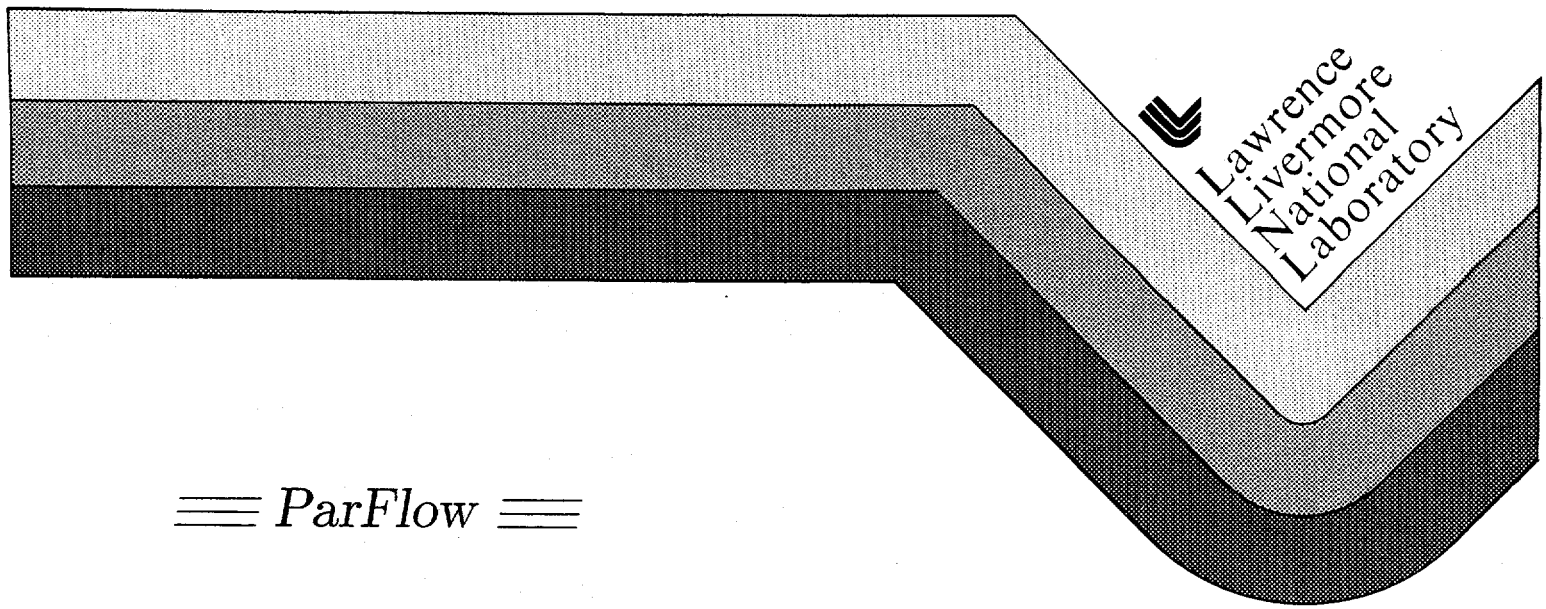
A Scalable Approach to Modeling Groundwater Flow on Massively Parallel Computers

Steven F. Ashby
Robert D. Falgout
Center for Computational Sciences & Engineering

Andrew F. B. Tompson
Earth Sciences Division of Environmental Programs



ParFlow

UCRL-JC-122591
December 1995

THE PARFLOW PROJECT

The goal of the ParFlow project is to apply high performance computing techniques to the three-dimensional modeling of fluid flow and chemical transport in heterogeneous porous media to enable more realistic simulations of subsurface contaminant migration. This is an interdisciplinary effort involving Laboratory scientists from the Center for Computational Sciences & Engineering, the Earth Sciences Division of Environmental Programs, and the Environmental Protection Department. Lawrence Livermore National Laboratory, International Technology Corporation, and Cray Research, Incorporated are partnering to commercialize some of the work described herein under the auspices of one or more Cooperative Research and Development Agreements (CRADAs).

Various aspects of this work were funded by one or more of the following: DOE Office of Scientific Computing (Applied Mathematical Sciences), DOE Defense Programs (Technology Transfer Initiative), and Laboratory Directed Research and Development.

DISCLAIMER

REPRINT

This is a reprint of a paper that will appear in the Proceedings of the U.S. EPA Workshop on Next Generation Environmental Models Computational Methods, which was held August 7-9, 1995 at the EPA's National Environmental Supercomputing Center in Bay City, MI.

A Scalable Approach to Modeling Groundwater Flow on Massively Parallel Computers

Steven F. Ashby
Robert D. Falgout
Andrew F. B. Tompson

Lawrence Livermore National Laboratory

Abstract

We describe a fully scalable approach to the simulation of groundwater flow on a hierarchy of computing platforms, ranging from workstations to massively parallel computers. Specifically, we advocate the use of *scalable conceptual models* in which the subsurface model is defined independently of the computational grid on which the simulation takes place. We also describe a scalable multigrid algorithm for computing the groundwater flow velocities. We are thus able to leverage both the engineer's time spent developing the conceptual model *and* the computing resources used in the numerical simulation. We have successfully employed this approach at the LLNL site, where we have run simulations ranging in size from just a few thousand spatial zones (on workstations) to more than eight million spatial zones (on the CRAY T3D)—all using the same conceptual model.

1 Introduction

Groundwater contamination is a serious problem throughout the United States and the world. At the Lawrence Livermore National Laboratory (LLNL), for instance, chemical solvents were dumped onto the ground surface in the 1940's (when the present site was a naval air station), and over time, these contaminants (now classified as hazardous wastes) have migrated through the unsaturated zone and into the more mobile groundwaters (Figure 1). These contaminants are moving slowly (25m/year) toward aquifers that provide drinking water for the city of Livermore. LLNL, as the current property owner, is obligated (under the SuperFund Act) to determine the extent of the contamination (i.e., plume distribution), and then design, implement, and monitor remediation procedures. Similar problems plague the DOE complex (most notably the Hanford, Savannah River, and Nevada Test Site facilities), as well as U.S. industry.

In carrying out their cleanup mission, colleagues in the Environmental Protection Department have installed over 400 monitoring wells. These wells provide data that is used to characterize the nature of the subsurface medium and the contaminant plume. For example, hydrologists use well tests and core samples to determine the types of soils comprising the subsurface. Direct measurements of contaminant concentrations also are made. Once the subsurface is characterized and the contaminant plume is identified, various remediation procedures are evaluated. The most commonly used procedure is pump-and-treat, in which the contaminated groundwater is extracted via a pumping well and treated above ground. Although pump-and-treat is widely used, it is slow and expensive. For example, it might take decades and hundreds of millions of dollars to clean up a site like LLNL. Consequently, environmental remediation engineers at LLNL and elsewhere are studying alternative, more cost-effective cleanup strategies. These include microbial biofilters [16] and dynamic steam stripping [14], both of which offer the promise of substantially reduced cleanup times, yielding potential savings of tens of millions of dollars.

Numerical simulations are increasingly being used to help guide the remediation process. Engineers use simulations to evaluate competing remediation strategies and to choose the best one for a given site. Once a procedure has been chosen, additional simulations can be used to manage it in the most cost-effective fashion. For example, simulations can be used to determine

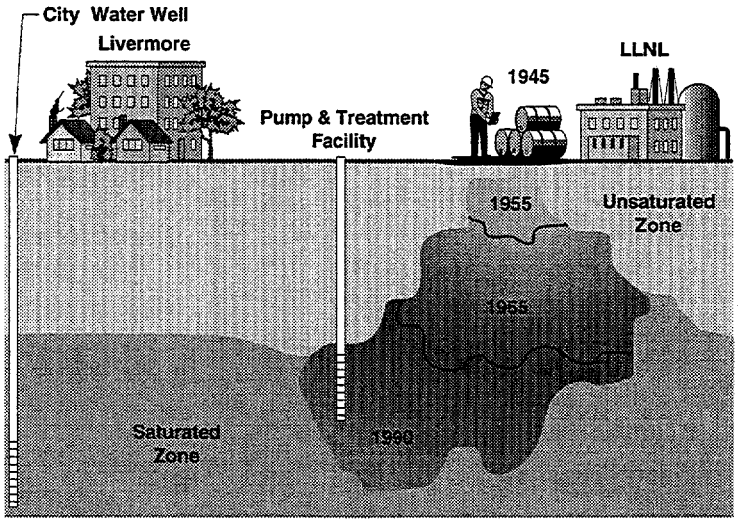


FIG. 1. *Contaminants have migrated through the unsaturated zone into the more mobile groundwaters at LLNL. We are working with colleagues in the Environmental Protection Department to develop a scalable model of the subsurface for use in the ParFlow simulator.*

the optimal pumping configuration (i.e., well locations and pumping rates) in a pump-and-treat strategy. Such a priori design optimization [8] can save millions of dollars over the course of a cleanup effort. Simulations also can be used to perform rational, scientific risk assessment and to demonstrate regulatory compliance.

1.1 Modeling a Complex Subsurface

Most simulation codes in use today make unrealistic assumptions about the nature of the subsurface medium and its associated flow behavior. These codes are generally incapable of modeling the complicated phenomena that arise in fluid flow through three-dimensional porous media. For example, many codes assume that the subsurface is one- or two-dimensional and homogeneous in composition and spatial distribution. The few three-dimensional codes that exist typically assume that the subsurface is composed of uniform layers (as in Figure 2, left). Such codes rely on the use of "effective" (or average) medium properties, for example, the use of a single hydraulic conductivity value to represent an entire layer. In reality, the subsurface is fully heterogeneous, that is, each layer is itself heterogeneous (as in Figure 2, right). This means that the hydraulic conductivity is spatially varying within layers, as well as across layers.

Such fine-scale heterogeneities can have a profound impact on contaminant plume migration, as shown in Figure 3. In this figure, we show two snapshots in time: one from a simulation based on a model with homogeneous layers (Figure 2, left) and one from a simulation based on a model with heterogeneous layers (Figure 2, right). The dispersion of the contaminant plume is dramatically different in the two cases. In the homogeneous case, we see the effect of the layers: as the plume moves through the subsurface, the portion in the top, more permeable layer begins to separate from the portion in the second, less permeable layer. In the heterogeneous case, however, preferential flow paths result in a much more disperse plume.

Some modeling codes attempt to compensate for the inadequacy of a homogeneous model by incorporating "effective medium" parameters [7, 9], in which an effective hydraulic conductivity value and dispersion coefficient are used to mimic the effects of the heterogeneities. Although this parsimonious approach can be used to approximate gross flow and transport behavior in an idealized medium, and often yields an improved simulation, it typically fails to predict plume travel times correctly. Moreover, this approach still requires that the variability of the hydraulic conductivity (or some other medium property) be measured and characterized. We advocate a complementary approach in which the same measurements and characterization of fine-scale heterogeneities are

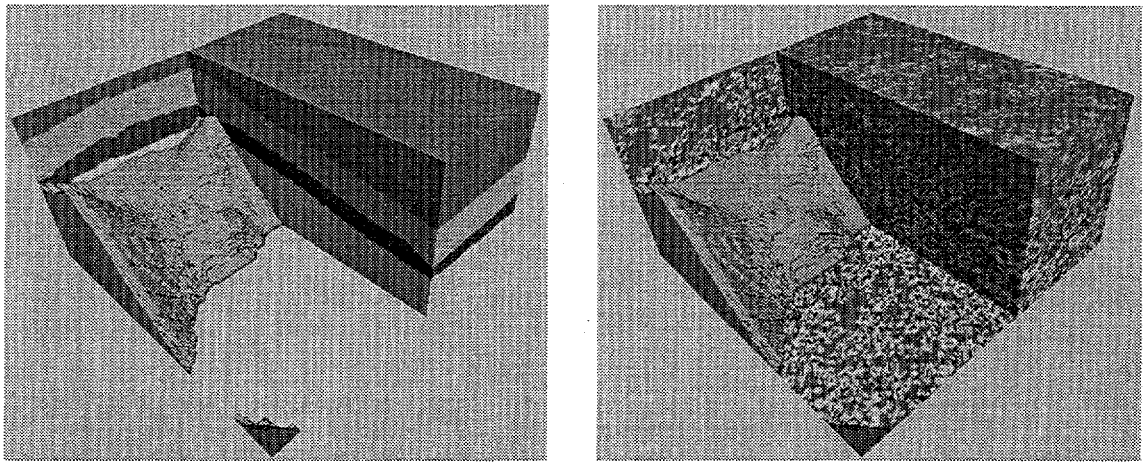


FIG. 2. *Conceptual model of a subsurface (exaggerated ten-fold in the vertical) with a clay layer, fault zone, and five layers. The layers are homogeneous on the left and heterogeneous on the right.*

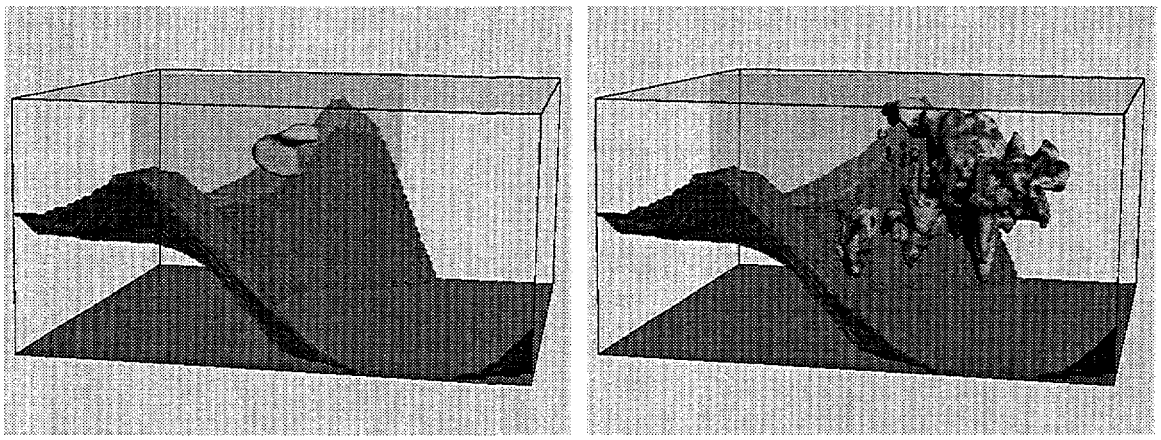


FIG. 3. *Snapshots in time in two simulations: one (on the left) using a model with homogeneous layers, and one (on the right) using a model with heterogeneous layers.*

used within a sophisticated computational framework that is capable of directly resolving property variability. Specifically, we develop a number of equally likely hydraulic conductivity *realizations* that we use in a series of simulations (see §2.1). This approach allows us to simulate the extreme behavior (arising from the fine-scale heterogeneities) that can impact plume migration (and the conclusions one might draw about a particular remediation strategy) without needing to reproduce the exact property values at every point in the computational grid.

1.2 Parallel Computing is Needed

The need to resolve fine-scale heterogeneities necessitates the use of high performance computing technology. Consider, for example, the LLNL site. The sheer physical size of the site (several square kilometers and hundreds of meters deep) and the need to resolve the heterogeneities to within meters leads to huge problems: one might need as many as 10^6–10^9 spatial zones (on a uniform mesh) to model the LLNL site adequately. Although adaptive and/or local mesh refinement might reduce the problem size by an order of magnitude, one is still left with a large problem. Moreover, we wish to run hundreds of time-dependent simulations as we explore different conceptual models, evaluate competing remediation strategies, and seek to optimize a given procedure. Parallel processing makes this possible.

To address the need for a simulation code that exploits the power of massively parallel

processing, we have formed a multidisciplinary team to build a new code for modeling fluid flow and chemical transport through heterogeneous porous media. We have brought together experts in subsurface modeling (geostatistical realizations), advanced computing methods (algorithms and parallel computing), and environmental restoration (expertise in the use of models) to develop the simulator and to apply it to *real* sites. The simulator, which we call ParFlow (for *par*allel *flow*), enables both detailed (i.e., high resolution) *and* large-scale (e.g., regional modeling) simulations of complicated sites.

The ParFlow simulator was built from scratch in C to be portable and scalable across a variety of computing platforms. The primary target architecture is distributed memory MIMD machines with message passing, which allows us to run on workstation clusters and massively parallel computers. Our main workhorse is the 256-node CRAY T3D located at LLNL, but earlier versions of the simulator have been run on a 1024-node nCUBE/2 and a 128-node IBM SP-1. In addition, we can run on shared memory multiprocessors (e.g., SGI Power Challenge). The simulator has even been run on a Pentium PC under Windows 95.

This portability is important for two reasons. First, we wish to support a variety of platforms to maximize the utility of the code (i.e., its marketability). Second, we see the code being used in a scalable fashion, and we wish to support this concept. Specifically, we imagine engineers running relatively small simulations on PCs and workstations to gain an understanding of a site, and then scaling up to the MPP only when detailed simulations are needed. It is critical that the same code (and graphical user interface) be available across the hierarchy of platforms.

Portability, however, is not enough. We also require that ParFlow be scalable, meaning that it can efficiently use additional processors to solve proportionately larger problems. For example, if we can solve a 1M zone problem on 16 processors of an MPP in ten seconds, we want to be able to solve a 16M zone problem on 256 processors in ten seconds as well. As a corollary, we emphasize that one should always seek to use the smallest number of processors necessary to do the job. If too many processors are allocated, scalability is lost, and the resources are inefficiently used.

2 Building a Scalable Subsurface Model

The first step in the simulation process is to construct a conceptual model of the subsurface. Here, one typically works with geologists and hydrologists to identify the "gross" geologic features (called "hydrostratigraphic units," or more simply, "geounits"). For example, in Figure 2, there is a low permeability clay layer, five roughly horizontal layers, and a fault zone (represented by the plane). Each of these geounits is then identified with a specific soil type.

In traditional, PC-based simulation efforts, each soil type might be assigned a single hydraulic conductivity value. In other words, the geounit is assumed to be effectively homogeneous. Moreover, the conceptual model is defined on a computational grid (on which the discrete problem will be solved). That is, the user specifies hydraulic conductivity values for each cell in some computational domain, often with the aid of a kriging program. So long as the computational grid is relatively small, say a few thousand cells, this approach is manageable. However, as we discussed earlier, we may need tens of millions of spatial zones to adequately resolve the fine-scale heterogeneities that can have a profound impact on the subsurface contaminant migration. It is therefore necessary to employ automatic algorithms in the definition of the hydraulic conductivity field. Moreover, since we may wish to run a hierarchy of simulations of varying sizes (on different platforms), it is essential that our conceptual model be scalable.

In ParFlow, the conceptual subsurface model is *independent* of the underlying computational grid. That is, features such as geounits are defined in real-space coordinates, and the resolution of these features is dictated by the hydrologists and geologists developing the model, *not* by the resolution of a computational grid. This gives the modeler the freedom to express the model as it is truly envisioned. For example, a sloping plane is represented as a sloping plane without the stair-stepping effect associated with a discrete (computational) grid.

The representation of these grid-independent features typically requires a small amount of data relative to that required by grid-dependent representations. Furthermore, the model need not be changed each time the computational grid is changed. For example, one way to represent the

sloping plane mentioned above is with two adjacent triangles. Here, one need only store four real-space coordinates (vertices), and six integers that describe which vertices make up the two triangles. This representation is read into PARFLOW, where it is then mapped onto a computational grid to form what we refer to as a *computational model.* In most groundwater codes, the conceptual model coincides with some computational model. This approach has the drawback of requiring large amounts of storage for the problem data, which must be read into the simulator. In addition, the conceptual model must be recreated each time one wishes to employ a new computational grid. (This is often so expensive that modelers continue to use a coarse computational grid even when a finer grid is clearly needed.)

To define our conceptual model, we use the GMS software [20]. GMS allows one to interactively manipulate the geounits on a PC or workstation, and then store the results mathematically. For example, we typically use *triangulated networks* to represent surfaces and layers (similar to the sloping plane example above), and these triangulations are generated via GMS. These triangulated surfaces are written to a file in a compact, mathematical representation, and can then be read directly into PARFLOW to generate the computational model.

This *scalable* approach is extremely valuable, and is one of the strengths of the PARFLOW simulator. One is free to use a PC or workstation to define the conceptual model, and to fine-tune it by hand, without restricting the model to a particular computational grid. The user need only define the conceptual model *once*, and is then free to run a variety of simulations of varying resolution. For example, the engineer might want to run a large number of "what if" scenarios to get a feel for a particular site. These simulations could be run on a workstation using a relatively coarse computational grid. Once the engineer is satisfied with the model and its parameters, the resolution of the simulation can be increased without changing the underlying conceptual model. The engineer simply changes a few input parameters and runs the simulator on a workstation cluster or MPP, depending on the size of the problem and the available resources. In this way, we allow the user to amortize the cost of the conceptualization process—which is the most time-consuming part of any modeling effort.

2.1 Dealing With a Paucity of Data

In our model, soil heterogeneity is reflected in the spatial variation of the hydraulic conductivity K. Unfortunately, property measurements are typically too sparse to specify the kind of detail required in our model. (Recall that 10^6–10^9 mesh points may be needed to properly resolve the subsurface heterogeneities, and one never has this much hard data.) At the LLNL site, for instance, there are over 400 monitoring wells collecting data on the subsurface medium and contaminant plume. Although this data can be supplemented with so-called soft data (obtained, for example, by cross-well tomography), there are still relatively few data points. Consequently, hydrologists and geologists often describe fine-scale heterogeneity with statistical interpolation techniques [10, 11, 19]. Here, measured values and statistical characteristics of a medium property may be used to produce a number of equally-representative *realizations* of the property distribution. These realizations may then be used to study subsurface behavior in a Monte Carlo fashion. Although these realizations cannot give the precise value of the hydraulic conductivity at an (x, y, z) coordinate, they do provide a more realistic representation of property variability between measured data points. This may, in turn, lead to more realistic simulations of flow and transport behavior over large spatial regions. Such simulations can aid site managers in determining where to concentrate their remediation efforts, thereby enabling a more cost-effective site cleanup. Moreover, these simulations can help answer the question: Will a given remediation strategy confine the contamination to the prescribed area?

Let us discuss the realization issue a little further. We wish to devise some way of mathematically representing K in a reasonably realistic manner. To do this, we will (i) interpolate the small number of known measurements and/or (ii) generate hypothetical "property realizations" that retain specific features of the variable properties [10, 11]. For example, one might assume that the variability of K can be described by a spectral random field [17]. Briefly, this means that property values at all points are assumed to be drawn from a "distribution" with a known (or measurable) mean, variance, and spatial correlation. (Correlation is used as a simple measure of the spatial persistence of property values.) An important feature is that the inherent length scales of variation are controlled by the

correlation model and not by the spacing between the actual measurement locations.

This type of model is frequently used to describe hydraulic conductivity distributions for both theoretical and computational investigations [1, 12, 15, 17, 19]. Distributions of $\ln K$ values measured at field sites (including LLNL) often appear to be Gaussian. Estimates for the mean, variance, and correlation scales of this $\ln K$ distribution can be made from preliminary data analyses, and this statistical information can be used to generate one or more property realizations. These realizations will not necessarily match data measured at any specific point, but this can be imposed if desired. When this type of realization is used in computational experiments, one is typically interested in the generic effects induced by the variable nature of the property. For example, one might be interested in modeling the dilution of a given chemical species as it moves distances much greater than the correlation scale.

2.2 Random Fields Via Turning Bands

To obtain spectral random field realizations, we use Tompson's turning bands algorithm [18]. Given certain statistical parameters that describe the hydraulic conductivity (including mean, variance, and spatial correlation), this algorithm generates a realization for the subsurface. That is, it generates a set of points with the specified statistical properties. These points provide the conductivity values needed in the model. See [18] for details.

In our early experiments, we used a serial code to generate the realization and then distributed the conductivity values across the processes. This proved to be too time-consuming. (It took longer to distribute the data than to solve the associated linear system!) Consequently, we implemented a parallel version of the turning bands algorithm within ParFlow. The subsurface realization is now computed in parallel on the proper processes—with no communication. See [5] for details. This is extremely advantageous when we run very large problems, as well as when the code is run in a Monte Carlo fashion to determine the most probable subsurface flow behavior.

We are currently implementing the "sequential Gaussian simulation" (SGS) method for generating a *conditioned* hydraulic conductivity realization. This method produces statistically better random fields, and despite its name, is more fully parallelizable than turning bands.

Once a realization is devised, the resulting model can be validated via history matching. Here, a simulation is run from some previous time to the present, and the results are compared to known data. If the predicted state differs significantly from the observed state, the model can be tuned by varying the statistical parameters and/or other model parameters (e.g., boundary and initial conditions). A more sophisticated approach to model validation and tuning is to embed the simulator in an optimization code (for determining the best model parameters).

3 Numerical Solution Approach

At present, we are focusing our attention on groundwater flow in a confined aquifer. Our model of fluid flow is based on Darcy's Law for single phase flow in a saturated porous medium. We use the following form of the equation,

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

where h is the hydraulic head (for which we wish to solve), $\mathbf{K}$ is hydraulic conductivity (i.e., the mathematical representation of the subsurface), Q is a source term (representing recharge, e.g.). At present, the problem domain is assumed to be a parallelepiped. The boundary conditions may be either Dirichlet (constant head) or Neumann (no flux).

We solve for the hydraulic head on a discrete (logically rectangular) mesh. In the numerical investigations to date, we have used a single uniform grid (with steplengths Δx, Δy, Δz), but in the future we will implement adaptive and/or local mesh refinement (which is needed around pumping wells, for instance). We employ a standard 7-point finite volume spatial discretization, which results in a large system of linear equations. The solution of this linear system (discussed below) yields the fluid pressures, from which the groundwater velocity field is then calculated. The velocity field is then passed to another code to simulate contaminant transport. At present, we have the choice of a particle tracking method with reactive chemistry and a second-order, grid-based Godunov method.

3.1 Multigrid Solution of the Linear System

The dominant cost in computing the fluid flow velocity field is the solution of the linear system of equations, $Ah = q$, resulting from the discretization of the elliptic partial differential equation (1). The coefficient matrix of this linear system is symmetric positive definite with the usual 7-stripe pattern (corresponding to the 7-point discretization stencil and a natural ordering of the unknowns). Since we are interested in detailed simulations (i.e., very large systems), we must use an iterative linear solver; a direct method such as Gaussian elimination would be prohibitively expensive (both in terms of CPU time and storage requirements). Within the hydrology community, the most commonly used iterative methods are SIP and SSOR. Although these methods are adequate for many problems, they suffer in comparison to the preconditioned conjugate gradient method (PCG) [6]. For example, polynomial preconditioned conjugate gradients was shown [13] to be an order of magnitude faster than SIP and SSOR.

The choice of preconditioner is critical to the performance of PCG. Typical preconditioners include incomplete Cholesky and diagonal scaling. Although incomplete Cholesky is extremely effective on serial machines, it is difficult to implement efficiently on parallel machines. Diagonal scaling is often surprisingly effective, but it does not scale, meaning that the number of iterations required for convergence (to some specified tolerance) increases with the problem size (e.g, as the resolution increases). In ParFlow, we have implemented a *multigrid preconditioned conjugate gradient* solver, denoted MGCG. In this algorithm, the preconditioning step is effected via one V-cycle of the multigrid algorithm described in the next paragraph. As we will show, this algorithm (and our implementation of it) are scalable.

Our multigrid algorithm employs standard V-cycles with a choice of damped Jacobi or pointwise red/black Gauss-Seidel smoothing options. (Pre- and post-smoothing is done.) Since the vertical dimensions are usually much smaller than the horizontal dimensions, we typically encounter highly skewed grid cell aspect ratios. We have devised an heuristic semi-coarsening strategy to deal with this grid-based anisotropy. Specifically, we semi-coarsen in the direction with smallest grid spacing (i.e., strongest coupling). In the case of a tie, we coarsen first in x, then in y, and finally in z. Since the K function (i.e, the hydraulic conductivity field) in equation (1) varies by as much as ten orders of magnitude, it is necessary to employ operator-induced prolongation and restriction, as well as an algebraic definition of the coarse grid operator. This approach is extremely effective, especially when used as a preconditioner (rather than as a stand-alone solver). See [3, 4] for a detailed description of this algorithm and its performance on the CRAY T3D massively parallel computer.

3.2 Implementation Issues

The ParFlow simulator was written in C from scratch with scalable parallelism as an underlying design criterion. The problem data (e.g., the hydraulic conductivities) are generated in parallel on a virtual process grid topology consisting of $P = p \times q \times r$ processors. The computations are arranged to avoid explicit data redistribution (which can be expensive).

To achieve portability, we employ a message-passing paradigm. Specifically, we isolate all communications in a layer we call AMPS, which is then layered on top of standard message-passing libraries such as the Reactive Kernel (for homogeneous workstation clusters), PVM (for heterogeneous machine clusters), and SHMEM (for best performance on the CRAY T3D). We also have emulated message-passing on shared memory machines (e.g., the multiprocessor SGI PowerChallenge). This approach allows us to port ParFlow to new machines in as few as two days: only the relatively small AMPS layer needs to be rewritten and the rest of the simulator ports immediately.

One of the key operations in our MGCG linear solver is matrix-vector multiplication. Here we exploit the fact that we are using a standard 7-point discretization on a structured mesh. Specifically, we represent the matrix as a distributed stencil, with the matrix stencil coefficient distributed analogous to the problem data. To effect the matrix-vector multiplication, one first exchanges inter-processor boundary data, and then each processor can independently carry out its part of the global matrix-vector multiplication. On machines with the appropriate hardware, the communication costs can be masked by properly arranging the computations.

TABLE 1
A scalable MGCG algorithm: As we refine the computational grid, the number of MGCG iterations remains nearly constant.

Problem Size			J2CG		MGCG	
n_x	n_y	n_z	iters	time	iters	time
17	17	9	456	1.1	9	0.4
33	33	17	963	5.7	10	0.7
65	65	33	1895	57.1	10	2.1
129	129	65	3706	772.1	11	12.6
257	257	129	7391	1549.5	11	12.9

3.3 Algorithmic Performance

In earlier papers [2, 3, 4, 5], we examined the algorithmic and parallel performance of our MGCG algorithm. In this paper, we will summarize a few of those results, especially with respect to scalability. In these results, we compare MGCG to PCG with 2-step Jacobi preconditioning. The iterations were halted once the relative residual was less than 10^{-9} in the 2-norm.

One of the most attractive features of multigrid is that it *can be* a scalable algorithm, meaning that the number of iterations remains constant as the computational grid is successively refined. Of course, the CPU time still increases since each successive problem is larger, but the time increases linearly with the problem size. As the results in Table 1 show, our MGCG algorithm is scalable.

Specifically, we consider the following experiment, in which we show that the rate of convergence is independent of grid resolution. The physical domain is $\Omega = 1024 \times 1024 \times 25.6$ with no-flow boundary conditions on the top and bottom, and constant hydraulic head of one on the vertical faces. We consider successively refined grids consisting of $n_x \times n_y \times n_z$ spatial zones (i.e., grid spacing of $\Delta_x = 1024.0/(n_x - 1)$, $\Delta_y = 1024.0/(n_y - 1)$, $\Delta_z = 25.6/(n_z - 1)$). The turning bands parameters are $\mu = 4$, $\sigma = 1.5$, $\lambda_x = 128$, $\lambda_y = 128$, and $\lambda_z = 6.4$. The timings in the first four rows are for $P = 2 \times 4 \times 4$ processors of the CRAY T3D; in the last row, we used $P = 4 \times 8 \times 8$ processors.

The number of MGCG iterations required for convergence remains approximately constant (about ten iterations) for problems ranging in size from 2,601 zones to 8,520,321 zones. Moreover, the CPU time grows slowly. In contrast, the number of iterations required by the J2CG algorithm increases exponentially, and the CPU time grows rapidly. For the 8M zone experiment, MGCG is more than two orders of magnitude faster than J2CG. Moreover, we wish to emphasize that J2CG is faster than the solvers used in most groundwater codes today. Finally, in a preview of our next topic (scalability of algorithm implementation), compare the MGCG timings in rows four and five. In row four, we solved a problem with 1M spatial zones on 32 processors in about 13 seconds—the same time required to solve a problem eight times larger on $8 \times 32 = 256$ processors (row five).

Although scalability of the algorithm is important, it also is important that the algorithm's *parallel implementation* be scalable. This means that we must handle communications in an efficient manner. In Figure 4 we demonstrate the scalability of our MGCG implementation on the CRAY T3D. (We have obtained similar results on other MPPs.) In this experiment, we fixed the problem size per processor at $64 \times 64 \times 32$ and enlarged the computational domain (i.e., problem size) by increasing the number of processors. Scaled speedup is defined as M_P/M_1, where M_P is the MFLOPs achieved on P processors. (Note that this definition measures only the scalability of our implementation of the algorithm, and *not* the scalability of the algorithm itself.) Ideally, the curve should be flat and close to 1. Observe the typical early drop off (going from one to two processors) and the nearly flat scalability curve. Scaled efficiency is near 80%. In this experiment, we used Cray's SHMEM primitives to get best AMPS message-passing performance.

In our last set of experiments, we study the effects on MGCG convergence of increasing the degree of heterogeneity in the subsurface model. We use the same test problem as in Table 1 with $N = 129 \times 129 \times 65$ grid cells (i.e., grid spacing of $\Delta = 8 \times 8 \times 0.4$). The turning bands parameters are $\mu = 4$, $\lambda_x = 16$, $\lambda_y = 16$, and $\lambda_z = 0.8$, and we vary σ. The timings are for $P = 2 \times 4 \times 4$ processors of the CRAY T3D. The results of this experiment are given in Table 2. Here, we increase σ from

TABLE 2
Modest dependence on the degree of heterogeneity: As we increase the variability of the hydraulic conductivity field (i.e., the K function) via the turning bands' σ parameter, the number of MGCG iterations grows slowly.

		J2CG		MGCG	
σ	σ_K^2	iters	time	iters	time
0.0	0×10^0	1701	354.4	9	10.4
0.5	6×10^0	3121	650.3	9	10.4
1.0	7×10^1	3388	705.7	9	10.4
1.5	1×10^3	3670	764.6	11	12.5
2.0	4×10^4	4273	889.5	17	18.8
2.5	4×10^6	5259	1094.4	26	28.2

0.0 (homogeneous) to 2.5 (extremely heterogeneous). The second column shows the variation in K, which manifests itself in the condition number of the underlying matrix A. As σ increases, the condition number of A grows rapidly, and the problems becomes increasingly more difficult. Thus, the iteration counts for both J2CG and MGCG increase with σ. However, notice that the MGCG iteration counts increase much more slowly than those for J2CG. (There is no reason to expect a constant number of iterations in this case since the problems are fundamentally different.) Once again, we see that the MGCG solver makes it possible to solve large problems relatively fast.

3.4 Detailed Simulations Using ParFlow

The PARFLOW simulator is being applied to several real sites, including LLNL, as part of its validation process. We have run a variety of simulations using 1M and 8M spatial zones. In addition, we have run a few "LLNL-like" problems with as many as 50M spatial zones. These studies have demonstrated the feasibility of using massively parallel computers and advanced numerical methods to enable more realistic simulations of complicated sites than is possible using traditional codes on conventional machines. This capability makes it possible to predict cleanup times more accurately. More important, engineers can use such simulations to evaluate more reliably competing remediation strategies and, once a procedure has been selected, to manage it in the most cost-effective manner. For example, detailed simulations can be used to determine the optimal pumping configuration in a pump-and-treat remediation scenario. Finally, simulations can be used to perform probabilistic risk assessment via a Monte Carlo approach.

4 Summary and Current Directions

In this paper we have described a scalable approach to modeling groundwater flow on parallel computers. The use of massively parallel processing power is motivated by the need to resolve fine-scale subsurface heterogeneities for large sites; upwards of one billion spatial zones may be required. Of course, in practice, one would run such detailed simulations only after a large number

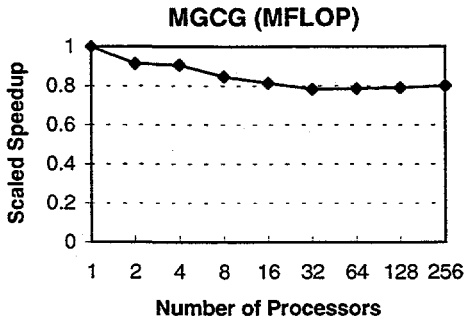


FIG. 4. *Our implementation of the MGCG algorithm scales well on the CRAY T3D.*

of smaller, preliminary simulations. Our simulator supports this mode of operation by running on a variety of computing platforms, ranging from PCs and workstations to large MPPs like the CRAY T3D. Moreover, we advocate the use of *scalable conceptual models*, in which the subsurface model is defined independently of the underlying computational grid.

We also have developed and implemented a scalable algorithm for the solution of the pressure equation. This allows us to make efficient use of additional processors to solve proportionately larger problems—in the same amount of time. Thus, we have devised a fully scalable approach to the simulation of groundwater flow. Both the conceptual model and the numerical solution are scalable to any size computational grid. This means that we can leverage both the engineer's time (spent developing the conceptual model) *and* the computing resources (applied to the numerical simulation). We have successfully employed this approach at the LLNL site, where we have run simulations ranging in size from just a few thousand spatial zones (similar to what PC-based codes would use) to more than 8M spatial zones.

We are commercializing some of this work in collaboration with our industrial partners, International Technology Corporation and Cray Research, Incorporated. As a prelude to commercial use of the simulator, we are conducting verification and validation studies. Specifically, we are comparing ParFlow to other, commonly used codes such as MODFLOW and CFEST. Our preliminary results show that we obtain qualitatively similar results, but we can solve much larger problems much faster. In addition, we are developing a prototype graphical user interface that will facilitate use of the ParFlow simulator. This GUI will, for example, enable an engineer to move easily between the various stages of the simulation process, from model conceptualization (via GMS on a PC) to running the simulation (via ParFlow on a workstation cluster or MPP) to visualizing the results (via AVS on a graphics workstation).

5 Acknowledgements

The ParFlow project is a multidisciplinary collaboration involving computational mathematicians, computer scientists, earth scientists, and environmental engineers. The key project personnel at LLNL are Steven Ashby, Andrew Tompson, Robert Falgout, Steven Smith, Chuck Baldwin, William Bosl, and John Bell. In addition, we are collaborating with Jack Murphy and Melanie Baltezore (International Technology Corporation) and Ilene Carpenter (Cray Research, Inc.). For additional information on this project, see our World Wide Web site (`http://www.nersc.gov/doc/Comp_Research/CCSE/ParFlow`).

This work was supported by Laboratory Directed Research and Development, and by the Mathematical, Information, and Computational Sciences Division of the Office of Energy Research, U.S. Department of Energy, by Lawrence Livermore National Laboratory under contract W-7405-ENG-48.

References

[1] 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 (1989), pp. 549–565.

[2] S. F. Ashby, W. J. Bosl, R. D. Falgout, S. G. Smith, A. F. B. Tompson, and T. J. Williams, *A numerical simulation of groundwater flow and contaminant transport on the CRAY T3D and C90 supercomputers*, in Proc. 34th Cray User Group Conference, Cray User Group, Inc., 1994, pp. 275–282. Held in Tours, France, October 10–14, 1994. Also available as LLNL technical report UCRL-JC-118635.

[3] S. F. Ashby and R. D. Falgout, *A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations*, Tech. Report UCRL-JC-122359, Lawrence Livermore National Laboratory, October 1995. Submitted to *Nuclear Science & Engineering*.

[4] S. F. Ashby, R. D. Falgout, S. G. Smith, and T. W. Fogwell, *Multigrid preconditioned conjugate gradients for the numerical simulation of groundwater flow on the CRAY T3D*, in Proc. ANS International Conference on Mathematics and Computations, Reactor Physics, and Environmental Analyses, 1995. Held in Portland, OR, April 30–May 4 1995. Refereed proceedings.

[5] 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, Society for Industrial and Applied Mathematics, 1995, pp. 131–136. Held in San Francisco, February 15–17, 1995. Also available as LLNL technical report UCRL-JC-118604.

[6] 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 Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309–332.

[7] G. DAGAN, *Flow and Transport in Porous Formations*, Springer Verlag, 1989.

[8] D. E. DOUGHERTY AND R. A. MARRYOTT, *Optimal groundwater management, 1, simulated annealing*, Water Resources Res., 27 (1991), pp. 2493–2508.

[9] L. W. GELHAR, *Stochastic Subsurface Hydrology*, Prentice Hall, 1993.

[10] N. M. JOHNSON AND S. J. DREISS, *Hydrostratigraphic interprestation using indicator geostatistics*, Water Resources Res., 25 (1989), pp. 2501–2510.

[11] A. JOURNEL AND F. ALABERT, *New method for reservoir mapping*, J. Petrol. Tech., (1990), pp. 212–218.

[12] K. MACQUARRIE AND E. SUDICKEY, *Simulation of biodegradable organic contaminants in groundwater, 2. Plume behavior in uniform and random flow fields*, Water Resources Res., 26 (1989), pp. 223–239.

[13] 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 (1989), pp. 1440–1446.

[14] R. L. NEWMARK AND R. D. AINES, *Summary of the LLNL gasoline spill demonstration—dynamic underground stripping project*, Tech. Report UCRL-ID-120416, Lawrence Livermore National Laboratory, 1995.

[15] D. POLMANN, D. MCLAUGHLIN, S. LUIS, L. GELHAR, AND R. ABABOU, *Stochastic modeling for large-scale flow in heterogeneous unsaturated soils*, Water Resources Res., 27 (1991), pp. 1447–1458.

[16] R. T. TAYLOR, M. L. HANNA, N. N. SHAH, D. R. SHONNARD, A. G. DUBA, W. B. DURHAM, K. J. JACKSON, R. B. KNAPP, A. M. WIJESINGHE, J. P. KNEZOVICH, AND M. C. JOVANOVICH, *In situ bioremediation of trichloroethylene-contaminated water by a resting-cell methanotrophic filter*, Hydrological Sciences J., 38 (1993), pp. 323–342.

[17] A. F. B. TOMPSON, *Flow and transport within the saturated zone beneath Lawrence Livermore National Laboratory: Modeling considerations for heterogeneous media*, Tech. Report UCID-21828, Lawrence Livermore National Laboratory, 1990.

[18] 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 (1989), pp. 2227–2243.

[19] A. F. B. TOMPSON AND L. W. GELHAR, *Numerical simulation of solute transport in three-dimensional, randomly heterogeneous porous media*, Water Resources Res., 26 (1990), pp. 2541–2562.

[20] UNITED STATES DEPARTMENT OF DEFENSE, *Groundwater Modeling System Reference Manual*, Engineering Computer Graphics Laboratory, Brigham Young University, Provo, UT, June 1994. Preliminary Draft.