Presented at the TOUGH Workshop 95
Lawrence Berkeley Laboratory, Berkeley, CA 94720, March 20-22, 1995

LBL-37066

A NEW SET OF DIRECT AND ITERATIVE SOLVERS FOR THE TOUGH2 FAMILY OF CODES

George J. Moridis

Earth Sciences Division, Lawrence Berkeley Laboratory
University of California, Berkeley, CA 94720

ABSTRACT

Two new solvers are discussed. LUBAND, the first routine is a direct solver for banded systems and is based on a LU decomposition with partial pivoting and row interchange. BCGSTB, the second routine, is a Preconditioned Conjugate Gradient (PCG) solver with improved speed and convergence characteristics. Bandwidth minimization and gridblock ordering schemes are also introduced into TOUGH2 to improve speed and accuracy.

Introduction

Most of the computational work in the numerical simulations of fluid and heat flows in permeable media arises from the solution of large systems of linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a banded matrix of order N, $\mathbf{x}$ is the vector of the unknowns, and $\mathbf{b}$ the right-hand side. These are solved using either direct or iterative methods. The most reliable (and often the simplest) solvers are based on direct methods. The robustness of direct solvers comes at the expense of large storage requirements and execution times. Iterative techniques exhibit problem-specific performance and lack the generality, predictability and reliability of direct solvers. These disadvantages are outweighed by their low computer memory requirements and their substantial speed especially in the solution of very large matrices.

In TOUGH2 the matrix $\mathbf{A}$ is a Jacobian with certain consistent characteristics. In systems with regular geometry, $\mathbf{A}$ has a known block structure with well defined sparsity patterns. In general, $\mathbf{A}$ matrices arising in TOUGH2 simulations are non-symmetric with typically no diagonal dominance. Although $\mathbf{A}$ can be positive definite in regular systems with homogeneous property distributions, it usually is not, and ill-conditioning is expected in realistic heterogeneous large systems. Due to the fact that $\mathbf{A}$ is a Jacobian, the elements of $\mathbf{A}$ in a single row may vary by several orders of magnitude. In TOUGH2 simulations it is possible to encounter a large number of zeros on the main diagonal of $\mathbf{A}$, making central $S = NB$ pivoting impossible and resulting in very ill-conditioned matrices. TOUGH2 creates matrices which are among the most challenging, with all the features that cause most iterative techniques to fail. In addition, the general-purpose nature of TOUGH2 means that different matrix characteristics may arise for different types of problems. This explains the past heavy reliance of TOUGH2 on the direct solver MA28 [*Duff*, 1977].

The LUBAND Solver

LUBAND is a direct solver intended to replace the MA28 solver currently used in the TOUGH2 family of codes. It is derived from routines in the LAPACK [1993] package, which have been enhanced and extensively modified to conform to the TOUGH2 architecture. It is based on a LU decomposition with partial pivoting and row interchange, and allows the solution of systems with a large number of zeros on the main diagonal. Unlike MA28 (which is a general solver), LUBAND is a banded matrix solver, and as such it capitalizes on the significantly lower and well defined memory requirements of these solvers. A pseudo-banded matrix structure is automatically created by LUBAND for systems with irregular grids.

Although LUBAND can be applied without any problem in the current TOUGH2 version, a new MESHMAKER routine was also developed to minimize the bandwidth of matrix $\mathbf{A}$ and maximize the benefits of LUBAND. This was prompted by the heavy penalty which non-

optimization of the bandwidth exacts. Defining *work* W as the number of multiplications and divisions necessary to convert the full matrix to an upper triangular form and to perform back substitution, *Price and Coats* [1974] showed that for direct solvers

$$W \propto NM^2 \qquad \text{and} \qquad S = NB \tag{1}$$

and S is the minimum storage requirement, N is the order of the matrix and M its half-bandwidth, the full bandwidth B being

$$B = 2M + 1. \tag{2}$$

The form of the matrix depends upon the ordering of equations. For a given problem size N, work and storage are minimized when M is minimized. If I, J, K are the number of subdivisions in the x-, y- and z-directions respectively, the shortest half-bandwidth is $M = JK$ when $I > J > K$. This is called *standard* ordering [*Aziz and Settari,* 1979], and the resulting matrices are banded. As W increases with the square of M, it is obvious that the penalty for non-optimization of the ordering of equations may be substantial. Note that it is possible to use the new MESHMAKER with MA28.

A further substantial improvement was added to the new MESHMAKER, which features as an option the implementation of the Alternating Diagonal Scheme (D4) for gridblock ordering. D4 is a direct solution technique belonging to the matrix-banding class, which derives its benefits from the numbering of the grid points. More details can be found in *Price and Coats* [1974]. D4 ordering partitions the matrix into four distinct entities. This structure allows forward elimination through the equations in the lower half of $\mathbf{A}$, which zeroes all original entries in the lower left quadrant of $\mathbf{A}$ and transforms it into a null matrix, while creating non-zero entries in the submatrix $\mathbf{A_{LR}}$ in the lower right quadrant of $\mathbf{A}$. The submatrix $\mathbf{A_{LR}}$ is of order *N/2*, and allows the calculation of the lower half of $\mathbf{x}$, from which the upper half is obtained by simple substitution. Depending on the grid geometry, D4 makes possible execution speed improvement by a factor ranging between a minimum of 2 and a maximum of 5.85 [*Price and Coats,* 1974] over standard ordering. Moreover, it reduces storage requirements by a factor of 2.

LUBAND makes possible the solution of large multi-dimensional problems. The maximum benefits of LUBAND are realized when used within the context of D4, i.e. to solve the submatrix $\mathbf{A_{LR}}$. However, D4 can only be used with regular grids. Although it is theoretically possible to solve $\mathbf{A_{LR}}$ using MA28, the user is strongly advised against for two reasons: the uncertainty over the storage requirements of MA28 and the known rapid deterioration of the MA28 performance as the matrix fill-in increases (e.g. in 3-D problems). The user has also the option of solving $\mathbf{A_{LR}}$ using the package of Preconditioned Conjugate Gradient (PCG) solvers available in TOUGH2 [*Moridis and Pruess*, 1995].

The BCGSTB Solver

BCGSTB, the second solver, belongs to the PCG family, and complements T2CG1 [*Moridis and Pruess,* 1995], the existing suite of iterative solvers in TOUGH2. It was developed based on the BiCGSTAB(ℓ) algorithm [*Sleijpen and Fokkema*, 1993], which is a recent extension of the more traditional BiCGSTAB algorithm of *van der Vorst* [1992]. It was developed to address the problem of irregular convergence behavior of the PCG solvers in T2CG1 in situations where the iterations are started close to the solution (e.g. when approaching steady state). This is a weakness which afflicts most PCG solvers, and may lead to severe residual cancellation and errors in the solution. BiCGSTAB(ℓ) alleviates the irregular (oscillatory) convergence common to the Bi-Conjugate Gradient (Bi-CG) [*Fletcher*, 1976] and Conjugate Gradient Squared (CGS) [*Sonneveld*, 1989] methods in T2CG1, thus improving the speed of

convergence. Finally, it alleviates potential stagnation or even breakdown problems which may be encountered in traditional BiCGSTAB. According to *Sleijpen and Fokkema* [1993], BiCGSTAB(ℓ) combines the speed of Bi-CG with the monotonic residual reduction in the Generalized Minimum Residual (GMRES) method, while being faster than both. Theoretical analysis of the underlying concepts also indicates that the BiCGSTAB(ℓ) algorithm is especially well-suited to the solution of very large (i.e. N > 50 000) problems [*van der Vorst* , 1992].

BCGSTB uses the Boeing-Harwell matrix storage scheme of TOUGH2, and has the same architecture as the other routines in T2CG1. It uses a modified LU decomposition for preconditioning, as well as ILU and MILU preconditioners with various levels of fill. Its memory requirements increase linearly with the order ℓ of the Minimal Residual polynomial. For ℓ = 4, it requires twice the memory of Bi-CG or CGS, which is half the GMRES requirement.

Examples

The solvers were tested in four test problems. Test problem 1 involves a laboratory convection cell experiment. A porous medium consisting of glass beads fills the annular region between the two vertical concentric cylinders. Application of heat generates a thermal buoyancy force, giving rise to the development of convection cells. This problem has been discussed in detail by *Moridis and Pruess* [1992]. The EOS1 module is used. The domain consists of 16x26 = 416 gridblocks in (r,z), with NK = 1 and NEQ = 2, resulting in a total of N = 832 equations.

Test problem 2 examines flow as it reaches steady state in a simple two-dimensional model of a heterogeneous porous medium. The basic computational grid is composed of 80 × 120 = 9600 grid blocks in (x,y). Impermeable obstacles with lengths uniformly distributed in the range of *2 - 4 m* are placed in the domain. These blocks are removed from the mesh, leaving a total of 8003 grid blocks. EOS1 is used, and NK = NEQ = 1. *Moridis and Pruess* [1995] present a thorough discussion of the problem.

Test problem 3 describes the WIPP (**W**aste **I**solation **P**ilot **P**lant) repository, which is planned for the disposal of transuranic wastes. It is located in a bedded salt formation, is brine saturated and consists of a large number of beds with thin interbeds. The layer permeabilities vary by four or five orders of magnitude. The purpose of the model is to evaluate effects of gas generation and two-phase flow on repository performance within a complex stratigraphy. The simulated domain consists of 1200 elements in a 2-D vertical grid. EOS3 is used in this isothermal (NK = NEQ = 2) problem, which results in a total of N = 2400 equations. More details can be found in *Moridis and Pruess* [1995].

Test problem 4 describes the TEVES (**T**hermal **E**nhanced **V**apor **E**xtraction **S**ystem) process, which is designed to extract solvents and chemicals contained in the Chemical Waste Landfill at Sandia National Laboratories. In this process the ground is electrically heated, and boreholes at the center of the heated zone are maintained at a vacuum to draw air and vaporized contaminants into the borehole and to a subsequent treatment facility. The 3-D grid consists of 1300 gridblocks. EOS3 is used (NK = 2, NEQ = 3), and N = 3900 equations are solved. Additional information can be found in *Moridis and Pruess* [1995].

Results and Conclusions

The results are presented in Tables 1 through 4 and Figures 1 and 2. In all the simulations a modified LDU preconditioner was used with no fill-up of the resulting LU preconditioned matrices. The following conclusions can be drawn:

(1) The LUBAND routine is a fast and efficient solver with modest memory requirements, and can solve large problems previously untractable with the MA28 solver.

(2) Bandwidth minimization significantly improves the LUBAND performance.

(3) Coupling D4 with LUBAND increases the solver speed by at least a factor of 2, and makes it competitive with iterative solvers in small and medium-sized matrices. However, the speed advantage of the PCG solvers becomes apparent in three-dimensional problems.

(4) BCGSTB is very competitive, and outperforms the T2CG1 iterative solvers in almost all cases. There seems to be no measurable difference in performance for $\ell = 2$ and $\ell = 4$.
(5) BCGSTB does not exhibit oscillatory behavior, and does not suffer from instability as it approaches the steady-state condition.

Acknowledgement

This work was supported by the Director, Office of Civilian Radioactive Waste Managenent, Yucca Mountain Site Characterization Project Office, under U.S. Department of Energy contract No. DE-AC03-76SF00098. Drs. Curt Oldenburg and Stefan Finsterle are thanked for their helpful review comments.

References

Aziz, K. and A. Settari (1979), Petroleum Reservoir Simulation, Elsevier, London and New York.
Duff, I. S. (1977) MA28 - A set of Fortran Subroutines for Sparse Unsymmetric Linear Equations, AERE Harwell Report R 8730.
Fletcher, R. (1976) Conjugate gradient methods for indefinite systems. Numerical Analysis, Lecture Notes in Mathematics 506, Springer-Verlag, New York.
LAPACK (1993), Univ. of Tennessee, Univ. of California at Berkeley, NAG ltd., Courant Institute, Argonne National Lab., and Rice University, Version 1.1.
Moridis, G.J., and K. Pruess (1992), TOUGH simulations of Updegraff's set of fluid and heat flow problem, Lawrence Berkeley Laboratory report LBL-32611.
Moridis, G.J., and K. Pruess (1995), T2CG1: A package of preconditioned conjugate gradient solvers for the TOUGH2 family of codes, Lawrence Berkeley Laboratory report LBL-36235.
Price, H. S., and K. H. Coats (1974), Direct methods in reservoir simulation, Trans. SPE of AIME (SPEJ), 257, 295-308.
Sleijpen, G.L.G., and D. Fokkema (1993), BICGSTAB(l) for linear equations involving unsymmetric matrices with complex spectrum, Electronic Transcations on Numerical Analysis, 1, 11-32.
Sonneveld, P. (1989) CGS, A fast Lanczos-type solver for nonsymmetric linear systems. SIAM J. Sci. Stat. Comput., 10(1), 36-52.
van der Vorst, H.A. (1992), Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG in the presence of rounding errors, SIAM J. Sci. Statist. Comput., 13, 631-644.

Table 1. Solver Performance in Test Problem 1

Number of Equations: 832 (2-D)			Apple Macintosh QUADRA 800		
SOLVER	Number of Δt's	Newtonian Iterations	Maximum N_{CG}	Minimum N_{CG}	CPU Time (sec)
MA28	26	91	-	-	331
DSLUBC	26	91	31	2	321
DSLUCS	26	91	35	2	295
DSLUGM	26	91	41	11	299
LUBAND	26	91	-	-	246
LUBAND/D4	26	91	-	-	98
BCGSTB(2)	26	91	36	4	301
BCGSTB(4)	26	91	32	8	300

Table 2. Solver Performance in Test Problem 2					
Number of Equations: 8,003 (2-D)			IBM RS/6000 370		
SOLVER	Number of Δt's	Newtonian Iterations	Maximum N_{CG}	Minimum N_{CG}	CPU Time (sec)
MA28	Could not solve the problem due to insufficient memory.				
DSLUBC	10	17	225	172	158
DSLUCS	10	17	173	154	86
DSLUGM	10	19	316	810	379
LUBAND	10	17	-	-	55
LUBAND/D4	10	17	-	-	26
BCGSTB(2)	10	17	64	16	40
BCGSTB(4)	10	17	64	24	33

Table 3. Solver Performance in Test Problem 3					
Number of Equations: 2400 (2-D)			IBM RS/6000 370		
SOLVER	Number of Δt's	Newtonian Iterations	Maximum N_{CG}	Minimum N_{CG}	CPU Time (sec)
MA28	97	521	-	-	450
DSLUBC	102	554	45	9	593
DSLUCS	110	594	45	6	408
DSLUGM	125	673	179	8	478
LUBAND	97	521	-	-	946
LUBAND(opt)	97	521	-	-	376
LUBAND/D4	97	521	-	-	168
BCGSTB(2)	106	531	40	4	388
BCGSTB(4)	104	529	40	8	399

Table 4. Solver Performance in Test Problem 4					
Number of Equations: 3,900 (3-D)			IBM RS/6000 370		
SOLVER	Number of Δt's	Newtonian Iterations	Maximum N_{CG}	Minimum N_{CG}	CPU Time (sec)
MA28	Could not solve the problem due to insufficient memory.				
DSLUBC	50	249	36	8	619
DSLUCS	50	239	20	4	462
DSLUGM	50	250	28	7	451
LUBAND	50	232	-	-	6412
LUBAND(opt)	50	232	-	-	4228
LUBAND/D4	50	232	-	-	1683
BCGSTB(2)	50	236	16	8	439
BCGSTB(4)	50	235	32	8	453

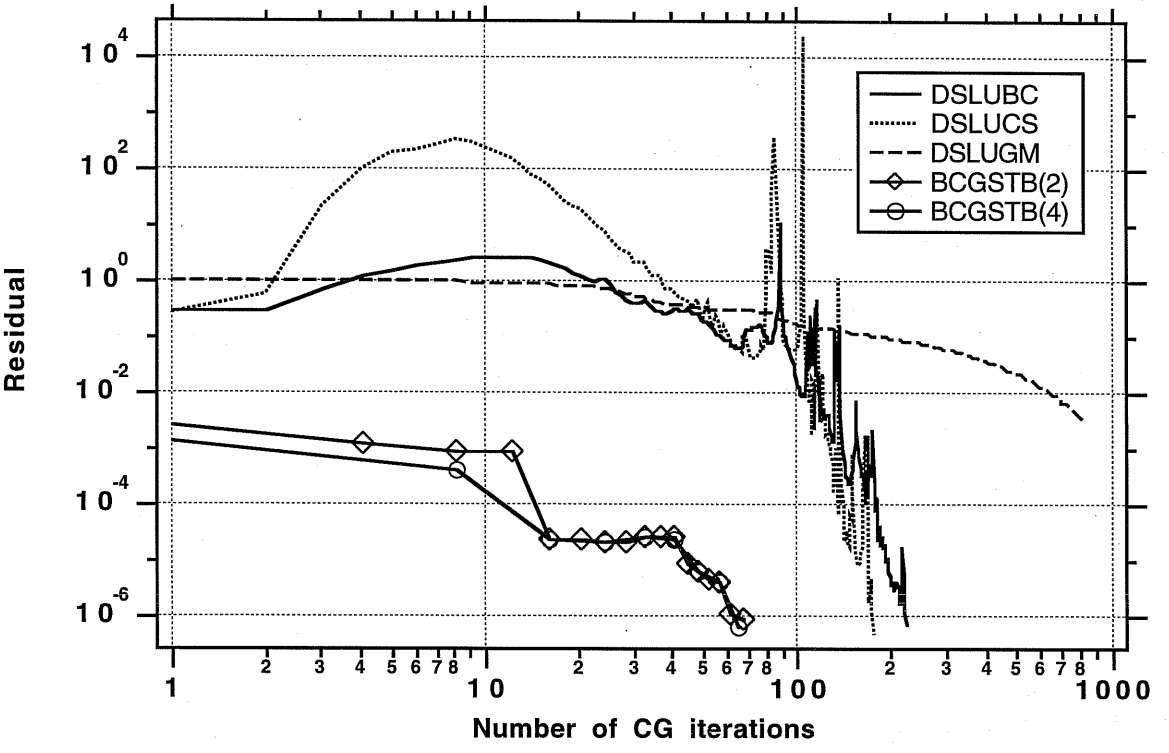

Figure 1. CG solver performance in the first Newtonian iteration of the 10th timestep in Test Problem 2.

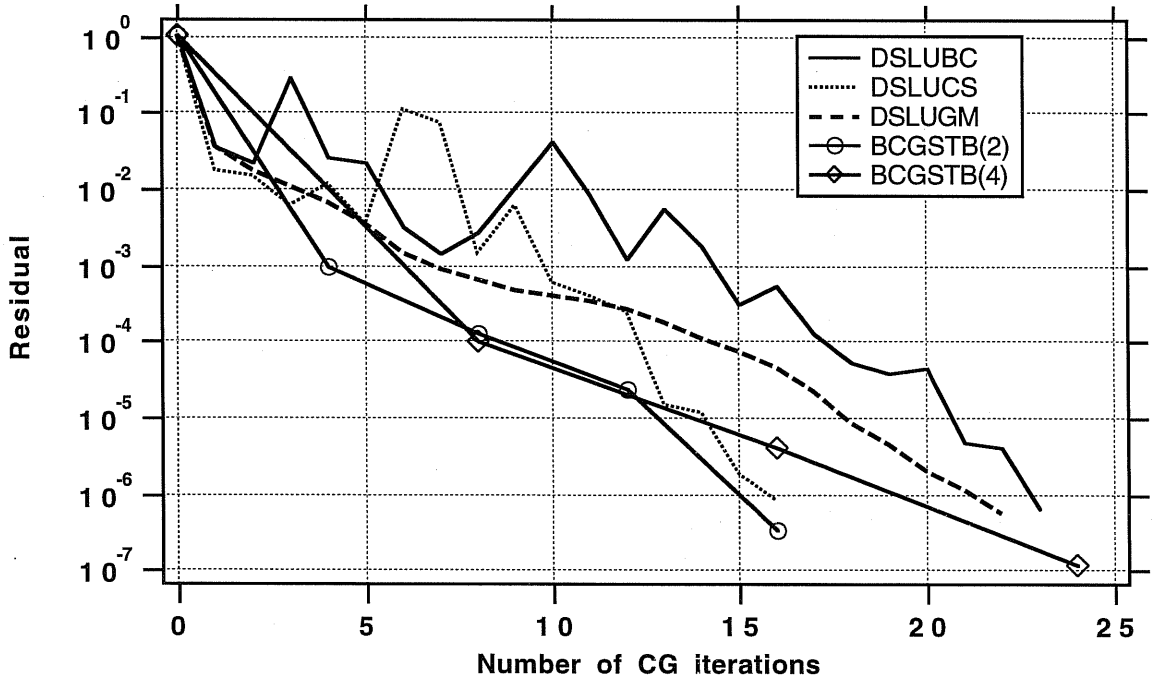

Figure 2. CG solver performance in the first Newtonian iteration of the 50th timestep in Test Problem 2.