COMPARISON OF SOME PARALLEL KRYLOV SOLVERS FOR LARGE SCALE GROUND-WATER CONTAMINANT TRANSPORT SIMULATIONS

G. Mahinthakumar

Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37831–6203
e–mail: kumar@ornl.gov

F. Saied

Dept of Computer Science, University of Illinois at Urbana–Champaign, Urbana, IL 61801

A. J. Valocchi

Dept of Civil Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801

KEYWORDS

Groundwater Transport, Multiprocessors, Numerical methods, Partial differential equations.

ABSTRACT

Some popular iterative solvers for non–symmetric systems arising from the finite–element discretization of three–dimensional groundwater contaminant transport problem are implemented and compared on distributed memory parallel platforms. This paper attempts to determine which solvers are most suitable for the contaminant transport problem under varied conditions for large scale simulations on distributed parallel platforms. The original parallel implementation was targeted for the 1024 node Intel Paragon platform using explicit message passing with the NX library. This code was then ported to SGI Power Challenge Array, Convex Exemplar, and Origin 2000 machines using an MPI implementation. The performance of these solvers is studied for increasing problem size, roughness of the coefficients, and selected problem scenarios. These conditions affect the properties of the matrix and hence the difficulty level of the solution process. Performance is analyzed in terms of convergence behavior, overall time, parallel efficiency, and scalability. The solvers that are presented are BiCGSTAB, GMRES, ORTHOMIN, and CGS. A simple diagonal preconditioner is used in this parallel implementation for all the methods. Our results indicate that all methods are comparable in performance with BiCGSTAB slightly outperforming the other methods for most problems. We achieved very good scalability in all the methods up to 1024 processors of the Intel Paragon XPS/150. We demonstrate scalability by solving 100 time steps of a 40 million element problem in about 5 minutes using either BiCGSTAB or GMRES.

BACKGROUND

The groundwater contaminant transport problem commonly involves the solution of the advection–dispersion equation (ADE). Some of the common numerical methods employed to solve the ADE include standard finite–elements or finite–differences, mixed method of characteristics, and particle tracking methods. For a single solute undergoing equilibrium adsorption and decay the ADE is given by [Huyakorn and Pinder, 1983; Istok, 1989]

$$R\frac{\partial c}{\partial t} = \nabla \cdot (D \cdot \nabla c) - \nabla \cdot (c\,v) - \lambda R c - \frac{q}{\theta}(c-c_0) \quad (1)$$

where v is the 3x1 velocity field vector, D is the 3x3 dispersion tensor dependent on v, c is the concentration field, R is the retardation factor, and $q(c-c_0)/\theta$ represents the source term with q being the volumetric flux, θ being the medium porosity, and c_0 being the injected concentration (e.g. from injection wells). The velocity field v is usually obtained from the solution of the groundwater flow equation [Mahinthakumar and Saied, 1996]. For saturated flow v is given by

$$\theta\,v = -K\nabla h \quad (2)$$

where h is the computed head field from the groundwater flow equation, K is the 3x3 hydraulic conductivity tensor (usually diagonal), and θ is the porosity. The elements of the 3x3 dispersion tensor D are given by –

$$D_{ij} = a_T|v|\delta_{ij} + (a_L-a_T)\frac{v_iv_j}{|v|} + D_m \quad (3)$$

where a_L and a_T are longitudinal and transverse dispersivities assumed to be constant, D_m is the coefficient of molecular diffusion assumed to be constant (usually very small), and δ_{ij} is the Kronecker delta (if $i=j$, $\delta_{ij}=1$ else $\delta_{ij}=0$). For linear equilibrium adsorption reactions the retardation factor R is given by

$$R = 1 + \frac{\rho K_d}{\theta} \quad (4)$$

where K_d is the adsorption distribution coefficient and ϱ is the bulk density. K_d can be spatially variable for some aquifers.

In recent years lot of attention has been devoted to the numerical solution of ADE especially for advection (or convection) dominated problems [e.g. Noorishad et al., 1992]. As a rule of thumb, for standard finite–element methods the time step size (Δt) and the discretization should be such that $Cr < 1$, $Pe < 2$, or $Cr\cdot Pe < 2$ to obtain stable non–oscillatory solutions [Noorishad et al., 1992; Perrochet and Berod, 1993]. Here the Courant number Cr is defined as $Cr=max\,(v_x\Delta t/dx, v_y\Delta t/dy, v_z\Delta t/dz)$, and the grid Peclet number is defined as $Pe=max\,(dx,dy,dz)/a_L$; where v_x, v_y, v_z and dx, dy, dz are the velocity components and grid spacings in x, y, z directions, and a_L is the longitudinal dispersivity. The upstream weighted formulation is a slight modification of the standard finite element method intended to deal with advection dominated problems (i.e., large Pe's) albeit with some loss of accuracy [Huyakorn and Pinder, 1983; Lapidus and Pinder, 1982]. Even if the Cr and Pe conditions are satisfied, standard finite element methods can still suffer from numerical problems for aquifers with highly heterogeneous K–fields (such as those arising in geosta-

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible
in electronic image products. Images are
produced from the best available original
document.

tistical simulations) where the resulting velocity field obtained numerically can vary strongly from element to element. Higher order finite–element methods, random–walk particle tracking methods [*Tompson*, 1993; *LaBolle et al.*, 1996] and mixed methods [*Neuman*, 1984; *Chiang et al.*, 1989] were devised to deal with some of these difficulties.

Our focus here is not to investigate the robustness and accuracy of the finite–element method, but to investigate different solvers for large scale simulations. A similar analysis was performed by *Peters* [1992] for a simple 2–D system with varying Cr and Pe numbers. We should note here that in some instances of our tests we violated the Cr and Pe conditions resulting in some numerical oscillations. The degree of these oscillations was monitored by the maximum and minimum concentrations in the solution.

KRYLOV SUBSPACE METHODS

Krylov subspace methods for solving a linear system $Ax = b$ are iterative methods that pick the j–th iterate from the following affine subspace

$$x_j \in x_0 + K_j(A, r_0),$$

where x_0 is the initial guess, r_0 the corresponding residual vector and the Krylov subspace $K_j(A, r_0)$ is defined as

$$K_j(A, r_0) \equiv \text{span}\{r_0, Ar_0, \ldots, A^{j-1}r_0\}.$$

These methods are very popular for solving large sparse linear systems because they are powerful and yet offer considerable savings in both computation and storage. In particular, for three–dimensional problems, iterative solvers are often much more efficient than direct (banded or sparse) solvers. Some of the more popular Krylov methods are Preconditioned Conjugate Gradients (PCG), Bi–Conjugate Gradient Stabilized (Bi–CGSTAB), Generalized Minimal Residual (GMRES), Quasi–Minimal Residual (QMR), and Adaptive Chebychev [*Barret et al.*, 1994; *Saad*, 1996]. Of these, PCG is used for only symmetric positive definite systems.

NUMERICAL IMPLEMENTATION

The three–dimensional form of ADE given by equation is descretized using linear hexahedral elements based on the Upstream Weighted Galerkin Formulation [*Huyakorn and Pinder*, 1983; *Huyakorn et al.*, 1985]. The time stepping is implemented using a variable weighted finite–difference scheme where the weight ω can vary from 0 to 1 (ω=0, explicit; ω=0.5, Crank–Nicolson; ω=0.67, Galerkin; ω=1.0 fully implicit). However, in all the tests that were performed in this paper we adopted ω=0.5 which corresponds to the Crank–Nicolson approximation. The upstream weighting factor α is assumed be the same in all three–directions. Although the code has been implemented to handle distorted and non–uniform grids, the tests performed in this paper use only uniform rectangular grids. A comprehensive mass balance checker which checks for mass balances in each time step has been implemented as outlined by *Huyakorn et al.* [1985]. The mass matrix and the zeroth order terms are evaluated us-
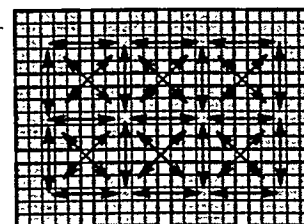
ing a lumped formulation [*Huyakorn*, 1983]. The full matrix is assembled only during the first time step or when the boundary conditions change; only the right hand side is assembled at all other time steps. The previous time step solution is used as initial guess for each time step.

The finite–element approximation of the ADE results in a matrix equation of the form $Ax = b$, where A is a sparse, non–symmetric matrix. For a rectangular grid structure and "natural ordering" of unknowns matrix A has a 27–diagonal banded non–zero structure. In this implementation the non–zero entries of the matrix are stored by diagonals. This enables vectorizing compilers to generate extremely efficient code for operations like a matrix vector product, which are used in Krylov methods.

PARALLELIZATION

Our parallel implementation was originally targeted for the Intel Paragon machines at the Oak Ridge National Laboratory's Center for Computational Sciences (CCS). The code was then ported to the SGI/ Power Challenge Array, SGI/Cray Origin 2000, and Convex Exemplar machines at NCSA (National Center for Supercomputing Applications) using an MPI (Message Passing Interface) implementation. Since most of our performance and scalability tests were performed on the Intel Paragon XPS/150 at CCS, we describe this architecture briefly here (for ease of reference we reproduce some of the information already given in *Mahinthakumar and Saied* [1996]). The XP/S 150 has 1024 MP (multiple thread) nodes connected by a 16 row by 64 column rectangular mesh configuration*. In our implementation we used these nodes only in single threaded mode. In single threaded mode, each node is theoretically capable of 75 Mflops (in double–precision arithmetic). Each node has a local memory of 64 Mb. The native message passing library on the Paragon is called NX.

For the parallel implementation we used a two–dimensional (2–D) domain decomposition in the x and y directions as shown in Fig 1. A 2–D decomposition is generally adequate for groundwater problems because common groundwater aquifer geometries involve a vertical dimension which is much shorter than the other two dimensions. For the finite–element discretization such decomposition involves communication with at most 8 neighboring processors. We note here that a 3–D decomposition in this case will require communication with up to 26 neighboring processors.

☐ overlapping processor regions   ☒ individual processor regions
(arrows show communication pattern)

Fig 1: Plan View of Two–Dimensional Domain Decomposition
(showing a 4x3 processor decomposition)

We overlap one layer of processor boundary elements in our decomposition to avoid additional communication during the assembly stage at the expense of some duplication in element computations. There is no overlap in node points. In order to preserve the 27–diagonal band structure within each processor submatrix, we perform a local numbering of the nodes for each processor subdomain. This resulted in non–contiguous rows being allocated to each processor in the global sense. For local computations each processor is responsible only for its portion of the rows which are locally contiguous. However, such numbering gives rise to some difficulties during explicit communication and I/O stages. For example, in explicit message passing, non–contiguous array segments had to be gathered into temporary buffers prior to sending. These are then unpacked by the receiving processor. This buffering contributes somewhat to the communication overhead. When the solution output is written to a file we had to make sure that the proper order is preserved in the global sense. This required non–contiguous writes to a file resulting in I/O performance degradation particularly when a large number of processors were involved.

All explicit communications between neighboring processors were performed using asynchronous NX or MPI calls. System calls were used for global communication operations such as those used in dot products. The codes are written in FORTRAN 77 using double–precision arithmetic.

MODEL PROBLEMS

Two different model problems were setup for the tests. The first one involves a single extraction well in the center of a square domain extracting a uniformly distributed cylindrical plume (Fig 2). This simple problem was chosen since the velocity field for this problem is analytically known and therefore eliminates the need for a flow solution. This scenario is justified for test runs which look at scalability, parallel performance, and the performance of each solver when the problem size is increased.
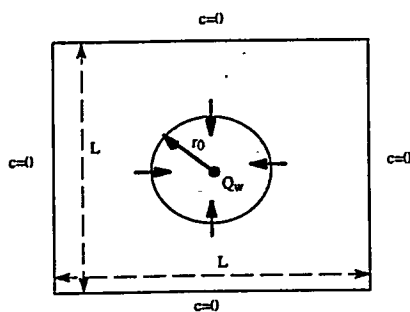
Fig 2: Plan view of Model Problem 1
(for scalability and performance tests)

The velocity field for Model Problem 1 is analytically obtained from the simple expression $v = Q_w/(2\pi rd)$. Here Q_w is the pumping rate, r is the radius from the center of the well and d is the constant vertical depth. This solution assumes infinite boundaries. The pumping rate Q_w and the initial radius of the cylindrical plume r_0 are set as described in individual test cases.

For tests investigating the efficiency of each solver when the roughness of the coefficients is increased (i.e., increase in the spatial variability of the velocity field and reaction coefficients) we chose a different model problem (Model Problem 2) shown in Fig 3. This setup corresponds to a contamination scenario where the contaminant leaches from a single rectangular source (80 x 80) into a naturally flowing groundwater aquifer. The dimensions of the aquifer are fixed at 1600 x 800 x 20 with a uniform rectangular grid of size 2 x 2 x 1. Therefore the problem size is 801x401x21 which results in a matrix of size nn = 6.75 million.
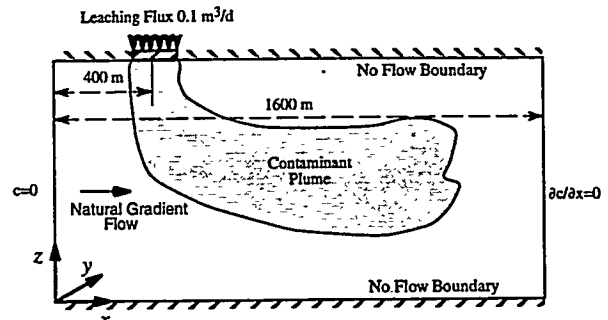
Fig 3: Vertical Cross–Section of Model Problem 2
(for convergence testing with velocity field variability)

The velocity field for Model Problem 2 is generated from the solution of the steady state groundwater flow problem [*Mahinthakumar and Saied*, 1996]. Boundary conditions for this problem are as follows: zero Dirichlet boundary at upstream end ($x=0$) and free exit boundary ($\partial c/\partial x=0$) at downstream end ($x=1600$) vertical boundary faces; constant Dirichlet concentrations of $c=100$ at the rectangular patch and $c=0$ elsewhere on the top horizontal boundary face; no flow boundaries elsewhere. Initial condition is $c=100$ at the top rectangular patch and $c=0$ elsewhere. For tests involving heterogeneous K–fields or K_d–fields (i.e. rough coefficients), we obtained the spatially correlated random fields by using a parallelized version of the turning bands code [*Tompson et al*, 1989]. The degree of heterogeneity is measured by the parameter σ, which is an input parameter to the turning bands code.

PERFORMANCE RESULTS AND DISCUSSION

In this section we present and compare the performance of our implementations with respect to problem size, scalability, and roughness of coefficients. The following selections and notations were used for all performance tests unless otherwise stated:

- convergence tolerance = 1.e–10 (two–norm of relative residual)

- restart parameter for GMRES = 10, ORTHOMIN = 5 (reason: ORTHOMIN takes up twice the memory as GMRES for the storage of restart vectors)

- upstream weighting factor $\alpha = 0.5$, bulk density $\varrho = 1.0$, porosity $\theta = 0.3$, decay coefficient $\lambda = 0.005$, longitudinal and transverse dispersivities $\alpha_L=4.0$ and $\alpha_T=0.05$, adsorption distribution coefficient $K_d = 1.0$.

- timings were obtained by the dclock() (when using NX on XPS/150) or MPI_wtime() (when using MPI on other systems) system calls. Timings reported are for the processor that takes the maximum time.

- For all the tests the following parameter values are noted alongside tables or figures: nn = size of matrix or number of unknowns ($= nx \cdot ny \cdot nz$), np = number of parallel processors, nt = total number of time steps for which the results are reported.

Increasing Problem Size

This test was performed solely to test the performance of each solver when the problem size is increased. Model Problem 1 was used in this test with r_0=L/4, and Q_w=L/10 (set arbitrarily). The problem size was increased from L=60 to L=1920. The vertical z dimension was fixed at 10. Grid spacing was fixed in all three directions with dx=dy=dz=1m. i.e. The problem size was increased from 60 x 60 x 10 for the 1 processor case to 1920 x 1920 x 10 for the 1024 processor case. As the problem size doubled in both x and y directions, the 2–D processor configuration was also changed accordingly (i.e., problem size 60 x 60 x 10 corresponds to 1 x 1 processor configuration, 120 x 120 x 10 to 2 x 2, ..., and 1920 x 1920 x 10 to 32 x 32). In Table 1 we present the total iteration counts and solution timings for the four Krylov methods as the problem size increases. The timings and the iteration counts are for the first 100 time steps of the simulation.

nn	np	BiCGSTAB		GMRES(10)		OMIN(5)		CGS	
		iter	time	iter	time	Iter	time	iter	time
40K	1	1237	568	2464	681	2349	741	1505	679
160K	4	1516	582	3284	926	3068	978	2097	940
640K	16	1645	766	3348	965	3105	1013	2442	1103
2.5M	64	1109	538	2043	613	1933	664	1816	851
10M	256	632	323	1073	329	1062	387	969	472
40M	1024	538	305	859	286	856	346	932	497

Table 1: Effect of increasing problem size for Model Problem 1 (total iterations and time in seconds for 100 time steps)

From Table 1 it is apparent that the convergence behavior does not deteriorate with the increase in problem size for all the methods. The iterations increase slightly in the beginning and then decrease rapidly as the problem size increases. We attribute this to the fact that the problem actually becomes easier for larger problems since $\partial v/\partial r$ at the front ($r \leq r_0$) decreases.

Scalability Test

The scalability test is performed by increasing the problem size accordingly with the processor count. i.e. nn/np is kept approximately constant. Model Problem 1 is used with r_0=20 and Q_w=10 for all the problems. This is similar to the test performed in the previous section except for the values of r_0 and Q_w which are kept fixed here. When these are fixed we found that the total iteration count for each method

remained approximately constant as the problem size was increased thus providing a good test for scalability. The results are shown in Fig 4. It is evident from Fig 4, that all methods have similar scalability behavior. This is mainly because the solution times for all the methods are dominated by the matvec times (about 82% for BiCGSTAB and CGS, and 71% for GMRES and ORTHOMIN) and they all use the same matvec routine.
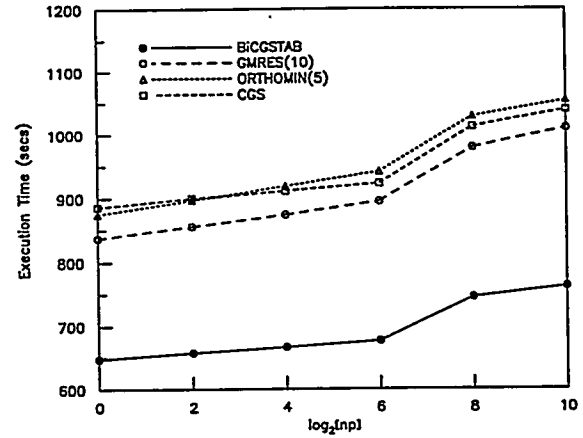

Fig 4: Comparison of Scalability of Each Method (nn/np maintained constant, times for 100 time steps)

Convergence Behavior

Convergence behavior for each method is shown in Fig 5 for Model Problem 2 with $\sigma = 4.0$ (see Table 2). Note that the horizontal axis denotes the CPU time and not the iteration count. Except CGS, all the methods seem to exhibit a smooth convergence behavior for this problem. We should note here that the Courant number condition was violated in this test case giving oscillatory solutions.
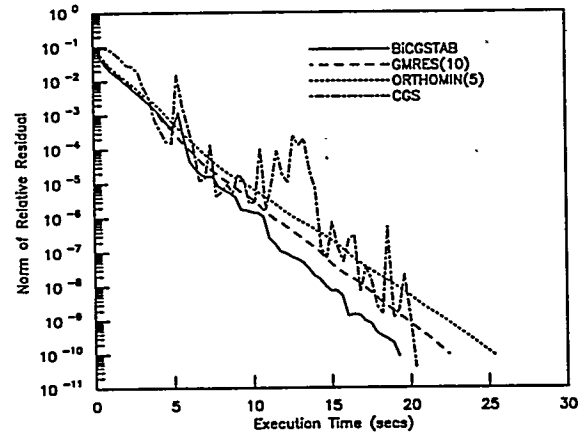

Fig 5: Convergence behavior for Model Problem 2 ($np = 242$, first time step corresponding to $\sigma = 4.0$ in Table 2)

Parallel Performance for Fixed Problem Size

We measured the parallel performance of each solver for a fixed problem of size 481 x 481 x 11 (2.5 M nodes) by increasing the number of parallel processors from 64 to 1024. The results are shown in Fig 6.
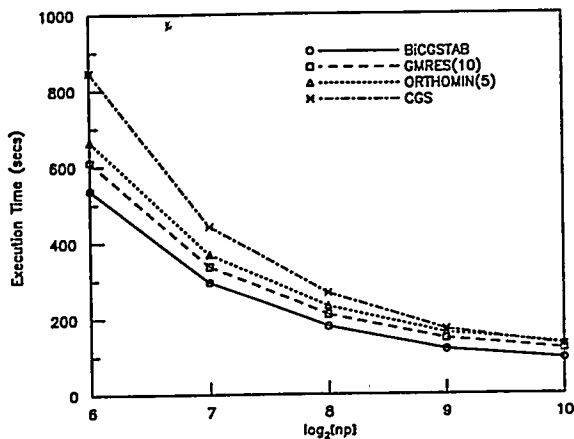
Fig 6: Speed up behavior for fixed problem size
($nt = 100$, $nn = 2.5M$ problem in Table 1)

From Fig 4 we see that all methods have very similar speed up behavior for the case tested here. BiCGSTAB and CGS seem to speed up slightly better than either GMRES or ORTHOMIN.

Comparison with other machines

In Fig 7 we compare the performance of BiCGSTAB on various machines. Timings are reported for model problem 1 with size 241 x 241 x 11 using 16 processors. This problem is the same problem reported in row 3 of Table 1. The total time includes initial setup, finite-element matrix assembly, matrix solution and I/O.
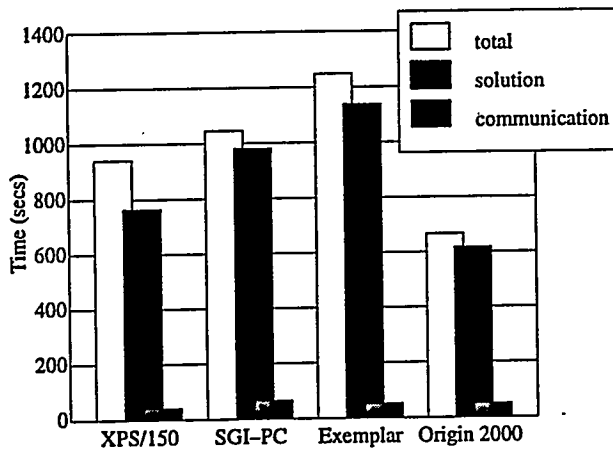

Fig 7: Performance of BiCGSTAB on four parallel platforms
($np = 16$, $nn = 640K$, $nt = 100$)

From Fig 7 we can observe that the SGI/Cray Origin 2000 gives the best overall performance. However, considering that each processor of the Origin 2000 is at least 3 times more powerful than the Intel Paragon XPS/150, this performance is not that good. We should note here that no additional effort was expended in optimizing the code for architectures other than the Intel Paragon. From this figure we can also observe that the matrix solution time will dominate the total time as long as sufficient time steps are taken (in this case 100 time steps). The

explicit communication time is insignificant for all the machines for the 16 processor case.

Roughness of Coefficients

The efficiency of each solver as we increase the variability of the K–field (denoted by parameter σ) is shown in Table 2. Model Problem 2 (problem size 801x401x21) is used in this study with a fixed time step size of 100 days for all cases using a 22x11 processor configuration ($np = 242$). The timings reported are for 10 time steps. $\sigma=0$ corresponds to a homogeneous K–field and $\sigma = 4.0$ corresponds to an extremely heterogeneous K–field with more than 4 orders of magnitude difference in some adjacent cell K-values. As we increased σ from 0 to 4, the maximum Courant number also increased from 0.8 to 500. The case with $\sigma = 4.0$ did not produce an acceptable solution exhibiting very high numerical oscillations. We attribute these oscillations not simply to the violation of the Cr condition but also due to the discontinuities in the velocity field. We note here that additional runs performed for this case with a much smaller time step (0.1 day instead of 100 days) also exhibited some oscillatory behavior even though the convergence of the Krylov solvers greatly improved.

σ	BiCGSTAB		GMRES(10)		OMIN(5)		CGS	
	Iter	Time	Iter	Time	Iter	Time	Iter	Time
0.0	109	40.2	195	47.5	194	51.4	113	41.1
1.0	257	91.6	454	109.1	457	119.8	298	103.7
2.0	356	125.8	648	155.1	649	165.3	479	164.8
3.0	449	157.7	807	189.7	813	206.1	664	227.4
4.0	515	180.8	861	219.5	985	249.1	786	268.9

Table 2: Effect of K–field Heterogeneity
($nn = 6.75M$, $np = 242$, $nt = 10$)

It is evident from the above table that all methods showed difficulty in convergence as σ is increased. Although BiCGSTAB performed slightly better than the other methods the differences are within a factor of 2 of each other.

We also performed some tests with variable K_d–fields (in equation (4)) generated in a similar fashion with σ ranging from 0 to 4. The convergence behavior of all the solvers in this case did not change appreciably with σ. This behavior can be attributed to the fact that any variations in K_d would simply add randomly variable positive values to the diagonal entries of the matrix thus adding no additional difficulty to the linear system solution.

Floating Point Performance

The floating point performance of all the methods were in the range of 10–12 Mflops per processor on the XPS/150. For the largest problem we obtained performances close to 10 Gflops on 1024 processors.

CONCLUSIONS

Our results indicate that all the solvers perform reasonably well for most of the test problems. i.e., The overall performance is within a

factor of 2 of each other. Within these close limits the performance is in the following decreasing order for most problems: BiCGSTAB, GMRES(10), ORTHOMIN(5), and CGS. All the methods exhibit very good scalability up to 1024 processors. This is demonstrated by the fact that we are able to solve 100 time steps of a 40 million element problem in around 300 seconds using either BiCGSTAB or GMRES(10) (see Table 1). For all the methods tested, convergence behavior is not sensitive to the problem size. This result is different compared to the diagonal preconditioned conjugate gradient solver (DPCG) used in the steady state groundwater flow problem [*Mahinthakumar and Saied*, 1996] where the convergence behavior deteriorated with increasing problem size. Since the number of iterations per time step is also generally small for all the methods (compared to DPCG in the steady state flow problem), we do not see any major advantage in using a method such as multigrid for this non–symmetric transport problem [see *Mahinthakumar and Saied*, 1996]. This result can be attributed to the fact that a steady state problem is generally more difficult to solve in the linear algebra sense than a transient problem. Convergence behavior of all the methods deteriorated when the variability of the velocity field was increased or when the Courant number was increased. However, these conditions affected the accuracy and the oscillatory behavior of the solution more than the convergence behavior of each solver.

ACKNOWLEDGEMENTS

REFERENCES

Barret, R., M. Berry, T. 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, *SIAM Publication*, 1994.

Chiang C.Y., M. F. Wheeler, P. B. Bedient, A modified method of charactersitics technique and mixed finite elements method for simulation of groundwater solute transport, *Water Resour. Res.*, 25(7), 1541–1549, 1989.

Huyakorn, P.S., and G.F. Pinder, *Computational Methods in Subsurface Flow*, Academic Press, New York, N.Y., 1983.

Huyakorn, P. S., J. W. Mercer, and D. S. Ward, Finite element matrix and mass balance computational schemes for transport in vari-

ably saturated porous media, *Water Resour. Res.*, 21(3), 346–358, 1985.

Istok, J.D., *Groundwater modeling by the finite element method*. Water Resources Monograph 13, American Geophysical Union, Washington, D.C., 1989.

LaBolle, E. M., G. E. Fogg, and A. F. B. Tompson, Random–walk simulation of transport in heterogeneous porous media: Local mass–conservation problem and implementation methods, *Water Resour. Res.*, 32(3), 583–593, 1996.

Lapidus L., and G. F. Pinder, Numerical solution of partial difference equations in science and engineering, John Wiley & Sons, 1982.

Mahinthakumar, G., and F. Saied. 1996. "Distributed memory implementation of multigrid methods for groundwater flow problems with rough coefficients", Editor: A. M. Tentner, High Performance Computing '96, "Grand Challenges in Computer SImulation", In *Proceedings of the 1996 Simulation Multiconference*, (New Orleans, LA, Apr. 8–11), 52–57.

Neuman, S.P., Adaptive Eulerian–Langrangian finite element method for advection–dispersion, *Int. j. numer. methods eng.*, 20, 321–337, 1984.

Noorishad, J., C. F. Tsang, P. Perrochet, and A. Musy, A perspective on the numerical solution of convection–dominated transport problems: A price to pay for the easy way out, *Water Resour. Res.*, 28(2), 551–561, 1992.

Perrochet, P., and D. Berod, Stability of the standard Crank–Nicolson–Galerkin applied to the diffusion convection equation: Some new insights, *Water Resour. Res.*, 29(9), 3291–3297, 1993.

Peters, A., CG–like algorithms for linear systems stemming from the FE discretization of the advection–dispersion equation, Numerical Methods in Water Resources, Vol. 1, 511–518, Elservier Applied Science, 1992.

Saad, Y., Iterative methods for sparse linear systems, PWS publishing company, Boston, MA, 1996.

Tompson, A.F.B., Numerical simulation of chemical migration in physically and chemically heterogeneous porous media, *Water Resour. Res.*, 29(11), 3709–3726, 1993.

Tompson, A.F.B., R. Aboubu, and L.W. Gelhar. 1989. "Implementation of the three–dimensional turning bands random field generator." *Water Resources Research*. vol. 25, no. 10 (Oct.): 2227–2243.