



ERNEST ORLANDO LAWRENCE BERKELEY NATIONAL LABORATORY

T2SOLV: An Enhanced Package of Solvers for the TOUGH2 Family of Reservoir Simulation Codes

George J. Moridis and Karsten Pruess
Earth Sciences Division

November 1997



REFERENCE COPY |
Does Not |
Circulate |
Bldg. 50 Library - Ref.
Lawrence Berkeley National Laboratory

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

LBNL-40933
UC-1240

**T2SOLV: AN ENHANCED
PACKAGE OF SOLVERS FOR THE
TOUGH2 FAMILY OF
RESERVOIR SIMULATION CODES**

George J. Moridis and Karsten Pruess

**Earth Sciences Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720**

November 1997

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Geothermal Technologies, of the U.S. Department of Energy, under contract No. DE-AC03-76SF00098.

T2SOLV: AN ENHANCED PACKAGE OF SOLVERS FOR THE TOUGH2 FAMILY OF RESERVOIR SIMULATION CODES

GEORGE J. MORIDIS and KARSTEN PRUESS

*Lawrence Berkeley National Laboratory, Earth Sciences Division,
1 Cyclotron Road, 90-1116, Berkeley, California 94720*

Abstract - T2SOLV is an enhanced package of matrix solvers for the TOUGH2 family of codes. T2SOLV includes all the Preconditioned Conjugate Gradient (PCG) solvers used in T2CG1, the current solver package, as well as LUBAND, a new direct solver, and DLUSTB, a PCG solver based on the BiCGSTAB method. Additionally, T2SOLV includes the D4 grid numbering scheme and two sets of preprocessors. Results from test problems indicate that LUBAND is faster, more reliable and requires less storage than MA28, the current direct solver. BiCGSTAB solver is shown to be superior to the other PCG methods in T2SOLV. Finally, the preprocessors improve the performance of the PCG solvers and allow the solution of previously intractable problems.

INTRODUCTION

This paper discusses enhancements of the linear solvers for the TOUGH2 general-purpose fluid and heat flow simulator. TOUGH2 is capable of modeling most of the processes arising in the natural state of geothermal reservoirs and in response to production and injection operations. It can handle the appearance and disappearance of liquid and vapor phases, boiling and condensation, multiphase flow due to pressure, gravity, and capillary forces, vapor adsorption with vapor pressure lowering, heat conduction, and heat exchange between rocks and fluids. It is applicable to flow systems of arbitrary geometry from one to three dimensions, and has special provisions for flow in fractured-porous media. A brief summary of the equations and methods used in TOUGH2 is given in the appendix; additional information is available in a number of reports [*Pruess*, 1991, 1995; *Pruess et al.*, 1996, 1997), and on the web at URL <http://ccs.lbl.gov/TOUGH2/>.

Most of the computational work in the numerical simulation of fluid and heat flow 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 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 memory requirements and their speed especially in the solution of very large matrices.

In the TOUGH2 general-purpose reservoir simulator [Pruess, 1991] the matrix A is a Jacobian with a consistent structure. For a flow problem with NEQ mass-and-energy balance equations per grid block, the Jacobian consists of many NEQ -dimensional sub-matrices. For a grid block n , derivatives corresponding to accumulation terms will generate a sub-matrix in location (n,n) . Each flow term between n and a neighboring grid block m will give rise to two off-diagonal sub-matrices in locations (n,m) and (m,n) , and will also contribute to the diagonal sub-matrices at (n,n) and (m,m) . Thus, the incidence matrix will be symmetric, although, for multi-phase and non-isothermal problems, matrix A generally may be far from symmetric. Each grid block is typically connected only to a few other blocks so that A will be sparse. The Integral Finite Difference Method, IFDM, [Narasimhan and Witherspoon, 1976] used in TOUGH2 does not need to make reference to a global system of coordinates. Very irregular grid systems may be used, which may result in a nearly random, but symmetric, sparsity structure. However, for regular grid systems involving global coordinates, such as (r,z) and (x,y,z) grids, the IFDM is equivalent to conventional finite differences, giving rise to Jacobian matrices with regular banded structure.

The Jacobian matrices generated from mass and heat balance equations for multi-phase non-isothermal flows have a number of properties that can make linear equation solution quite challenging. Matrix elements can have a very large numerical range, often spanning 20 orders of magnitude or more. Off-diagonal elements are proportional to time step size, and for "practical" time steps may be orders of magnitude larger than diagonal elements. In fact, diagonal elements may often be zero (see below). Thus, TOUGH2 creates very challenging matrices which can be non-symmetric, not positive definite, not diagonally dominant and ill-conditioned, i.e., all the attributes that cause most iterative techniques to fail. In addition, the general-purpose nature of TOUGH2 allows simulation of a great diversity of flow problems which may produce quite different matrix characteristics. This explains the past heavy reliance of TOUGH2 on the robust but slow direct solver MA28 [Duff, 1977], which has large and imprecise memory requirements and in 3-D simulations is limited to impractically small problems (≤ 500 gridblocks).

In the current TOUGH2 version, T2CG1 [Moridis and Pruess, 1995], a package of preconditioned conjugate gradient solvers, complements the MA28 direct solver and significantly increases the size of tractable problems. T2CG1 includes three Preconditioned Conjugate Gradient (PCG) solvers: (a) DSLUBC, a routine based on the Bi-Conjugate Gradient (BiCG) method, (b) DSLUCS, a Conjugate Gradient Squared (CGS) routine, and (c) DSLUGM, a Generalized Minimum Residual (GMRES) routine. Tests of T2CG1 on a variety of computing platforms and for problems with Jacobian matrices of order 30,000 have shown that the PCG routines in T2CG1 are significantly (and invariably) faster than MA28 and require far less memory [Moridis and Pruess, 1995].

T2CG1 is a reliable and fast solver package for most TOUGH2 simulations. In limited cases, however, the PCG solvers in T2CG1 are challenged by classes of certain very demanding numerical simulation problems, as well as by limitations in the underlying

algorithms of the PCG (such as occasional oscillatory behavior as steady-state is approached).

In this paper we discuss T2SOLV, an enhanced package of solvers for the TOUGH2 family of codes, which was developed as a replacement for T2CG1, the current solver package. T2SOLV includes all the PCG solvers used in T2CG1 as well as a new routine, DLUSTB, based on the Bi-Conjugate Gradient Stabilized (BiCGSTAB) method. T2SOLV also replaces MA28 by LUBAND, a general banded-matrix direct solver. Additionally, it includes an option for using the D4 ordering scheme and two sets of matrix preprocessors to enhance the PCG performance.

THE LUBAND SOLVER

LUBAND is a direct solver that replaces the direct solver MA28 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 and memory management approach. It is based on a LU decomposition with partial pivoting and row interchange, and allows the solution of systems with a large number of zeroes 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 this class of solvers. Although the savings in execution time and memory are maximized in matrices with banded structure, LUBAND is capable of solving any matrix generated by TOUGH2 (i.e. even matrices with nearly-random sparsity structure) faster than MA28.

LUBAND can be applied without any problem in the current TOUGH2 version and is fully backward compatible with all older input data files. The MESHMAKER routine [Pruess, 1991], which discretizes the domain and generates the simulation grid in TOUGH2, was also enhanced to minimize the bandwidth of matrix **A**. 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 $W = NM^2$ and the minimum storage $S = NB$, where N is the order of the matrix and M its half-bandwidth, the full bandwidth being $B = 2M + 1$.

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.

THE DLUSTB SOLVER

DLUSTB was developed based on the BiCGSTAB(m) algorithm [Sleijpen and Fokkema, 1993], an extension of the BiCGSTAB algorithm of van der Vorst [1992] which is still an option in T2SOLV. It was developed to solve nonsymmetric linear systems while avoiding the irregular convergence patterns of PCG solvers 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 errors. BiCGSTAB(m) alleviates the irregular (oscillatory) convergence common to the BiCG [Fletcher, 1976] and CGS [Sonneveld, 1989] methods, thus improving the speed of convergence. It also alleviates potential stagnation or even breakdown problems which may be encountered in traditional BiCGSTAB. According to Sleijpen and Fokkema [1993], BiCGSTAB(m) combines the speed of BiCG with the monotonic residual reduction in the Generalized Minimum Residual (GMRES) method, while being faster than both. Theoretical analysis indicates that the BiCGSTAB(m) algorithm is especially well-suited to the solution of very large (i.e. $N > 50,000$) problems [van der Vorst, 1992].

The BiCGSTAB(m) algorithm is shown as pseudocode in Figure 1. The vectors \mathbf{r} are residuals, and \mathbf{M} is the preconditioner. The preconditioner used in DLUSTB is based on an incomplete LU (ILU) factorization of the matrix \mathbf{A} , which can be obtained by the slightly modified Gaussian elimination procedure described in Moridis and Pruess [1992]. The modified BiCGSTAB(m) can be interpreted as the product of the BiConjugate Gradient method BiCG [Fletcher, 1976] and repeatedly applied Generalized Minimum Residual GMRES(1) method [Saad and Schultz, 1986]. At least locally, a residual vector is minimized, leading to a considerably smoother convergence behavior. In the traditional BiCGSTAB, if the local GMRES(1) stagnates, then the Krylov subspace does not expand and the method breaks down, in addition to failure possibilities due to weaknesses of the underlying BiCG algorithm. BiCGSTAB(m) addresses this problem by combining BiCG with GMRES(m).

DLUSTB uses the Boeing-Harwell matrix storage scheme of TOUGH2, and has the same architecture as the other routines in T2SOLV. As in all other PCG solvers in T2SOLV, it uses a modified LU decomposition for preconditioning. Its memory requirements increase linearly with the order m of the Minimal Residual polynomial. For $m = 4$, it requires twice the memory of BiCG or CGS. The optimum value of m is calculated internally in DLUSTB.

THE D4 SCHEME

The Alternating Diagonal Scheme (D4) for gridblock ordering was added as an option to T2SOLV. The ordering of unknowns can drastically affect the amount of computation and storage. For a long time the best ordering scheme was the standard ordering [Aziz and Settari, 1979]. Figure 2 shows the standard ordering of a 2-D grid.

D4 is a matrix-banding technique, which derives its benefits from the numbering of the grid points. For the 2-D problem shown in Figure 2, Figure 3 depicts the D4 numbering scheme. More details can be found in Price and Coats [1974]. D4 ordering partitions the matrix into four distinct entities according to the equation

$$\mathbf{Ax} = \mathbf{b} \Rightarrow \begin{bmatrix} \mathbf{A}_{UL} & \mathbf{A}_{UR} \\ \mathbf{A}_{LL} & \mathbf{A}_{LR} \end{bmatrix} \begin{bmatrix} \mathbf{x}_U \\ \mathbf{x}_L \end{bmatrix} = \begin{bmatrix} \mathbf{b}_U \\ \mathbf{b}_L \end{bmatrix},$$

where \mathbf{A}_{UL} and \mathbf{A}_{LR} are diagonal submatrices, and \mathbf{A}_{LL} and \mathbf{A}_{UR} are sparse submatrices. This structure allows forward elimination on the equations in the lower half of \mathbf{A} , which zeroes all original entries in \mathbf{A}_{LL} and transforms it into a null matrix, while creating non-zero entries in the submatrix \mathbf{A}_{UL} in the lower right quadrant of \mathbf{A} , i.e.,

$$\mathbf{A}^* \mathbf{x} = \mathbf{b}^* \Rightarrow \begin{bmatrix} \mathbf{A}_{UL} & \mathbf{A}_{UR} \\ \mathbf{0} & \mathbf{A}_{LR}^* \end{bmatrix} \begin{bmatrix} \mathbf{x}_U \\ \mathbf{x}_L \end{bmatrix} = \begin{bmatrix} \mathbf{b}_U \\ \mathbf{b}_L^* \end{bmatrix},$$

where the asterisk denotes transformed matrices. In the transformed equation \mathbf{A}_{UL} and \mathbf{A}_{UR} remain unchanged, while \mathbf{A}_{LR}^* is a banded matrix. The equation

$$\mathbf{A}_{LR}^* \mathbf{x}_L = \mathbf{b}_L^*$$

can then be solved independently. The submatrix \mathbf{A}_{LR}^* is of order $N/2$, and allows the calculation of \mathbf{x}_L , the lower half of \mathbf{x} , from which the upper half \mathbf{x}_U is obtained by simple substitution. The resulting reduced matrix \mathbf{A}_{LR}^* can be solved using either direct (D4-direct) or iterative (D4-iterative) methods.

D4 numbering reduces the order of the matrix by 50% while not increasing the bandwidth. Depending on the grid geometry, D4 makes possible execution speed improvement by a factor ranging between 2 and 5.85 [Price and Coats, 1974] over standard ordering. Moreover, it reduces storage requirements by a factor of 2. Compared to iterative solvers, D4-direct is competitive in 2-D problems but slower in 3-D problems, while yielding a robust solution. D4 with LUBAND makes possible the robust direct solution of large multi-dimensional problems. However, D4 can only be used with regular grids.

THE Z-PREPROCESSORS

Some of the most numerically challenging matrices arising in TOUGH2 simulations involve a large number of zero entries on the main diagonal of the Jacobian. Such matrices are quite common in non-isothermal two-component systems (such as modeling of “two-water” geothermal systems involving separate tracking of two different masses of water in tracer studies) and result in at least $0.5N$ non-zero entries on the main diagonal of the matrix.

Such matrices pose no problem for the LUBAND direct solver. The iterative solvers, however, are directly affected by the diagonal dominance of the matrix and the relative number of the zero entries on the main diagonal. Up to $0.1N$ zero elements have little effect on the PCG solvers in T2SOLV. Matrices with as many as $0.3N$ (and occasionally up to $0.5N$) zero elements may be tractable without any special treatment, but usually require a large number of iterations for convergence, i.e. exceeding $0.5N$.

The three Z-preprocessors implemented in T2SOLV enhance the performance of the PCG solvers in matrices with a large number of main-diagonal zeroes. These preprocessors are invoked only when (a) PCG solvers are used, (b) the matrices have

main diagonals populated with a large number of zeroes and (c) the number of the primary variables $NEQ > 1$.

The first option, Z1, replaces the zeroes with a small number (typically 10^{-25}), and can substantially decrease the number of iterations for convergence in matrices with as many as $0.5N$ zero main-diagonal elements. The performance of the PCG solvers in Z1-processed matrices deteriorates rapidly when the number of the main-diagonal zero elements exceeds $0.5N$.

The second pre-processing option, Z2, is computationally more intensive and involves linear combinations of the flow equations in each gridblock. Z2 includes a search algorithm that identifies the appropriate equation to be added to the equation corresponding to the zero main-diagonal element. By adding the two equations, the corresponding elements in the Jacobian are replaced with the non-zero sum of the original elements. The Z2 option requires limited computational effort and significantly improves the performance of the PCG solvers.

While very effective, Z2-preprocessing can still suffer from poor conditioning because of persistent lack of diagonal dominance and large differences in the magnitude of the added elements. The problem can sometimes be alleviated by the Z3 option, which precedes the linear combination with normalization with respect to the largest element in the corresponding row. Addition of the normalized elements leads to an improved PCG performance because the relative magnitude of the elements and the corresponding roundoff error can be reduced. The Z3 option is computationally more intensive than Z2. The Z2 and Z3 preprocessors can easily handle up to $0.75N$ zero diagonal elements.

THE O-PREPROCESSORS

The O-preprocessors are applied to matrices with no zero entries on the main diagonal and aim to improve the PCG solver performance by improving the matrix conditioning. Three such preprocessors are available in T2SOLV. These options, O1 through O3, are in essence steps in the replacement of the A_M submatrix by the unit matrix through a central pivoting process, and involve increasing levels of computational effort.

The O1 option eliminates the lower half of the main-diagonal submatrix, and thus removes $NEQ-1$ subdiagonals from the global matrix. This reduces the computational effort by reducing the number of non-zero matrix entries and can improve the PCG performance. Execution times are burdened by the additional work for the elimination of the lower half of the matrix, but usually this is overcome by the savings in the PCG computations.

In the O2 option, in addition to O1 the upper half of the main-diagonal submatrix is eliminated, resulting in a diagonal submatrix and eliminating an additional $NEQ-1$ superdiagonals from the global matrix. Compared to the original, the O2-processed matrix is significantly sparser and better-conditioned and the performance of the PCG solvers can be enhanced. The increased computational effort for the O2 preprocessing is usually compensated by the reduction in the PCG iterations.

The O3 option involves normalization of the O2 matrix, resulting in a unity main diagonal. O3 does not further increase matrix sparsity, but may improve the matrix conditioning.

TEST PROBLEMS

Test Problems 1a and 1b

Test problems 1a and 1b involve studies of non-isothermal flow in a "two-water" system. Such systems are known to be the most challenging for the solvers in TOUGH2, as they routinely create matrices with $0.67N$ zeros on the main diagonal. The PCG routines in T2CG1 have in the past been unable to solve even the smallest of this class of problems. The problem discussed here involves injection of "water 2" at a temperature of 30°C into a geothermal reservoir of "water 1" at 280°C . The EOS1 module of TOUGH2 is used [Pruess, 1991]. The two different masses of water are tracked independently. This system is composed of 2 water components and involves 3 equations per gridblock ($NK = 2$ and $NEQ = 3$, respectively, in the TOUGH2 nomenclature). The geometry, properties and discretization of the two problems are shown in Table 1.

Problem 1a is one-dimensional, and consists of 5 gridblocks at the geothermal reservoir conditions. Cold water is injected into the first gridblock at the rate indicated in Table 1 for a total of 20 timesteps. The small size of the problem demonstrates that the computational difficulties encountered in this class of problems are not related to the size of the matrix, but rather to fundamental issues of matrix conditioning. Table 2 and Figures 4 through 8 show the performance of the various solvers in T2SOLV. For comparison, the same scale was used in Figures 4 through 8, in which the closure criterion of the conjugate gradient (CG) iterations (10^{-6}) is indicated by a horizontal dashed line.

The results are shown in Table 2. MA28 and LUBAND can solve this problem without difficulty. None of the PCG methods is capable of solving the matrix with Z0 (i.e. no preprocessing) or Z1 matrix preprocessing. With Z2 and Z3 preprocessing, all iterative methods can solve the problem but their performance differs significantly.

The performance of DSLUBC with the Z2 and Z3 preprocessors (Figure 4) is practically the same, requiring a total of 146 and 142 CG iterations respectively. With Z2, the maximum number of allowable iterations (21) is reached only once and the CG closure criterion is always met. With Z3, the maximum number of allowable iterations is reached only once but at a different time step than for Z2. The CG closure criterion is not met once. However, this does not pose a problem as the solution is sufficiently accurate to satisfy the Newtonian convergence criterion of 10^{-5} . It is obvious that the most challenging matrices arise during the first few timesteps, after which solutions are obtained within 2-4 iterations.

DSLUCS with Z2 (Figure 5) reaches the maximum number of allowable iterations (21) at the first timestep (which needs a single Newtonian iteration), at which the CG closure criterion is not met, but which satisfies the Newtonian convergence criterion. At the first timestep, DSLUCS with Z3 requires a single Newtonian iteration and a single CG iteration to obtain a very accurate solution (a residual of 9.4×10^{-22}). After the first timestep, the DSLUCS performance with Z3 has a distinct advantage in terms of total number of CG iterations (70 vs. 92 for Z2). With both Z2 and Z3, 6 to 10 iterations are

required for the CG solutions, a relatively large number compared to the size of the problem (15 equations).

The performance of DSLUGM (Figure 6) is practically the same with either Z2 or Z3 preprocessing, and requires the least number of total CG iterations (61 and 60 respectively). CG convergence in this case is achieved within 1 or 2 iterations at all the timesteps and Newtonian iterations. DSLUGM with either Z2 or Z3 appears to be the best overall solver of Test Problem 1a.

DLUSTB with Z2 (Figure 7) is unable to solve the problem, and the matrix solution is stopped by the TOUGH2 main program at the fourth timestep after Newtonian convergence on the first iteration at two successive timesteps (and following repeated timestep cutbacks). When employing DLUSTB with Z2, the maximum number of allowable iterations (21) is reached at each Newtonian iteration, while the residuals do not meet the CG closure criterion. The problem can be alleviated by reducing the Δt_0 to 0.4 s. DLUSTB with Z3 has a much better performance, and while it reaches the limit of 21 CG iterations in the first three timesteps, it is capable of solving the problem at all 20 timesteps. The residual at the third timestep exceeds the CG closure criterion, but the solution is sufficiently accurate to satisfy the Newtonian convergence criterion. It is noteworthy that after the first few iterations, DSLUGM with Z3 requires consistently the fewest iterations to convergence, i.e. 1 or 2.

The problem specificity of the PCG solvers is demonstrated in Figure 8, which shows the performance of DLUSTB with Z2 and Z3 in a variant of test Problem 1a, in which water is injected into the third (as opposed to the first) gridblock. With this minor change, DLUSTB with Z2 manages to solve the problem despite reaching the limit of 21 iterations on the first 6 timesteps and not achieving the CG closure criterion three times. After the sixth timestep, the solution of the matrix poses no particular problem and is attained within 1-3 iterations. The DLUSTB performance with Z3 is better than with Z2 in the first timesteps, and similar to that shown in Figure 7. After the sixth timestep, the Z2 and Z3 preconditioning have practically the same effect on the DLUSTB performance, which requires 1-2 iterations for convergence.

With Z2 and Z3 preprocessing, the execution times are practically the same for all solvers (Table 2). The Jacobians at the first Newtonian iteration of the first timestep of Problem 1a for Z0, Z2 and Z3 preprocessing are shown in Figures 9, 10, and 11 respectively. The zeros on the main diagonal in Figure 9 are replaced by non-zero entries in Figure 10 after linearly combining the gridblock equations by the Z2 preprocessor. Further manipulations result in the Z3 matrix of Figure 11.

Problem 1b involves a 3-D domain consisting of $9 \times 8 \times 5 = 360$ gridblocks in (x,y,z) , resulting in a total of $N = 1080$ equations. The fundamental weakness of MA28, i.e. its large (especially for 3-D problems) and not well defined memory requirement, is obvious in the problem. Despite memory allocation 15 times larger than the one needed for the LUBAND solution, MA28 could not complete the LU decomposition due to insufficient memory.

Table 3 and Figure 12 show that DLUSTB has the best performance. It is the fastest and requires the least number of PCG iterations to convergence. DLUSTB seems to be the only solver that can proceed with Z1-preprocessing. Note that the use of the Z-preprocessors makes possible the solution of a previously-intractable problem by all the

PCG solvers in T2SOLV. The Z2 preprocessor seems to offer the best overall performance.

Test Problem 2

Test problem 2 involves simulation of a laboratory experiment in a heat convection cell. 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 $16 \times 26 = 416$ gridblocks in (r,z) , with $NK = 1$ and $NEQ = 2$, resulting in a total of $N = 832$ equations.

Table 4 and Figures 13 and 14 show the performance of the various solvers in Problem 2, which does not pose any significant challenges to the T2SOLV routines. DLUSTB is the fastest routine and requires the least number of iterations to convergence.

In this 2-D problem, LUBAND appears as a competitive alternative. The effect of the O1 preprocessor is pronounced in terms of PCG iterations and execution times in DSLUBC and DLUSTB, but seems to be limited in DSLUCS and DSLUGM. The evolution of residuals of DSLUCS and DSLUGM in the first Newtonian iteration of the first timestep is identical with and without O1 preprocessing (Figures 13 and 14), while the DSLUCS execution time with O1 increases. Conversely, the use of the O2 and O3 preprocessors seems to offer the greatest improvement in the performance of DSLUCS and DSLUGM.

Test Problem 3

Test problem 3 examines fluid and mass flow in a mid-sized three-dimensional model of a geothermal reservoir. The basic computational grid is composed of $15 \times 15 \times 20 = 4500$ grid blocks in (x,y,z) . Cold water is injected through 4 wells, while hot water is withdrawn from 5 wells. EOS1 is used with $NK = 1$, $NEQ = 2$, resulting in a total of $N = 9000$ equations.

The solver performance is shown in Table 5. This is a relatively large but well-behaved problem, the size of which precluded the use of a direct solver. The use of D4 allowed a direct solution by LUBAND, which is competitive with the PCG solutions. D4 with DLUSTB had a performance on a par with DLUSTB, the fastest PCG solver. In light of the minor overhead needed to set up the D4 system, this result is very encouraging.

DLUSTB demonstrated its superiority by being the fastest solver and requiring the least number of PCG iterations to convergence. DSLUGM seems to be an inappropriate method for this type of problem. As expected, the benefits of O-preprocessing in this well-behaved system are not evident in the execution times, although the number of PCG iterations are often reduced. It is noteworthy, however, that despite the increased computational load, the execution times for the O-preprocessed solutions are practically identical to those without any preprocessing.

An important feature of TOUGH2 is the user's complete control over the numbering sequence of the gridblocks. To demonstrate the robustness of the iterative solvers, the order of the elements in the input file was rearranged by (a) generating 4500 random numbers, (b) ranking them, and (c) renumbering the sequence of grid elements according to

the ranking of the corresponding random number. This resulted in a matrix which, while maintaining the same number of non-zero entries, had a sparsity pattern which was almost symmetric but practically random in appearance. The effect of random numbering on the ability of the iterative solvers to produce a solution is shown in Table 6. All the solvers were capable of solving the problem, but required longer execution times than for the well-ordered case because the sparsity pattern of the matrix does not allow taking full advantage of the benefits of preconditioning and PCG solvers, which are most pronounced in well-ordered banded matrices. DLUSTB is again the fastest solver and requires the least number of PCG iterations, and DSLUCS is the second best solver. The execution times of the solvers are about 50% longer than for the well-ordered case, while that for DSLUGM is about 150% longer. It is remarkable that the relative speed of the solvers also remains the same as in the well-ordered case.

Test Problem 4.

This problem examines non-isothermal flow in a simple two-dimensional model of a heterogeneous porous medium. The basic computational grid has a grid spacing of $\Delta x = 0.25\text{ m}$, $\Delta y = 0.125\text{ m}$, for a total of $80 \times 120 = 9600$ grid blocks (Figure 15). The y -axis is rotated 90° against the horizontal to make the section vertical. A mesh preprocessing program is then used to place impermeable obstacles with lengths uniformly distributed in the range of $2\text{-}4\text{ m}$ (Figure 15). Problem parameters are chosen representative of typical alluvial soils and are given in Table 7.

The heterogeneous medium described above has been used to study the behavior of liquid infiltration plumes in isothermal systems [Pruess, 1994], and has been discussed in detail by Moridis and Pruess [1995]. In this problem, water at a temperature of $30\text{ }^\circ\text{C}$ is injected uniformly at a total rate of 1 kg/s into the fully-saturated domain across the top of the domain, while the bottom boundary is maintained at a constant pressure and temperature. The entire domain is initialized in single-phase conditions, at a pressure of $P = 4.0 \times 10^7\text{ Pa}$, and a temperature of $280\text{ }^\circ\text{C}$. The simulation is performed with the EOS1 fluid property module using $NK = 1$ and $NEQ = 2$, for a total of $N = 19200$ equations. This problem was chosen because it had confounded both the DSLUBC and the DSLUGM solvers [Moridis and Pruess, 1995], i.e. the building blocks of the traditional Bi-CGSTAB method. The performance of the DLUSTB was expected to be an indicator of the robustness of the Bi-CGSTAB(m) algorithm.

The simulation results are shown in Table 8. The size of the problem precluded the use of direct solvers. The superiority and efficiency of the DLUSTB routine in the solution of this problem is clear. DLUSTB required 1811 CPU sec and was significantly faster than DSLUCS, which required 2213 CPU sec. They were both significantly faster than DSLUBC (10063 sec) and DSLUGM (10135), both of which reached repeatedly the maximum number of iterations (because of break-down or stagnation) and needed repeated timestep size reductions in order to complete the 10-timestep run.

CONCLUSIONS AND SUMMARY

The following conclusions can be drawn:

(1) Without any matrix preprocessing, the BiCGSTAB algorithm coded in DLUSTB is shown to be a fast and efficient solver which outperforms the other PCG routines. It is the fastest and the most robust in T2SOLV and is shown to be practically free of stagnation, oscillation, and divergence problems.

(2) The use of the Z-preprocessors makes possible the solution of problems which were previously intractable to all the PCG solvers. The combination of the Z-preprocessors with the BiCGSTAB routine gives the best performance in such problems.

(3) In problems which are known to confound the other PCG solvers, DLUSTB converges smoothly but slowly to a solution without invoking the matrix-preprocessing facility.

(4) The O-preprocessors are shown to improve the robustness and decrease the number of iterations to convergence, but their effect depends on the PCG solver in T2SOLV. DLUSTB appears to be the solver most consistently responsive to the O-preprocessors. In well-behaved problems the effect of the O-preprocessors on the execution speed is not significant.

(5) LUBAND is shown to be consistently faster and more reliable than MA28, and can solve much larger problems.

(6) The gains in execution speed when the D4 scheme is used in regular grids are shown to be significant (especially compared to the direct solution). D4-direct seems to be competitive (in speed) to the PCG solvers in medium-sized problems.

In large problems (especially in 3-D systems) and when not limited by significant memory requirements, D4-direct will still offer a predictably large improvement in execution speed over the direct solution, but is expected to be consistently and significantly outperformed by the PCG solvers. The performance of the D4-iterative approach (in which the reduced matrix is solved by the PCG solvers) has not yet been fully assessed.

T2SOLV enables the user of the TOUGH2 family of codes to solve some of the most challenging numerical problems (previously tractable only with direct solvers) using the PCG routines. The suite of PCG solvers includes all the T2CG1 routines, and is enhanced by the addition of DLUSTB (based on the BiCGSTAB(m) algorithm), which combines speed of convergence with monotonic residual reduction and alleviates the oscillatory behavior of solutions as steady-state is approached (a common problem to most PCG solvers). T2SOLV enhances the performance and robustness of the PCG solvers by introducing a set of matrix preprocessors. Additionally, it introduces LUBAND, a new direct solver capable of solving problems orders of magnitude larger than the MA28 routine in T2CG1. It also doubles the size of problems tractable with direct solvers by implementing a D4 Alternative Diagonal Numbering option.

Acknowledgement - This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Geothermal Technologies, of the U.S. Department of Energy, under 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.
- Battistelli, A., C. Calore and K. Pruess (1997), The Simulator TOUGH2/EWASG for Modeling Geothermal Reservoirs with Brines and Non-Condensable Gas, *Geothermics*, **26**(4), pp. 437-464.
- 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.
- International Formulation Committee (1967), A Formulation of the Thermodynamic Properties of Ordinary Water Substance, IFC Secretariat, Dusseldorf, Germany.
- 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, Berkeley, CA.
- 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, Berkeley, CA.
- Narasimhan, T. N. and P. A. Witherspoon (1976), An Integrated Finite Difference Method for Analyzing Fluid Flow in Porous Media, *Water Res. Res.*, **12**(1), 57-64.
- Price, H. S. and K. H. Coats (1974), Direct methods in reservoir simulation, Trans. SPE of AIME (SPEJ), **257**, 295-308.
- Pruess, K. (1991), TOUGH2-A general-purpose numerical simulator for multiphase fluid and heat flow, Lawrence Berkeley Laboratory Report LBL-29400, Berkeley, CA.
- Pruess, K. (1994), On the validity of a Fickian diffusion model for the spreading of liquid infiltration plumes in partially saturated heterogeneous media, Lawrence Berkeley Laboratory Report LBL-35134, Berkeley, CA.
- Pruess, K., editor (1995), Proceedings of the TOUGH Workshop '95, Lawrence Berkeley Laboratory Report LBL-37200, Berkeley, CA.
- Pruess, K., A. Simmons, Y.S. Wu and G. Moridis (1996), TOUGH2 Software Qualification, Lawrence Berkeley National Laboratory Report LBL-38383, Berkeley, CA.
- Pruess, K., S. Finsterle, G. Moridis, C. Oldenburg, and Y.S. Wu (1997), General-Purpose Reservoir Simulators: The TOUGH2 Family, GRC Bulletin, pp. 53-57, February 1997. (also: Lawrence Berkeley National Laboratory Report LBL-40140)
- Saad, Y. and M.H. Schultz (1986), GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.*, **7**(3), 856-869.
- Sleijpen, G.L.G. and D. Fokkema (1993), BiCGSTAB(*m*) for linear equations involving unsymmetric matrices with complex spectrum, *Electronic Transactions 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. Parameters for Test Problems 1a and 1b

Initial Pressure	$P = 4.0 \times 10^7 \text{ Pa}$
Initial Temperature	$T = 280 \text{ }^\circ\text{C}$
Permeability	$k = 2.0 \times 10^{-13} \text{ m}^2$
Porosity	$\phi = 0.15$
Relative Permeability	
All phases perfectly mobile $k_{rl} = k_{rg} = 1$	
Capillary Pressure	
Zero capillary pressure	
Geometry of Flow Domain -Problem 1a	
1-D horizontal (x) section Gridding	5x1x1 blocks in (x,y,z) $\Delta x = 200 \text{ m}$ $\Delta y = 200 \text{ m}$ $\Delta z = 100 \text{ m}$
Injection wells Rate and location	1 30 kg/s at (1,1,1)
Geometry of Flow Domain -Problem 1b	
3-D horizontal (x,z) section gridding	9x8x5 = 360 blocks $\Delta x = 200 \text{ m}$ $\Delta y = 200 \text{ m}$ $\Delta z = 100 \text{ m}$
Injection wells Rate and location	1 30 kg/s at (3,3,5)

**Table 2. Solver Performance in Problem 1a - $\Delta t_0 = 4$ s
(Macintosh PowerPC 9500/132)**

SOLVER	PP	Dts	NI	Imx(#)	Imn	IT	ET(s)
MA28	-	20	51	-	-	-	0.58
LUBAND	-	20	51	-	-	-	0.52
DSLUBC	-	Fails					
	Z1	Fails					
	Z2	20	51	21(1)	2	136	0.56
	Z3	20	53	21(2)	2	132	0.57
DSLUCS	-	Fails					
	Z1	Fails					
	Z2	20	51	21	2	92	0.57
	Z3	20	51	4	1	70	0.56
DSLUGM	-	Fails					
	Z1	Fails					
	Z2	20	51	3	1	61	0.57
	Z3	20	51	3	1	60	0.56
DLUSTB	-	Fails					
	Z1	Fails					
	Z2	Fails					
	Z2 ($\Delta t_0=0.4s$)	20	46	21	1	171	0.58
	Z3	20	51	21	1	136	0.58

PP: Preprocessing

Dts: Number of timesteps

NI: Newtonian iterations

Imx(#): Maximum number of PCG iterations (number of times reached)

Imn: Minimum number of PCG iterations

IT: Total PCG iterations

ET: Execution time

Table 3. Solver Performance in Problem 1b (Macintosh PowerPC 9500/132)							
SOLVER	PP	Dts	NI	Imx	Imn	IT	ET(s)
MA28	Fails - insufficient memory						
LUBAND	-	8	25	-	-	-	63.9
DSLUBC	-	Fails					
	Z1	Fails					
	Z2	8	29	109	5	830	29.0
	Z3	8	28	109	7	877	29.6
DSLUCS	-	Fails					
	Z1	Fails					
	Z2	12	41	109	4	1002	38.6
	Z3	12	46	109	4	1253	46.5
DSLUGM	-	Fails					
	Z1	Fails					
	Z2	8	30	51	7	531	21.3
	Z3	8	28	27	2	496	19.3
DLUSTB	-	Fails					
	Z1	15	94	109	7	642	93.2
	Z2	8	25	20	4	289	16.9
	Z3	8	25	27	4	288	17.0

Table 4. Solver Performance in Problem 2 (Macintosh PowerPC 9500/132)							
SOLVER	PP	Dt	NI	Imx	Imn	IT	ET
MA28	-	26	108	-	-	-	78.1
LUBAND	-	27	111	-	-	-	55.8
DSLUBC	-	28	119	45	27	3275	56.2
	O1	26	105	41	23	2798	49.1
	O2	28	110	42	23	2868	50.9
	O3	26	106	53	21	2867	50.4
DSLUCS	-	26	112	37	18	2437	49.7
	O1	29	131	33	18	2851	59.0
	O2	28	111	35	18	2281	48.6
	O3	27	106	35	18	2187	46.9
DSLUGM	-	26	107	59	21	3644	50.3
	O1	26	105	58	21	3532	49.4
	O2	26	96	59	21	3021	44.0
	O3	26	98	59	21	3118	45.2
DLUSTB	-	26	106	37	1	1829	42.1
	O1	26	98	41	1	1685	38.9
	O2	26	102	39	1	1684	40.6
	O3	26	98	30	14	1728	39.7

Table 5. Solver Performance in Problem 3 (IBM RS/6000 370)							
SOLVER	PP	Dt	NI	Imx	Imn	IT	ET
MA28	Insufficient Memory						
LUBAND	Insufficient Memory						
D4 + LUBAND	-	10	46	-	-	-	786
D4 + DLUSTB	-	10	46	57	43	1736	426
DSLUBC	-	10	46	106	63	2623	579
	O1	10	46	75	57	2477	565
	O2	10	46	75	52	2475	563
DSLUCS	-	10	46	95	50	2051	488
	O1	10	46	98	50	2034	485
	O2	10	46	94	50	2033	487
DSLUGM	-	10	46	930	95	14842	2087
	O1	10	46	930	95	14994	2178
	O2	10	46	930	95	15113	2189
DLUSTB	-	10	46	58	41	1736	423
	O1	10	46	60	37	1695	421
	O2	10	46	58	37	1719	429

Table 6. Solver Performance in Problem 3 with Random Element Numbering (No O-Preprocessing) (IBM RS/6000 370)							
SOLVER	PP	Δt	NI	Imx	Imn	IT	ET(sec)
DSLUBC	-	25	165	417	78	13697	1954
DSLUCS	-	25	165	901	61	11998	1847
DSLUGM	-	25	175	930	138	84702	7383
DLUSTB	-	25	164	513	52	8899	1520

Table 7. Parameters for Test Problem 4

Permeability	$k = 10^{-11} \text{ m}^2$
Porosity	$\phi = 0.35$
Relative Permeability	
<p><i>van Genuchten</i> function [1980]</p> $k_{rl} = \sqrt{S^*} \left\{ 1 - \left(1 - [S^*]^{1/\lambda} \right)^\lambda \right\}^2$ <p>irreducible water saturation exponent</p>	$S^* = (S_1 - S_{lr}) / (1 - S_{lr})$ $S_{lr} = 0.15$ $\lambda = 0.457$
Capillary Pressure	
<p><i>van Genuchten</i> function [1980]</p> $P_{cap} = -(\rho_w g/a) \left([S^*]^{-1/\lambda} - 1 \right)^{1-\lambda}$ <p>irreducible water saturation exponent strength coefficient</p>	$S^* = (S_1 - S_{lr}) / (1 - S_{lr})$ $S_{lr} = 0.0 \text{ or } 0.15$ $\lambda = 0.457$ $a = 5 \text{ m}^{-1}$
Geometry of Flow Domain	
<p>2-D vertical (x,y) section</p> <p>width (x)</p> <p>depth (y)</p> <p>gridding</p>	<p>20 m</p> <p>15 m</p> <p>$80 \times 120 = 9600$ blocks</p> <p>$\Delta x = 0.25 \text{ m}$</p> <p>$\Delta y = 0.125 \text{ m}$</p>
heterogeneity: stochastic distribution of impermeable obstacles	

**Table 8. Solver Performance in Problem 4
(IBM RS/6000 370)**

SOLVER	PP	Dt	NI	Imx	Imn	IT	ET(sec)
DSLUBC	-	10	100	1601	62	83041	10063
DSLUCS	-	10	84	321	36	14994	2213
DSLUGM	-	10	83	1620	49	90516	10135
DLUSTB	-	10	84	315	117	11681	1811

```

x(0) = 0
r(0) := b - Ax(0);
r̃ = r(0);
for i = 1, 2, ...
     $\rho_{-1} = \tilde{\mathbf{r}}^T \mathbf{r}^{(i-1)}$ ;
    if  $\rho_{-1} = 0$  method fails;
    if i = 1
         $\mathbf{p}^{(0)} = \mathbf{r}^{(i-1)}$ ;
    else
         $\beta_{i-1} = \left( \frac{\rho_{i-1}}{\rho_{i-2}} \right) \frac{\alpha_{i-1}}{\omega_{i-1}}$ ;
         $\mathbf{p}^{(i)} = \mathbf{r}^{(i-1)} + \beta_{i-1} (\mathbf{p}^{(i-1)} - \omega_{i-1} \mathbf{v}^{(i-1)})$ ;
    end if
    solve  $\mathbf{M}\hat{\mathbf{p}} = \mathbf{p}^{(i)}$ ;
     $\mathbf{v}^{(i)} = \mathbf{A}\hat{\mathbf{p}}$ ;
     $\alpha_i = \rho_{i-1} / \tilde{\mathbf{r}}^T \mathbf{v}^{(i)}$ ;
     $\mathbf{s} = \mathbf{r}^{(i-1)} - \alpha_i \mathbf{v}^{(i)}$ ;
    if  $\|\mathbf{s}\|_n$  acceptable
         $\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha_i \hat{\mathbf{p}}$ ;
    else
        solve  $\mathbf{M}\hat{\mathbf{s}} = \mathbf{s}$ ;
         $\mathbf{t} = \mathbf{A}\hat{\mathbf{s}}$ ;
         $\omega_i = \mathbf{t}^T \mathbf{s} / \mathbf{t}^T \mathbf{t}$ ;
         $\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha_i \hat{\mathbf{p}} + \omega_i \hat{\mathbf{s}}$ ;
         $\mathbf{r}^{(i)} = \mathbf{s} - \omega_i \mathbf{t}$ ;
    end if
end do

```

Fig. 1. The BiCGSTAB(*m*) algorithm.

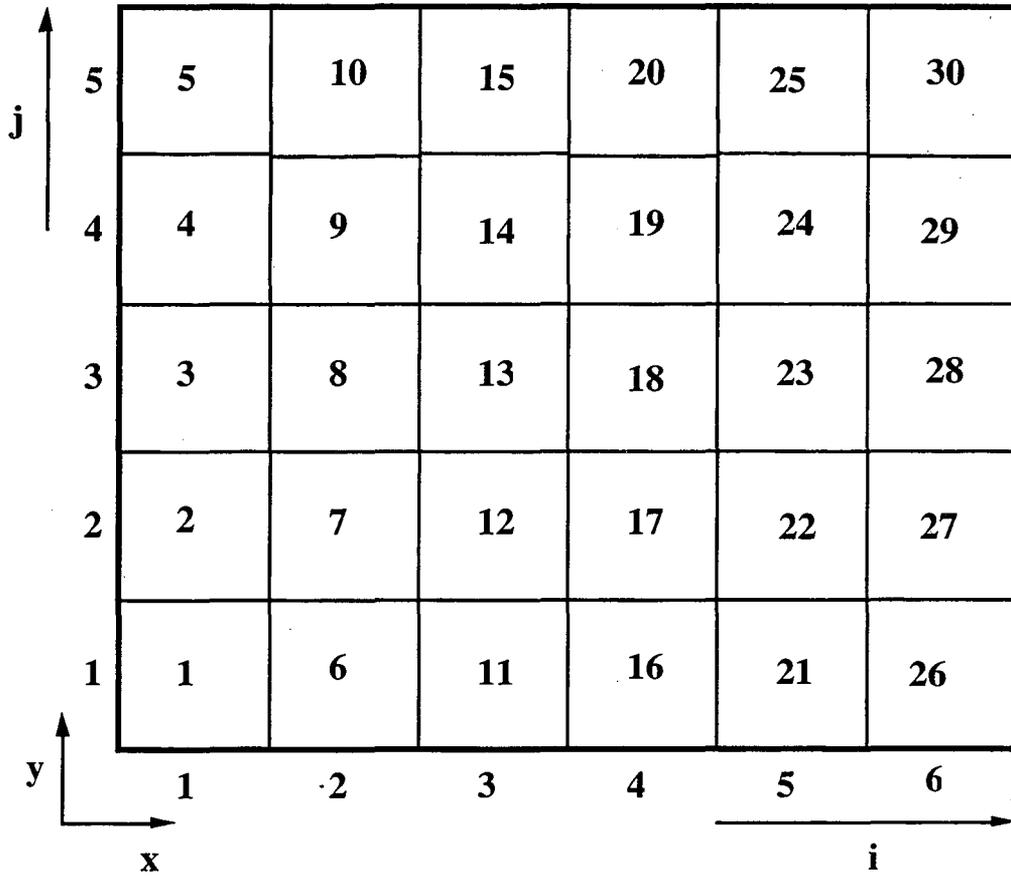


Fig. 2. Natural or standard ordering for a 2-D grid.

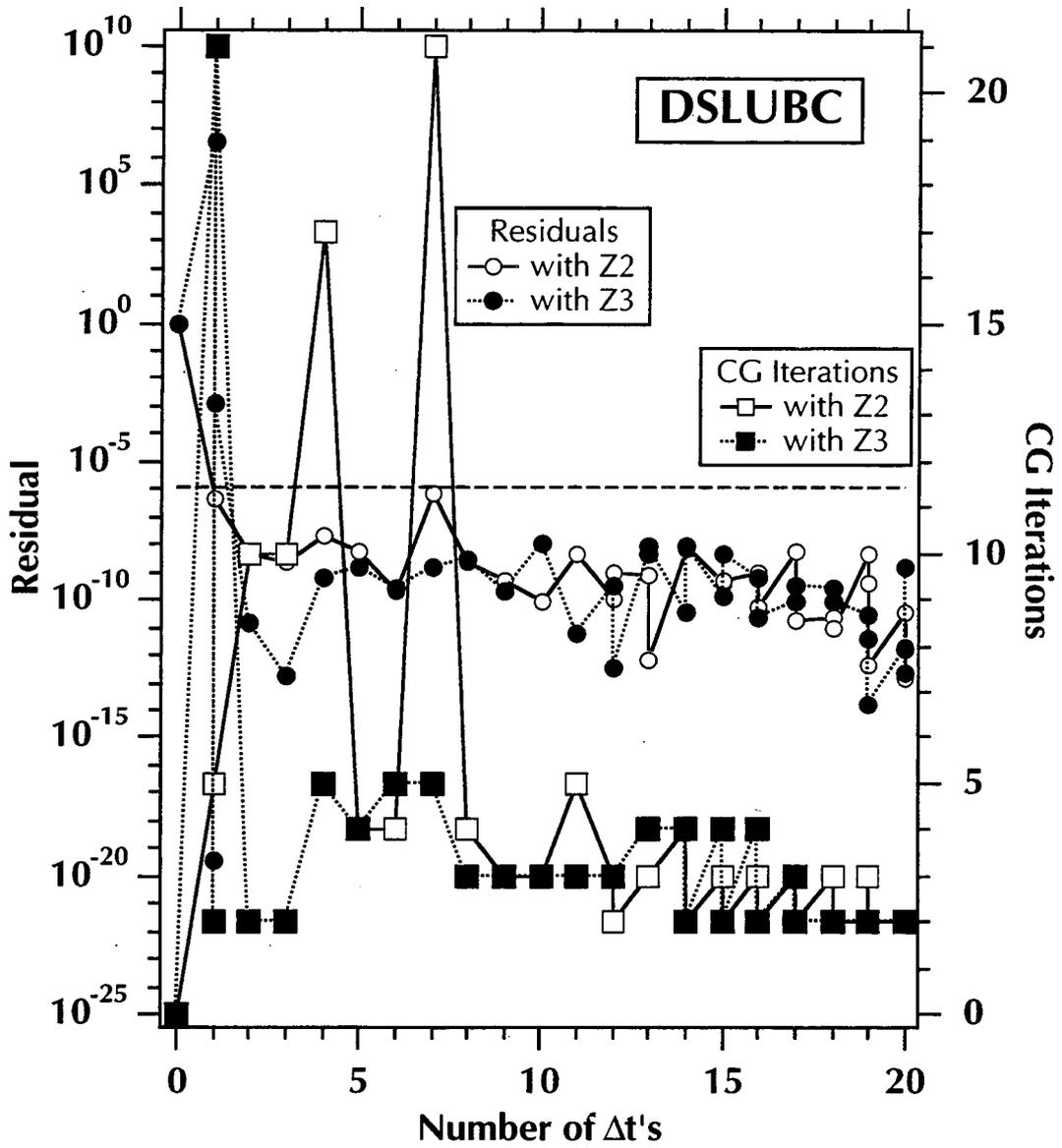


Fig. 4. DSLUBC performance in Test Problem 1a.

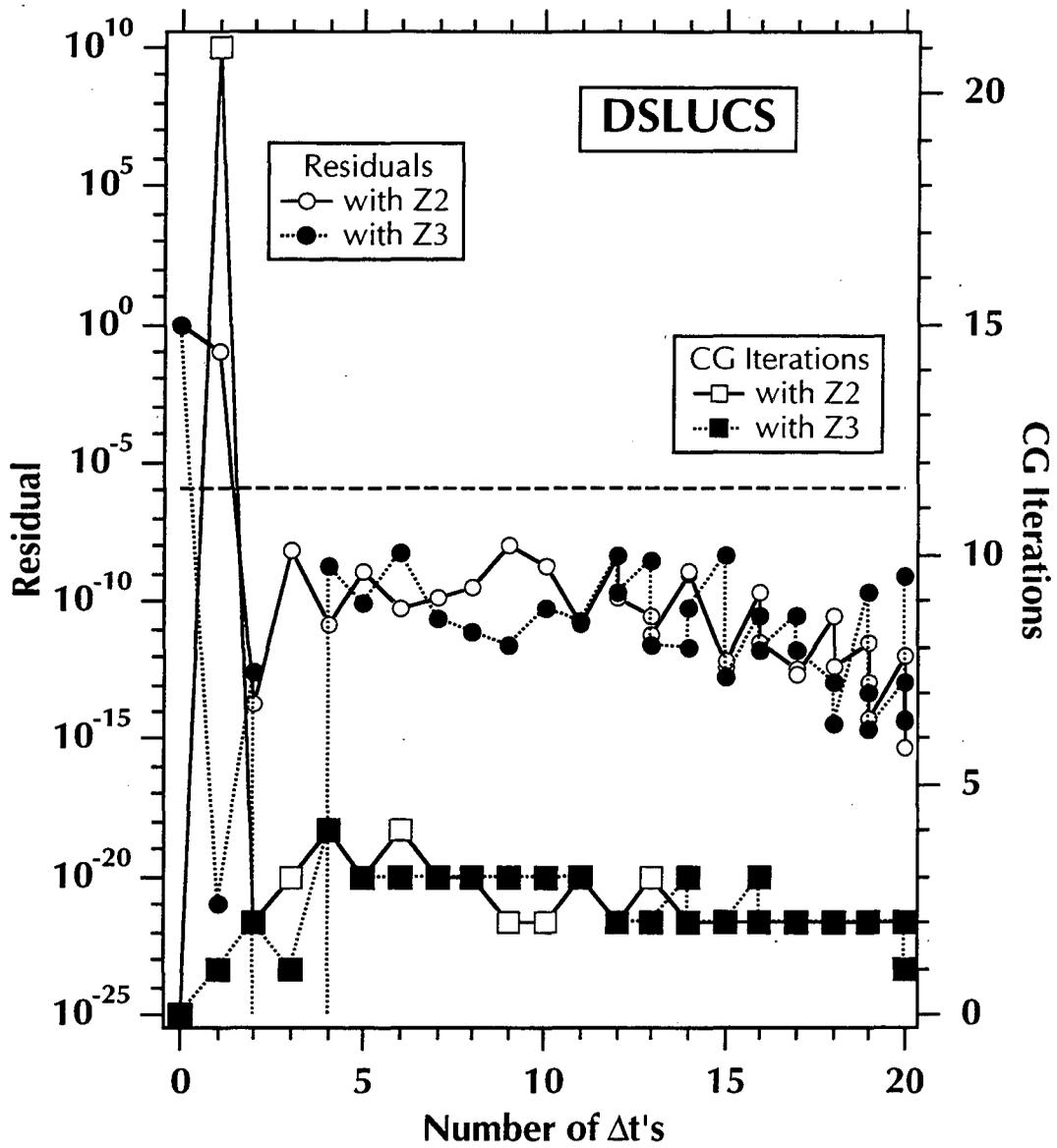


Fig. 5. DSLUCS performance in Test Problem 1a.

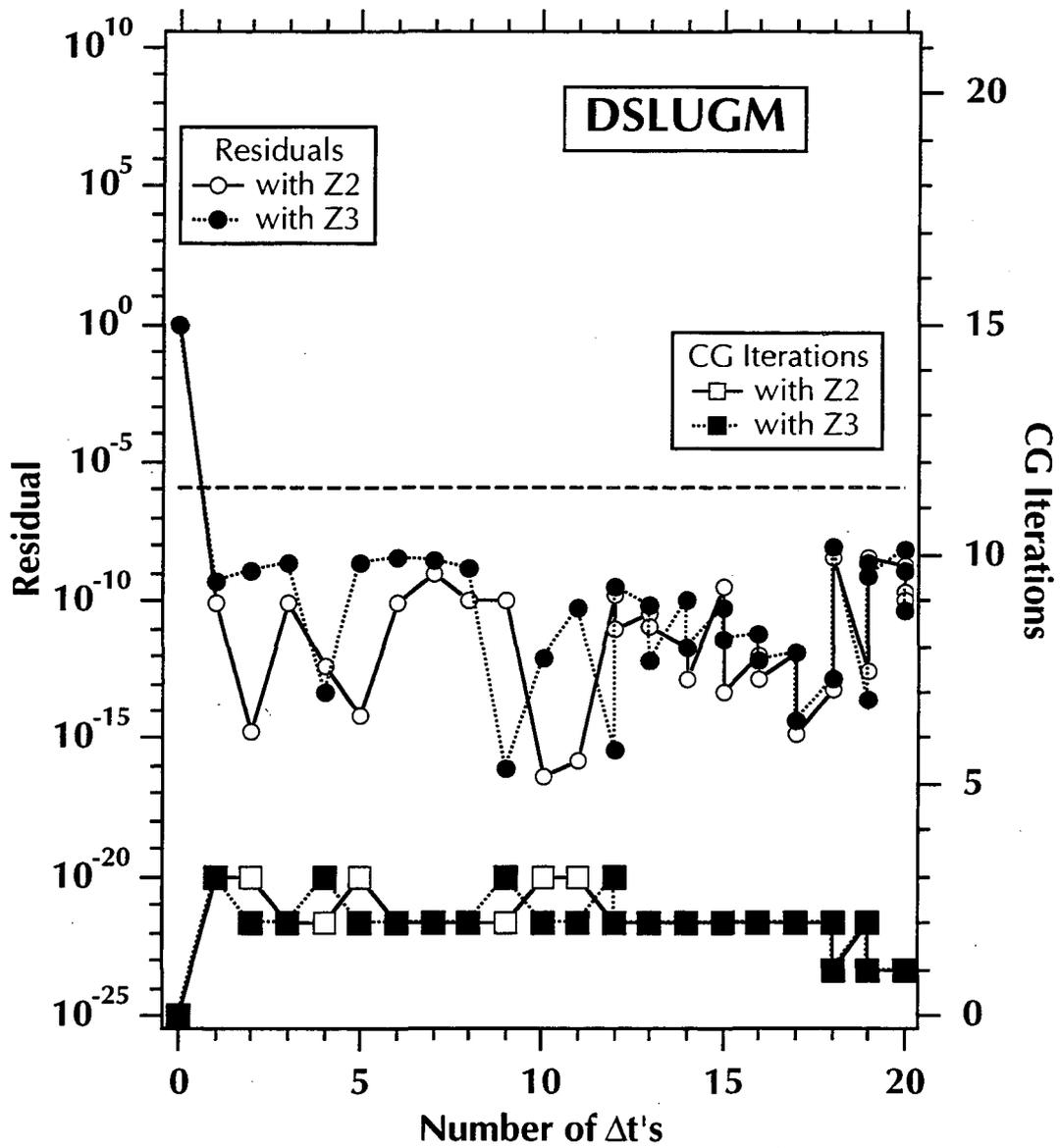


Fig. 6. DSLUGM performance in Test Problem 1a.

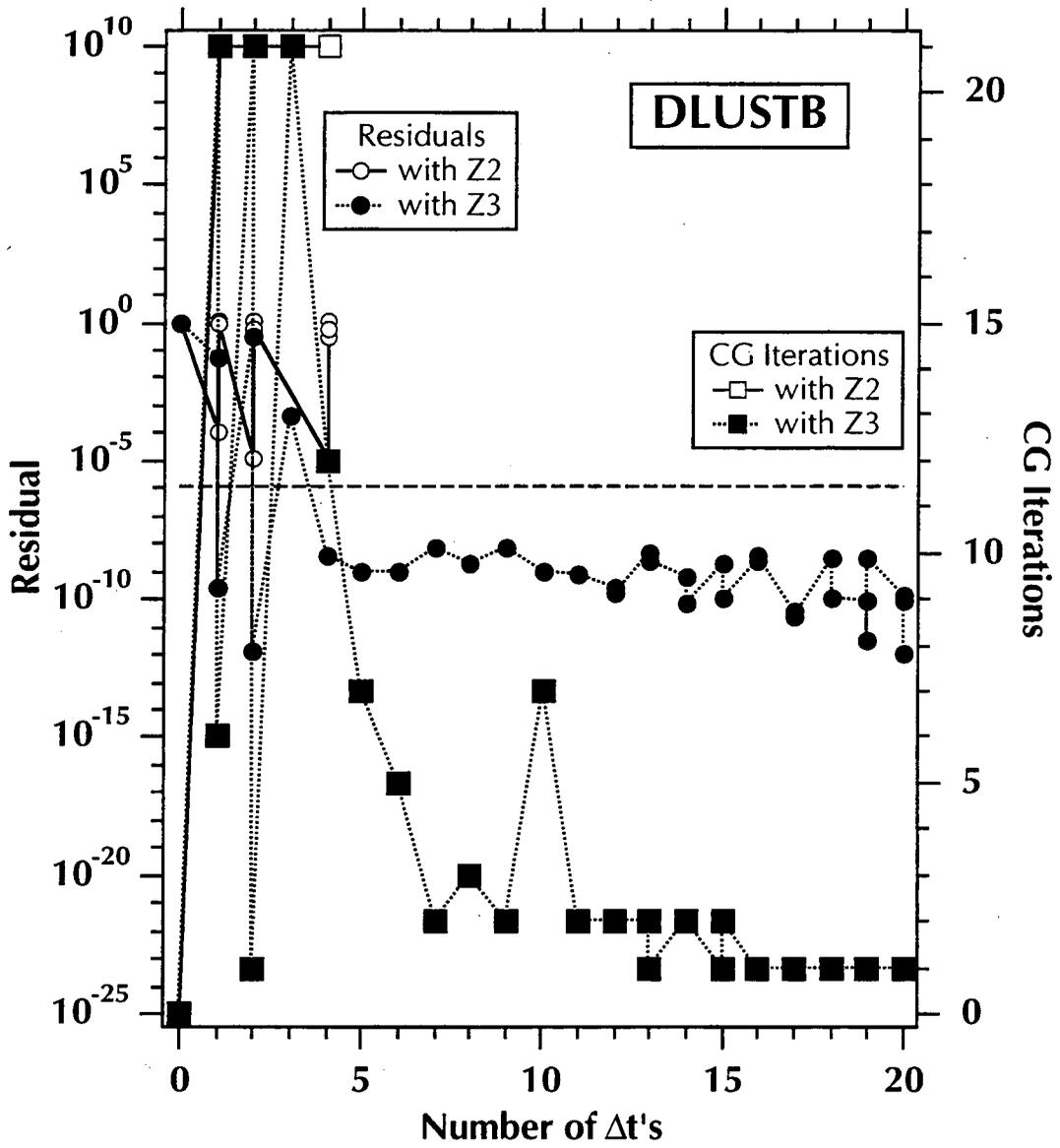


Fig. 7. DLUSTB performance in Test Problem 1a (injection into the first gridblock).

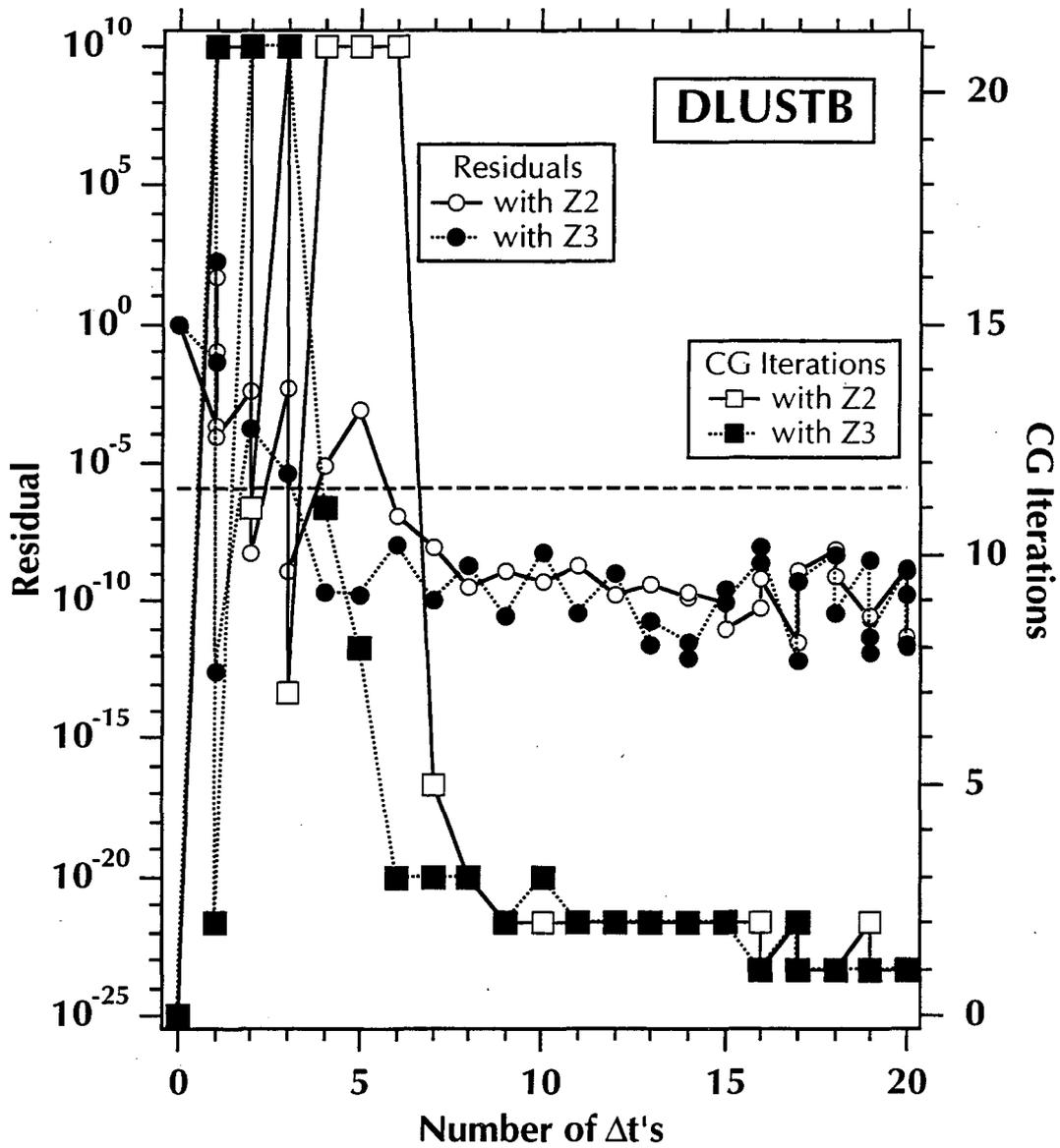


Fig. 8. DLUSTB performance in Test Problem 1a (injection into the middle gridblock).

-1.5896E-07	2.1387E-01	1.1921E+02	1.5219E-10																	
0	0	-1.1921E+02	0																	
-4.6528E-02	-2.4248E+06	0	1.8708E-04	2.0000E-04																
1.5219E-10	0	0	-1.5911E-07	2.1387E-01	1.1921E+02	1.5219E-10	0	0												
0	0	0	0	0	-1.1921E+02	0	0	0												
1.8708E-04	2.0000E-04	0	-4.6715E-02	-2.4248E+06	0	1.8708E-04	2.0000E-04	0												
			1.5219E-10	0	0	-1.5911E-07	2.1387E-01	1.1921E+02	1.5219E-10	0	0									
			0	0	0	0	0	-1.1921E+02	0	0	0									
			1.8708E-04	2.0000E-04	0	-4.6715E-02	-2.4248E+06	0	1.8708E-04	2.0000E-04	0									
			1.5219E-10	0	0	1.5219E-10	0	0	-1.5911E-07	2.1387E-01	1.1921E+02	1.5219E-10	0	0						
			0	0	0	0	0	0	0	0	-1.1921E+02	0	0	0						
			1.8708E-04	2.0000E-04	0	1.8708E-04	2.0000E-04	0	-4.6715E-02	-2.4248E+06	0	1.8708E-04	2.00000000E-04	0						
			1.5219E-10	0	0	1.5219E-10	0	0	0	0	0	0	0	-1.5896E-07	2.1387E-01	1.1921E+02	0	0	0	
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			1.8708E-04	2.0000E-04	0	1.8708E-04	2.0000E-04	0	-4.6528E-02	-2.4248E+06	0	-4.6528E-02	-2.4248E+06	0	0	0	0	0	0	0

Fig. 9. The Jacobian in Problem 1a with no Z-preprocessing at the first Newtonian iteration of the first timestep.

-1.5896E-07	2.1387E-01	1.1921E+02	1.5219E-10	0	0														
-1.5896E-07	2.1387E-01	0	1.5219E-10	0	0														
-4.6529E-02	-2.4248E+06	1.1921E+02	1.8708E-04	2.0000E-04	0														
1.5219E-10	0	0	-1.5911E-07	2.1387E-01	1.1921E+02	1.5219E-10	0	0											
1.5219E-10	0	0	-1.5911E-07	2.1387E-01	0	1.5219E-10	0	0											
1.8708E-04	2.0000E-04	0	-4.6715E-02	-2.4248E+06	1.1921E+02	1.8708E-04	2.0000E-04	0											
			1.5219E-10	0	0	-1.5911E-07	2.1387E-01	1.1921E+02	1.5219E-10	0	0								
			1.5219E-10	0	0	-1.5911E-07	2.1387E-01	0	1.5219E-10	0	0								
			1.8708E-04	2.0000E-04	0	-4.6715E-02	-2.4248E+06	1.1921E+02	1.8708E-04	2.0000E-04	0								
						1.5219E-10	0	0	-1.5911E-07	2.1387E-01	1.1921E+02	1.5219E-10	0	0					
						1.5219E-10	0	0	-1.5911E-07	2.1387E-01	0	1.5219E-10	0	0					
						1.8708E-04	2.0000E-04	0	-4.6715E-02	-2.4248E+06	1.1921E+02	1.8708E-04	2.0000E-04	0					
									1.5219E-10	0	0	-1.5896E-07	2.1387E-01	1.1921E+02	1.5219E-10	0	0		
									1.5219E-10	0	0	-1.5896E-07	2.1387E-01	0	1.5219E-10	0	0		
									1.8708E-04	2.0000E-04	0	-4.6529E-02	-2.4248E+06	1.1921E+02	1.8708E-04	2.0000E-04	0		
												1.5219E-10	0	0	-1.5896E-07	2.1387E-01	1.1921E+02		
												1.5219E-10	0	0	-1.5896E-07	2.1387E-01	0		
												1.8708E-04	2.0000E-04	0	-4.6529E-02	-2.4248E+06	1.1921E+02		

Fig. 10. The Jacobian in Problem 1a with Z2 preprocessing at the first Newtonian iteration of the first timestep.

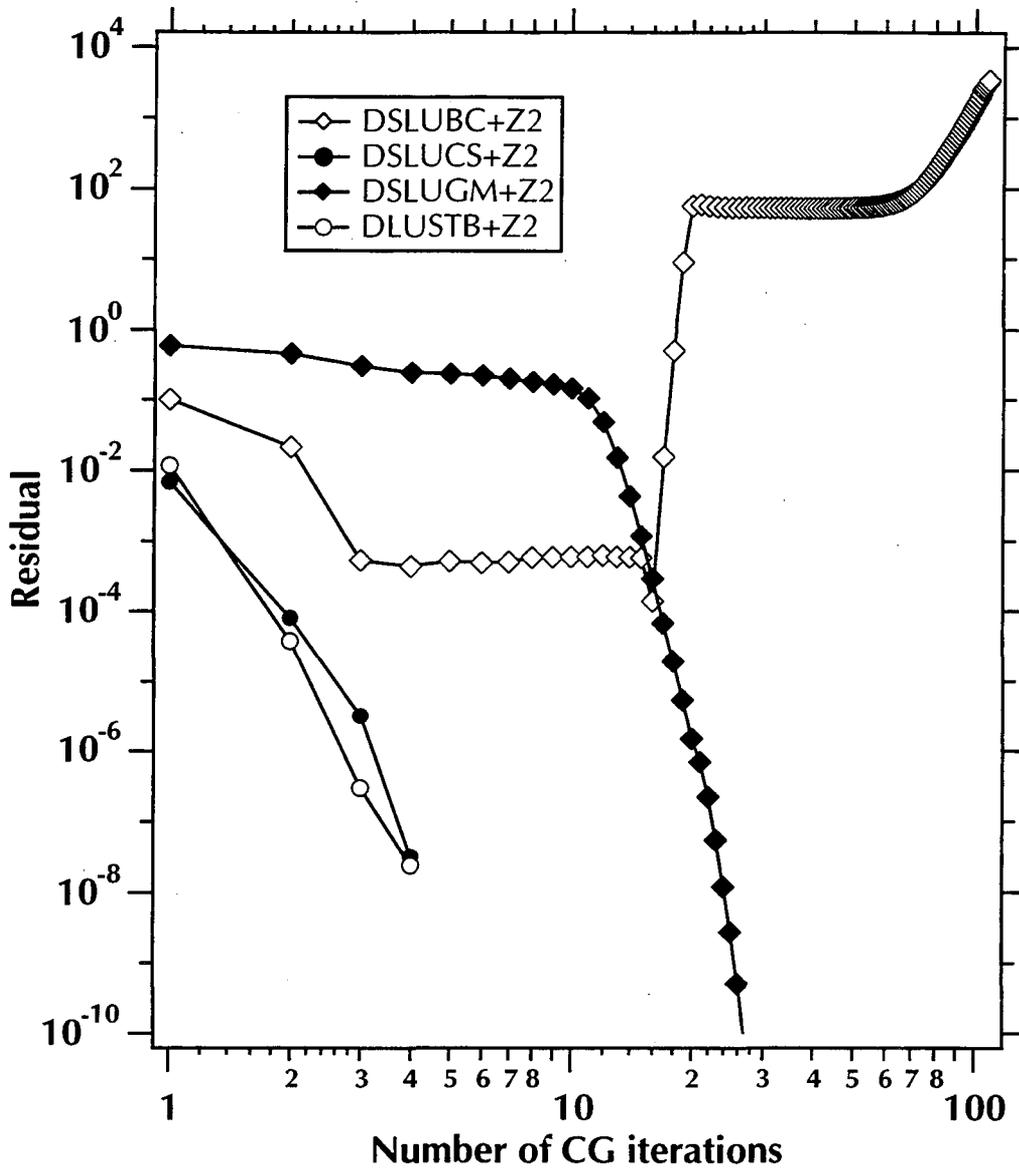


Fig. 12. PCG solvers with Z2 preprocessing in Test Problem 1b (1st NI of the 1st Dt).

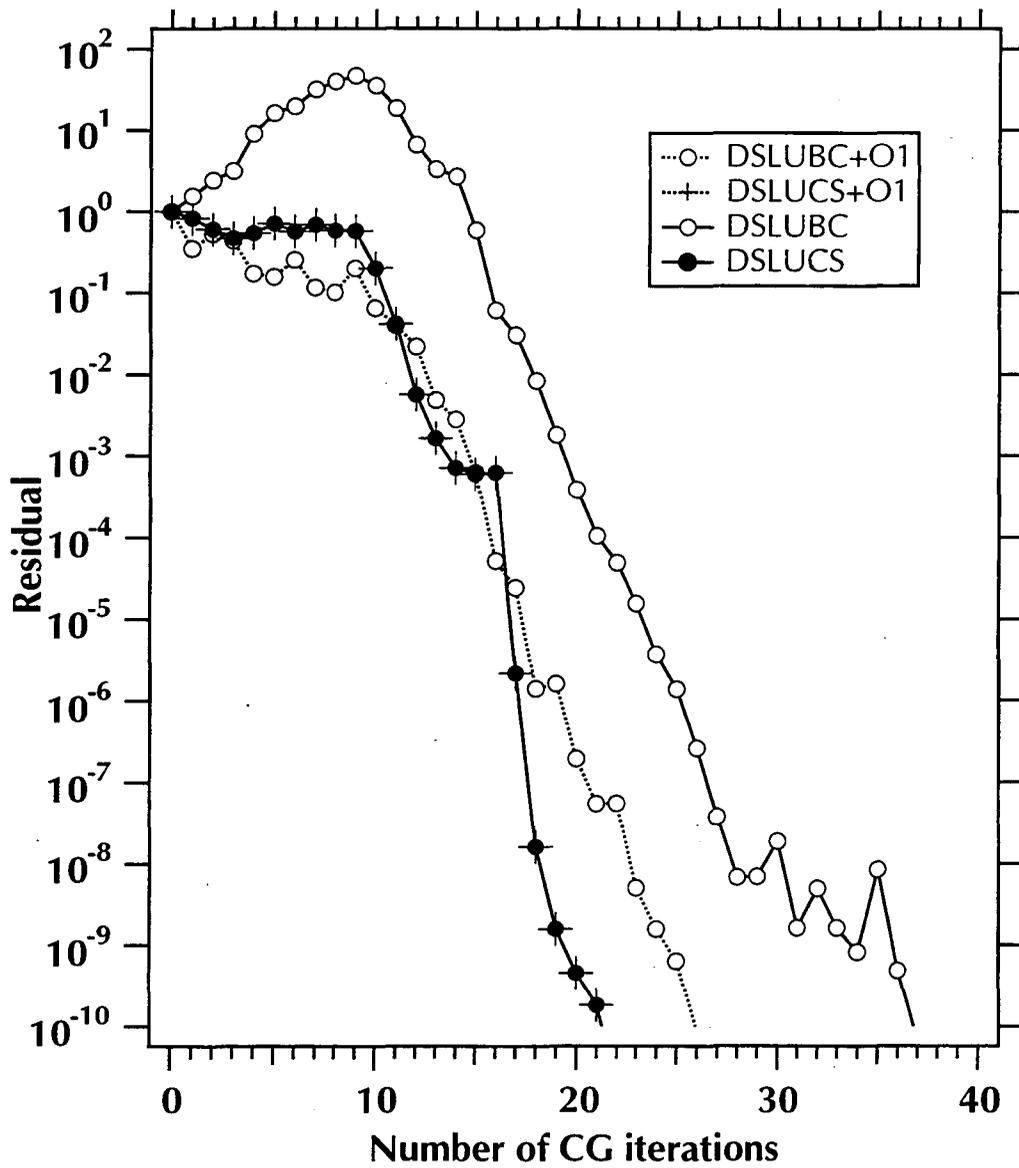


Fig. 13. DSLUBC and DSLUCS performance with and without O1 preprocessing in Test Problem 2 (1st NI of the 1st Dt).

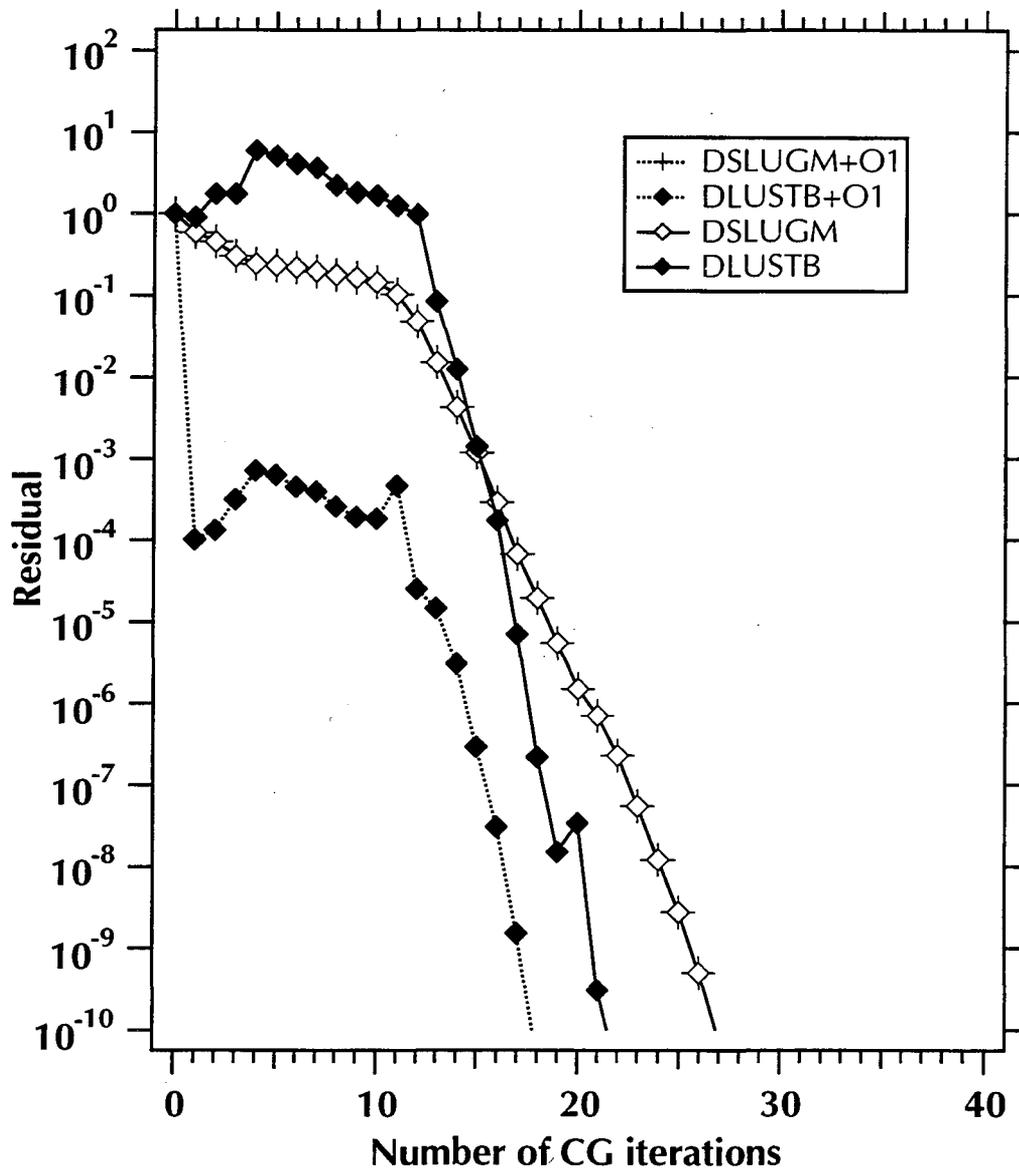


Fig. 14. DSLUGM and DLUSTB performance with and without O1 preprocessing in Test Problem 2 (1st NI of the 1st Dt).

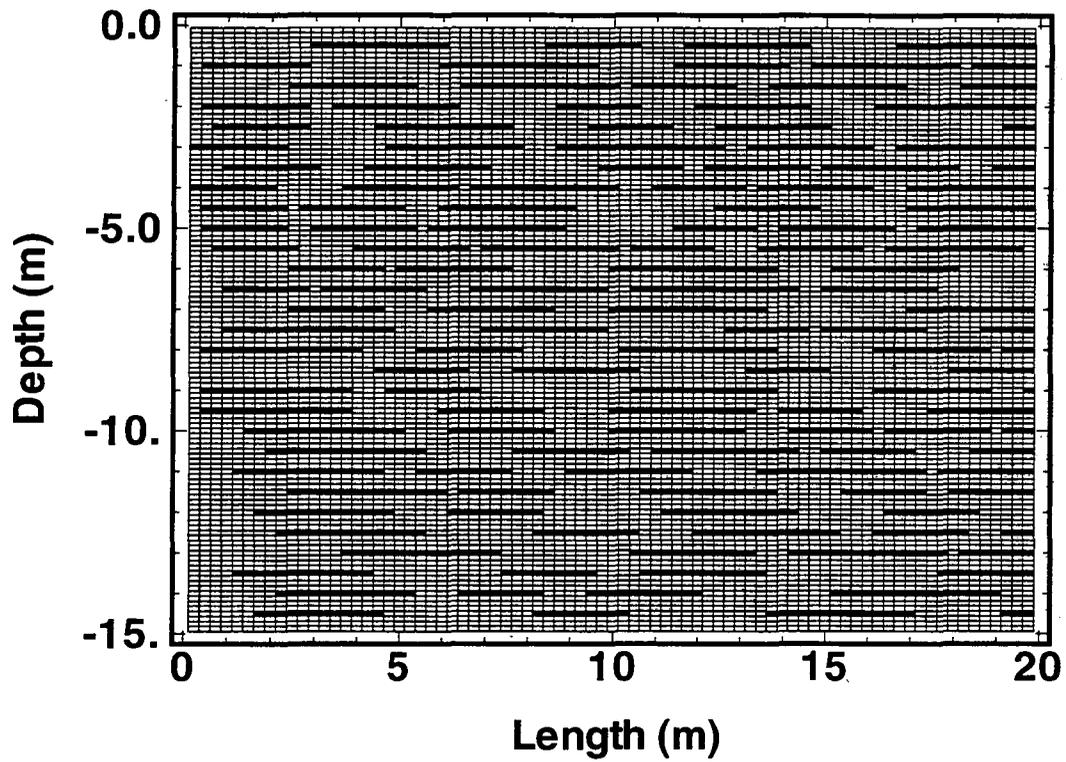


Fig. 15. The computational grid in Problem 4. The regions shown in black indicate impermeable obstacles.

APPENDIX

TOUGH2 is a general-purpose simulator for nonisothermal flows of NK fluid components distributed among NPH phases. Its modular structure was built on the recognition that the mass-and-energy balance equations for flow in permeable media have the same form, regardless of the nature and number of fluid components and phases present. The balance equations for component k ($k = \text{water, CO}_2, \text{NaCl, tracers, ...}$) are written in integral form for an arbitrary flow region V_n with surface area G_n as follows

$$\frac{d}{dt} \int_{V_n} M^k dV_n = \int_{\Gamma_n} \mathbf{F}^k \cdot \mathbf{n} d\Gamma_n + \int_{V_n} q^k dV_n \quad (\text{A.1})$$

Here M^k is the mass of component k per unit porous medium volume, \mathbf{F}^k is the mass flux of component k into V_n , \mathbf{n} is the inward unit normal vector, and q^k is the rate of mass generation of component k per unit volume. For the heat balance, M^k is the amount of energy (heat) per unit porous medium volume, \mathbf{F}^k is the heat flux, and q^k is the rate of heat generation per unit volume. The mass accumulation terms contain a sum over the phases b ($b = \text{g-gas, w-aqueous}$).

$$M^k = \phi \sum_{\beta} S_{\beta} \rho_{\beta} X_{\beta}^k \quad (\text{A.2})$$

ϕ denotes porosity, S_{β} is the saturation (pore volume fraction) occupied by phase β , ρ_{β} is the β phase density, and X_{β}^k is the mass fraction of component k in phase β . The heat accumulation term ($k = h$) includes contributions from both the solid and the fluid phases,

$$M^h = (1 - \phi) \rho_R C_R T + \phi \sum_{\beta} S_{\beta} \rho_{\beta} u_{\beta}, \quad (\text{A.3})$$

where ρ_R is the soil grain density, C_R is the heat capacity of the soil grains, T is the temperature, and u_{β} is the specific internal energy of phase β .

The mass flux \mathbf{F} is a sum over the fluxes in liquid and vapor phases, which are written as a multiphase version of Darcy's law, as follows ($\beta = \text{g, w}$).

$$\mathbf{F}_{\beta} = -\mathbf{k} \frac{k_{r\beta}}{\mu_{\beta}} \rho_{\beta} (\nabla P_{\beta} - \rho_{\beta} \mathbf{g}) \quad (\text{A.4})$$

\mathbf{k} denotes the permeability tensor, k_r is relative permeability, μ is viscosity, P_{β} is the pressure in phase β , and \mathbf{g} is acceleration of gravity. Heat flux contains conductive and convective components:

$$\mathbf{F}_h = -K \nabla T + (h_w \mathbf{F}_w + h_g \mathbf{F}_g) \quad (\text{A.5})$$

with K the thermal conductivity of the rock-fluid mixture, and h the specific enthalpy. Thermophysical properties of water substance are calculated, within experimental accuracy, from steam table equations given by the International Formulation Committee [IFC, 1967]. Empirical correlations are used for thermophysical properties of fluid mixtures that contain non-condensable gases and dissolved solids [Battistelli *et al.*, 1997].

For numerical solution, the continuum equations (A.1) are discretized in space and time. Space discretization is made with the “Integral Finite Difference” method [IFD; 1967; Narasimhan and Witherspoon, 1976]. This method permits irregularly shaped grid blocks in 1, 2, and 3 dimensions. It includes double porosity, dual permeability, and multiple interacting continua (MINC) formulations for fractured-porous media as special cases. For grid systems of regular blocks referred to a fixed global coordinate system, the IFD reduces to conventional finite differences. Time is discretized fully implicitly as a first-order (backward) finite difference.

Discretization results in a system of coupled non-linear algebraic equations. These are cast in residual form and solved simultaneously by means of Newton-Raphson iteration. Iteration is continued until all residuals are reduced below a user-specified convergence tolerance. A choice of different algorithms is available for solving the linear equations arising at each iteration step.

**ERNEST ORLANDO LAWRENCE BERKELEY NATIONAL LABORATORY
ONE CYCLOTRON ROAD | BERKELEY, CALIFORNIA 94720**