, polynomial preconditioned conjugate gradients (PPCG) was shown [8] to be an order of magnitude faster than SIP and SSOR. In this paper, we present results for multigrid and 2-step Jacobi preconditionings; the algorithms are denoted MGCG and J2CG, respectively. Our emphasis will be on the multigrid preconditioner, which consists of a single V-cycle with semi-coarsening; it is described more fully in the next section. The 2-step Jacobi preconditioner is implemented via an inner iteration consisting of two steps of the basic Jacobi method; see, e.g., [9, pages 384-385] for details.

## *Multigrid preconditioned conjugate gradients*

Multigrid techniques are among the fastest methods currently available for the solution of linear systems arising from the discretization of elliptic partial differential equations. Unlike most other iterative methods, a good multigrid solver's rate of convergence is independent of problem size, meaning that the number of iterations remains fairly constant. Hence, both the multigrid algorithm *and* its parallel implementation are scalable. On the other hand, multigrid (MG) algorithms tend to be highly problem specific and generally less robust than Krylov iterative methods such as conjugate gradients (CG). Fortunately, it is easy to combine the best features of MG and CG into one algorithm: multigrid preconditioned conjugate gradients (MGCG). The resulting algorithm is robust, efficient, and scalable. Another advantage of this approach is that one can quickly implement a simple multigrid algorithm that is extremely effective as a preconditioner, but perhaps less effective as a stand-alone solver. This is especially valuable when the underlying PDE has a nearly discontinuous coefficient function, as in our case.

Our current implementation of MGCG is simple but effective. The MG preconditioning step consists of a single standard *V-cycle* with weighted Jacobi as the smoother. Since our problems tend to be anisotropic, we employ a *semi-coarsening* strategy in which the grid is coarsened in one spatial direction at a time. To determine the direction of semi-coarsening, we use a heuristic based on the grid spacing: the algorithm chooses a direction with smallest spacing (i.e., strongest coupling). If this minimum spacing occurs in one or more directions, the algorithm attempts to coarsen first in $x$, then in $y$, and finally in $z$. The coarsest grid is never allowed to get smaller than $3 \times 3 \times 3$, and we currently use a fixed number (1) of unweighted Jacobi iterations to approximate the solution on this grid. In the future, we will implement other smoothings, including pointwise red-black.

Although this semi-coarsening strategy effectively ameliorates anisotropies due to large grid aspect ratios, it does not deal with anisotropies in the rock matrix (permeability tensor). Some recent work has been done in this area (e.g., [10, 11]), and we plan to investigate these and other techniques in the future.

One of the keys to a successful multigrid algorithm is the choice of the prolongation operator, $P$, which defines how vectors on a coarse grid are mapped onto the next finer grid. (The restriction operator, used to project from a fine to coarse grid, is usually taken to be $P^T$.) In the case of constant coefficient elliptic PDEs, $P$ is usually defined via a simple interpolation scheme. However, when the coefficient function varies greatly, as in our problem, this is inappropriate. Instead, one should use *operator-induced* prolongation, meaning that $P$ is defined in terms of the coefficients of the fine grid matrix. Our prolongation operator is similar to those used in [12, 13, 14].

To elucidate, consider the prolongation of a vector, $e_c$, from the coarse grid, $\mathcal{G}_c$, to the fine grid, $\mathcal{G}$. For the sake of discussion, let us assume that $\mathcal{G}_c$ is obtained by coarsening $\mathcal{G}$ in the $x$-direction. Then, if $W$, $E$, $S$, $N$, $L$, $U$, and $C$ denote the west, east, south, north, lower, upper, and center stencil values, respectively, we define

$P$ algebraically by

$$Pe_c = \begin{cases} P^L_{i,j,k} e^c_{i-1,j,k} + P^R_{i,j,k} e^c_{i+1,j,k} , & x_{i,j,k} \in \mathcal{G} \setminus \mathcal{G}_c \\ e^c_{i,j,k} , & x_{i,j,k} \in \mathcal{G}_c \end{cases} \tag{2}$$

where

$$\begin{aligned} P^L_{i,j,k} &= W_{i,j,k} / P^C_{i,j,k} \\ P^R_{i,j,k} &= E_{i,j,k} / P^C_{i,j,k} \\ P^C_{i,j,k} &= C_{i,j,k} - S_{i,j,k} - N_{i,j,k} - L_{i,j,k} - U_{i,j,k} . \end{aligned}$$

Prolongation operators in $y$ and $z$ are defined analogously. The coarse-grid coefficient matrix is also defined algebraically, and has a 7-point stencil structure. This operator is intended to be an approximation to the 27-point Galerkin matrix, $P^TAP$, which requires considerably more storage.

Finally, we remark that when designing a preconditioner for CG, one needs to insure that the corresponding matrix is symmetric, and preferably positive definite. For multigrid preconditioning, this condition is satisfied by doing an equal number of symmetric smoothing steps both before and after each coarse-grid correction. (The smoothing step is symmetric if the iteration matrix of the associated method is symmetric.) However, this is not necessarily required (see, e.g., [15]). Multigrid algorithms also can be applied to nonsymmetric problems (e.g., [16]) and to problems with irregular meshes (e.g., [17]).

### *Implementation*

The computational approach described above has been implemented in ParFlow, a portable and scalable parallel flow simulator. We have successfully run ParFlow (in various incarnations) on the following platforms: a single Sparcstation, a cluster of Sparcstations, a multiprocessor SGI Onyx, an nCUBE/2, an IBM SP-1, and the Cray T3D. Portability is realized via message-passing. The problem data is distributed across a virtual 3D process grid consisting of $P = p \times q \times r$ processes; the grid points within a process are arranged as a 3D subgrid. The computations are organized so as to avoid explicit data redistribution, thereby improving the code's efficiency.

Although each process has a piece of the problem domain, we are *not* doing domain decomposition in the algorithmic sense. We are solving the *full* problem rather than independent subproblems. When we compute a matrix-vector product, we use the matrix $A$ corresponding to the global problem. We view this coefficient matrix as a stencil distributed across the processes analogous to the data distribution. To compute the matvec result at a given grid point $(i, j, k)$, we "apply" the stencil to the grid: For each neighboring grid point specified by the stencil, we multiply the vector value at that point by the corresponding stencil coefficient, and then sum these products. The parallel implementation of this routine is discussed in [3].

## Numerical Results

Our model problem corresponds to a physical domain of size $1260 \times 1260 \times 31$ with $N = 129 \times 129 \times 65$ grid points. The hydraulic conductivity field was generated using a turning bands algorithm with parameters $\mu = 4$, $\sigma = 1.0$, $\lambda_x = 40$, $\lambda_y = 40$, and $\lambda_z = 2$. We impose Dirichlet boundary conditions (hydraulic head value of 1) on the four vertical sides of the domain, and no flow conditions on the top and bottom. The PCG algorithms (MGCG and J2CG) were halted once the 2-norm of the relative residual was less than $10^{-9}$. All times are wall-clock times, and they are given in seconds. Although the test problems are contrived, they serve to illustrate the performance of the MGCG algorithm. We are currently working with colleagues in the Laboratory's Environmental Protection Department to incorporate their model of the LLNL site into ParFlow.

### *PCG convergence study*

In Table 1 we vary the degree of heterogeneity, represented by the standard deviation, $\sigma$, for the model problem. The first row in the table, $\sigma = 0.0$, corresponds to a homogeneous subsurface medium, in which case the coefficient

| $\sigma$ | J2CG iters | J2CG time | MGCG iters | MGCG time |
|---|---|---|---|---|
| 0.0 | 1593 | 929.6 | 14 | 23.3 |
| 0.5 | 3039 | 1772.5 | 15 | 24.9 |
| 1.0 | 3302 | 1926.0 | 17 | 28.1 |
| 1.5 | 3689 | 2152.4 | 30 | 48.4 |
| 2.0 | 4352 | 2538.5 | 68 | 103.7 |

Table 1: *Varying the degree of heterogeneity: the problem size is fixed at* $N = 129 \times 129 \times 65$ *(for a domain of size* $1260 \times 1260 \times 31$*), and the standard deviation,* $\sigma$*, is varied. The other geostatistical parameters are fixed:* $\mu = 4$, $\lambda_x = 40$, $\lambda_y = 40$, $\lambda_z = 2$. $P = 1 \times 4 \times 4$.

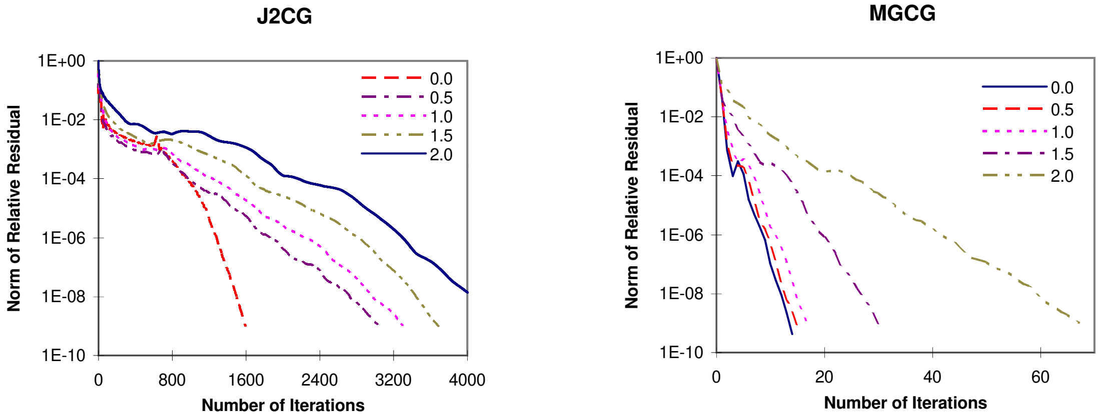


Figure 1: *Convergence plots for J2CG and MGCG for varying values of* $\sigma$ *(as in Table 1).*

function $K$ is constant, and so the matrix $A$ is Laplacian-like. As $\sigma$ increases, so does the degree of heterogeneity, and the matrix $A$ becomes more ill-conditioned as the coefficient function $K$ in equation (1) varies more widely. The degree of variability is proportional to $e^{2\sigma}$. Convergence plots for J2CG and MGCG are shown in Figure 1 for each of the values of $\sigma$ in Table 1. Notice that the MGCG convergence curves are nearly linear and quite steep in comparison to J2CG, indicating that MGCG is making rapid and steady progress toward the solution.

In some remediation studies, one wishes to enlarge the initial site to encompass neighboring property. This might be necessary, for instance, if a contaminant were discovered to have migrated off-site. In such a scenario, the engineer might wish to use the same geostatistics and "stretch" the domain. This situation is considered in Table 2, where we use the geostatistical parameters from the model problem and fix the grid spacing ($\Delta_x = 20$, $\Delta_y = 20$, $\Delta_z = 1$), but "grow" the domain by increasing the number of spatial zones. As one would expect, the time to solution increases with the problem size $N$. The J2CG iteration count increases dramatically—again as expected. The number of MGCG iterations, on the other hand, remains nearly constant—once $n_z$ is large enough. The iteration counts for the first two problems ($n_z = 9, 17$) are higher because there are not enough grid points in the $z$ direction for the semi-coarsening strategy to realize its full potential: Since the grid aspect ratio is 20:20:1, the algorithm needs to coarsen four or five times in the vertical direction ($2^4 < 20 < 2^5$), but there are not enough points.

We see nearly identical qualitative and quantitative behavior in Table 3, where the problem domain and geostatistics are the same as in Table 1 (with $\sigma = 1.0$), but the grid resolution is increased. This means that we are increasing the number of grid points per correlation length, which has the effect of smoothing the PDE coefficient function $K$. As in the previous table, MGCG converges in a near constant number of iterations despite the increasing problem size. Again, the larger iteration counts in the first two rows is probably a result of the semi-coarsening strategy.

In Table 4 we study the effect of varying the grid aspect ratio. To isolate the effect of the grid aspect ratio, we

| Problem Size | | | Process Grid | J2CG | | MGCG | |
|---|---|---|---|---|---|---|---|
| $n_x$ | $n_y$ | $n_z$ | $P = p \times q \times r$ | iters | time | iters | time |
| 17 | 17 | 9 | $1 \times 4 \times 4$ | 391 | 2.8 | 28 | 1.1 |
| 33 | 33 | 17 | $1 \times 4 \times 4$ | 883 | 14.0 | 21 | 1.7 |
| 65 | 65 | 33 | $1 \times 4 \times 4$ | 1669 | 125.5 | 16 | 4.4 |
| 129 | 129 | 65 | $1 \times 4 \times 4$ | 3284 | 1915.5 | 17 | 26.5 |
| 257 | 257 | 129 | $4 \times 4 \times 8$ | 6444 | 7394.7 | 16 | 50.0 |

Table 2: *Growing domain: the grid spacing is fixed ($\Delta x = 20$, $\Delta y = 20$, $\Delta z = 1$) while the number of grid points is increased. Note that the number of grid points per correlation length remains constant (the geostatistical parameters are as in the model problem).*

| Problem Size | | | Process Grid | J2CG | | MGCG | |
|---|---|---|---|---|---|---|---|
| $n_x$ | $n_y$ | $n_z$ | $P = p \times q \times r$ | iters | time | iters | time |
| 17 | 17 | 9 | $1 \times 4 \times 4$ | 424 | 3.1 | 29 | 1.2 |
| 33 | 33 | 17 | $1 \times 4 \times 4$ | 875 | 13.9 | 20 | 1.6 |
| 65 | 65 | 33 | $1 \times 4 \times 4$ | 1736 | 130.7 | 16 | 4.4 |
| 129 | 129 | 65 | $1 \times 4 \times 4$ | 3302 | 1926.0 | 17 | 28.0 |
| 257 | 257 | 129 | $4 \times 4 \times 8$ | 6381 | 7247.6 | 17 | 53.0 |

Table 3: *Increasing resolution: the geostatistical parameters are fixed ($\mu = 4$, $\sigma = 1$, $\lambda_x = 40$, $\lambda_y = 40$, $\lambda_z = 2$) and the number of grid points per correlation length is increased. The domain size is* $1260 \times 1260 \times 31$.

vary the correlation lengths and grid spacing in tandem to maintain a constant two grid points per correlation length (the minimum that makes physical sense). We call this the $\frac{\lambda}{\Delta}$-ratio, and it is 2:2:2 in these experiments. The grid aspect ratio has a significant impact on the rate of convergence of J2CG: As the the grid becomes more skewed, the iteration takes longer to converge. However, one also must take into account the boundary conditions. When there is strong coupling in a direction corresponding to a Dirichlet condition (the $x$ and $y$ directions in our experiments), the problem is easier than for a Neumann condition (compare rows 2, 3, and 4). This is a consequence of how the eigenvalues of $A$ are distributed. When the grid aspect ratio is near 1:1:1, the eigenvalues are more tightly clustered in the middle of the spectrum, and the *effective* condition number is less than the *true* condition number. When the grid is skewed, the eigenvalues cluster near the endpoints of the spectrum, and the effective and true condition numbers are nearly identical. Moreover, the component of the eigenvalue due to the Neumann boundary condition is much smaller than for a Dirichlet condition, further increasing the effective condition number. (Recall that the CG rate of convergence is governed by the effective condition number, not the true condition, because CG is able to damp outlying eigenvalues quickly.) MGCG behaves quite differently. The semi-coarsening strategy is designed to ameliorate the effects of a poor grid aspect ratio, and as we see, it works quite well. As with J2CG, it works best when there is strong coupling in at least one direction corresponding to a Dirichlet boundary condition.

In Table 5 we fix the grid aspect ratio at 20:20:1 and look at the effect of the varying the correlation lengths. In general, we expect the J2CG and MGCG iteration counts to decrease as the $\frac{\lambda}{\Delta}$-ratio increases because the coefficient matrix $A$ becomes smoother. However, we have noticed in these and other experiments that the iteration counts also depend on the specific subsurface realization. An interesting experiment would be to look at the trends over a range of subsurface realizations (e.g., vary the random number seed in the turning bands generator).

## *Parallel performance on the Cray T3D*

In Figures 2 and 3, we report efficiencies and scaled speedups for the matrix-vector multiplication, multigrid preconditioning, and MGCG routines for the Cray T3D. Our machine has 128 nodes, each consisting of a 150MHz DEC Alpha processor and 64MB of memory; our message-passing library is layered on top of PVM. We

| $\lambda_x$ | $\lambda_y$ | $\lambda_z$ | $\Delta_x$ | $\Delta_y$ | $\Delta_z$ | J2CG iters | J2CG time | MGCG iters | MGCG time |
|---|---|---|---|---|---|---|---|---|---|
| 40 | 40 | 40 | 20 | 20 | 20 | 350 | 204.5 | 15 | 22.2 |
| 2 | 40 | 40 | 1 | 20 | 20 | 525 | 306.7 | 12 | 18.6 |
| 40 | 2 | 40 | 20 | 1 | 20 | 533 | 311.5 | 12 | 18.2 |
| 40 | 40 | 2 | 20 | 20 | 1 | 3284 | 1915.8 | 16 | 26.5 |
| 40 | 2 | 2 | 20 | 1 | 1 | 607 | 354.3 | 11 | 16.5 |
| 2 | 2 | 40 | 1 | 1 | 20 | 421 | 249.0 | 11 | 16.7 |
| 40 | 40 | 4 | 20 | 20 | 2 | 1683 | 981.7 | 15 | ($*$) 16.1 |
| 40 | 4 | 4 | 20 | 2 | 2 | 588 | 347.5 | 11 | 16.8 |

Table 4: *Correlation lengths and grid spacing are varied in tandem to maintain constant* $\frac{\lambda}{\Delta}$*-ratio of 2:2:2. The problem size* $N = 129 \times 129 \times 65$ *with* $\mu = 4$ *and* $\sigma = 1.0$. $P = 1 \times 4 \times 4$ *except for* ($*$)*, which used* $P = 2 \times 4 \times 4$.

| $\lambda_x$ | $\lambda_y$ | $\lambda_z$ | $\frac{\lambda}{\Delta}$-ratio | J2CG iters | J2CG time | MGCG iters | MGCG time |
|---|---|---|---|---|---|---|---|
| 40 | 40 | 2 | 2:2:2 | 3284 | 1081.2 | 16 | 14.7 |
| 40 | 40 | 4 | 2:2:4 | 3498 | 1161.7 | 16 | 14.7 |
| 40 | 40 | 40 | 2:2:40 | 3870 | 1271.0 | 19 | 17.3 |
| 80 | 80 | 4 | 4:4:4 | 3270 | 1077.4 | 18 | 16.4 |
| 160 | 80 | 4 | 8:4:4 | 3106 | 1022.1 | 17 | 15.5 |

Table 5: *Correlation lengths are varied for a fixed spatial grid (*$N = 129 \times 129 \times 65$ *with* $\Delta_x = 20$, $\Delta_y = 20$, $\Delta_z = 1$*; and* $\sigma = 1.0$, $\mu = 4$*).* $P = 4 \times 4 \times 4$.

expect much better results when we replace PVM with Cray's SHMEM library. (For a more complete discussion of the parallel performance of these and other ParFlow routines, see [4].) In calculating the efficiency, we fixed the total problem size at $N = 64 \times 64 \times 32$ and measured the time, $T_P$, as the number of processes $P$ increased. The efficiencies (given by $T_1/PT_P$) drop off sharply because the problem size per process is too small for larger $P$. (When $P = 128$, we have only 1024 grid points per process, and there is too little work for the amount of communication.) This poor performance is also due, to a possibly greater extent, to the inefficiency of the message-passing routines, and we are currently investigating various ways to improve them. Scaled speedup is defined to be $T_1/T_P$, where now the total problem size is allowed to grow with $P$. In our experiments, each processor was given $2^{17} \approx 10^5$ grid points. Note that both the MG preconditioner and the MGCG algorithm appear to be fairly scalable, meaning that the time per iteration remains constant as we increase the problem size *and* number of processes in tandem. To calculate the efficiency and scaled speedup graphs, it was necessary to vary the $p : q : r$ and $n_x : n_y : n_z$ ratios. As discussed in [4], this has a dramatic impact on node performance, which (at least in part) explains the erratic performance graphs. In these experiments, we chose $p = 1$, which gives the best overall time to solution. (Overall MFLOP rates were observed to vary by more than an order of magnitude depending on the number of grid points in the innermost loop of the matvec routine; see [4].)

## Acknowledgements

We acknowledge the valuable participation of Charles Baldwin, John Bell, William Bosl, Gwen Loosmore, Andrew Tompson, and John Ziagos in the ParFlow project. In particular, we thank Dr. Andrew Tompson for his patient tutoring in the application area, especially in the use of geostatistics.

The Cray T3D experiments described in this paper were run on the 128-node machine located in the Center for Computational Sciences and Engineering at Lawrence Livermore National Laboratory as part of the H4P Industrial Computing Initiative funded by DOE Defense Programs. Early code development was done on the Cray T3D located at the Pittsburgh Supercomputer Center.

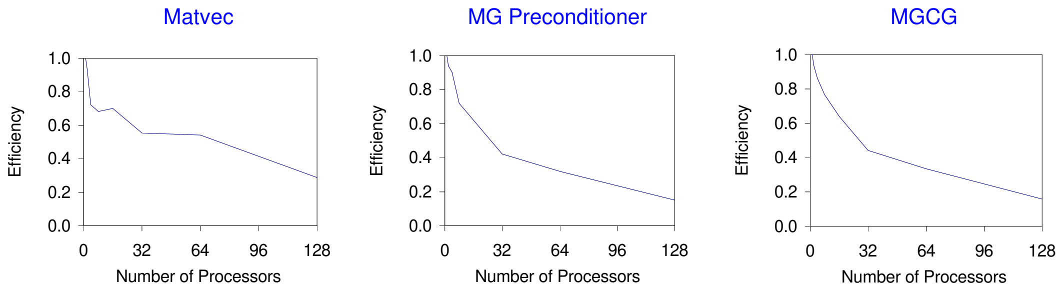


Figure 2: *Parallel efficiency on the Cray T3D.*

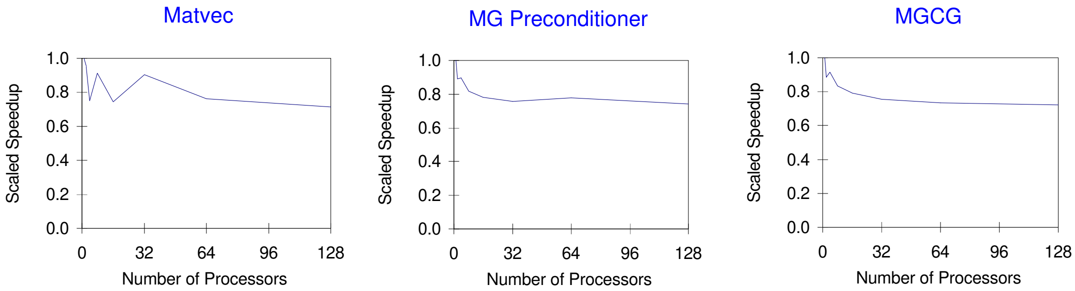


Figure 3: *Scaled speedup on the Cray T3D.*

This work was performed at Lawrence Livermore National Laboratory (LLNL) under contract W-7405-ENG-48 for the U.S. Department of Energy (DOE). It was supported in part by the DOE Defense Programs Technology Transfer Initiative, the Laboratory Directed Research and Development program at LLNL, and by the Applied Mathematical Sciences program of the DOE Office of Scientific Computing.

## References

[1] A. F. B. Tompson, S. F. Ashby, R. D. Falgout, S. G. Smith, T. W. Fogwell, and G. A. Loosmore. Use of high performance computing to examine the effectiveness of aquifer remediation. In A. Peters, G. Wittum, B. Herrling, U. Meissner, C. Brebbia, W. Gray, and G. Pinder, editors, *Proc. X International Conference on Computational Methods in Water Resources*, volume 2. Kluwer Academic Publishers, Dordrecht, 1994. Held in Heidelberg, Germany, July 19-22, 1994. Also available as LLNL technical report UCRL-JC-115374.

[2] R. Ababou, D. B. McLaughlin, L. W. Gelhar, and A. F. B. Tompson. Numerical simulation of three-dimensional saturated flow in randomly heterogeneous porous media. *Transport in Porous Media*, 4(6):549–565, 1989.

[3] S. F. Ashby, R. D. Falgout, S. G. Smith, and A. F. B. Tompson. Modeling groundwater flow on MPPs. In A. Skjellum and D. Resse, editors, *Proc. 1993 Scalable Parallel Libraries Conference*, pages 17–25. IEEE Computer Society Press, 1994. Held at Mississippi State University, October 6–8, 1993. Also available as LLNL technical report UCRL-JC-115602.

[4] S. F. Ashby, R. D. Falgout, S. G. Smith, and A. F. B. Tompson. The parallel performance of a groundwater flow code on the CRAY T3D. In *Proc. Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pages 131–136. Society for Industrial and Applied Mathematics, 1995. Held in San Francisco, February 15–17, 1995. Also available as LLNL technical report UCRL-JC-118604.

[5] A. F. B. Tompson, R. Ababou, and L. W. Gelhar. Implementation of of the three-dimensional turning bands random field generator. *Water Resources Res.*, 25(10):2227–2243, 1989.

[6] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49:409–435, 1952.

[7] P. Concus, G. H. Golub, and D. P. O'Leary. A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. In J. R. Bunch and D. J. Rose, editors, *Sparse Matrix Computations*, pages 309–332. Academic Press, New York, 1976.

[8] P. D. Meyer, A. J. Valocchi, S. F. Ashby, and P. E. Saylor. A numerical investigation of the conjugate gradient method as applied to three-dimensional groundwater flow problems in randomly heterogeneous porous media. *Water Resources Res.*, 25(6):1440–1446, 1989.

[9] G. H. Golub and J. M. Ortega. *Scientific Computing: An Introduction with Parallel Computing.* Academic Press, San Diego, CA, 1993.

[10] W. Hackbusch. The frequency decomposition multigrid method, part I: Application to anisotropic equations. *Numer. Math.*, 56:229–245, 1989.

[11] N. H. Naik and J. R. van Rosendale. The improved robustness of multigrid elliptic solvers based on multiple semicoarsened grids. *SIAM J. Numer. Anal.*, 30:215–229, 1993.

[12] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter. The multi–grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2:430–454, 1981.

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

[14] T. W. Fogwell and F. Brankhagen. Multigrid method for the solution of porous media multiphase flow equations. In *Nonlinear Hyperbolic Equations – Theory, Computation Methods, and Applications*, volume 24 of *Notes on Numer. Fluid Mech.*, pages 139–148. Vieweg, Braunschweig, 1989.

[15] O. Tatebe. The multigrid preconditioned conjugate gradient method. In N. D. Melson, T. A. Manteuffel, and S. F. McCormick, editors, *Sixth Copper Mountain Conference on Multigrid Methods*, volume CP 3224, pages 621–634, Hampton, VA, 1993. NASA.

[16] J. E. Dendy. Black box multigrid for nonsymmetric problems. *Appl. Math. Comput.*, 13:261–284, 1983.

[17] D. J. Mavriplis and A. Jameson. Multigrid solution of the Euler equations on unstructured and adaptive meshes. In S. F. McCormick, editor, *Multigrid Methods: Theory, Applications, and Supercomputing*, volume 110 of *Lecture Notes in Pure and Applied Mathematics*, pages 413–429. Marcel Dekker, New York, 1988.