JOURNAL OF COMPUTATIONAL PHYSICS 91, 298–309 (1990)

Preconditioned Biconjugate Gradient Methods for Numerical Reservoir Simulation

P. JOLY

Laboratoire d'Analyse Numérique, Tour 55-65 5ème étage,
Université Pierre et Marie Curie 75252 Paris Cedex 05, France

AND

R. EYMARD

S.N.E.A.P./S.M.G., Tour Elf-Aquitaine,
2 Place de la Coupole, 92078 Paris-La Defense Cedex 45, France

Received March 18, 1987; revised September 28, 1989

This paper describes numerical experiments on solving linear systems of equations that arise in reservoir simulations. The well-known conjugate-gradient methods Orthomin and Gmres are compared to the biconjugate-gradient method and to an accelerated version called the conjugate-gradient squared method. An incomplete factorization technique based on the level of fill-in idea is used, with investigations to find the appropriate level. Finally, the influence of a reordering method on the convergence rate is tested. © 1990 Academic Press, Inc.

1. INTRODUCTION

In recent years the use of conjugate-gradient type methods to solve linear systems of equations has been developed and some improvements have been made in order to construct fast and robust algorithms. The acceleration of iterative methods devoted to ill-conditioned systems seems to be promising. In this paper the conjugate gradient squared method introduced by Sonneveld is tested on reservoir simulation problems.

Though the natural way to solve a linear system of equations progresses in a reordering phase, an incomplete factorization phase, and a solving phase, we present our results inversely in order to preserve the relative influence of each phase on the final computing time.

2. ITERATIVE METHODS

Orthomin [3] and Gmres [12] are very popular methods in reservoir simulation, so we summarize their main properties in a few words. Both of them deal with

0021-9991/90 $3.00

the minimization of the residual norm $\|r\| = \|b - Ax\|$ on the finite dimension space E^k spanned by the Krylov sequence r^0, Ar^0, ..., $A^k r^0$, where r^0 is the initial residual. A basis of E^k is generated iteratively by adding a new direction orthogonal to all the previous ones. With appropriate hypothesis, convergence is obtained after n iterations, where n is the number of unknowns of the linear system. As the process needs the storage of the complete directions set, it is periodically restarted after m steps (m to be chosen) to save storage. Although they both work identically, Orthomin and Gmres differ on the orthogonalization procedure: Orthomin uses the scalar product associated to the residual norm (Ad, Ad'), while Gmres uses the natural scalar product (d, d'). The first method needs the storage of the m directions d used and the m associated products Ad, but gives the result without an extra computation cost. The second method uses only the m directions d, but on the other hand, the minimization step requieres to solve a least squares problem.

The biconjugate-gradient method (Bcg) [2] is an extension of the standard preconditioned conjugate gradient method (Cg) to a symmetric system of double size

$$\begin{bmatrix} 0 & A \\ A^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix},$$

using preconditioner

$$\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$$

see [8]. Since the matrix is indefinite, the minimization argument of Cg becomes ineffective and the residual norm may increase during iterations. So the algorithm is based rather on an orthogonalization process of the residuals, than on a minimization procedure of the residual norm. Furthermore, the adjoint system $A^T y = c$ doubles the computational cost per iteration. In spite of these defects, the method works well, and because it deals with a symmetric system, it does not need the storage of all the previous directions.

The usual form of Beg is

— Choose an arbitrary x^0

— Set

$$r^0 = b - Ax^0$$

and

$$s^0 = q^0 = p^0 = r^0$$

— until convergence Do

$$\alpha^k = (s^k, r^k)/(q^k, Ap^k)$$
$$x^{k+1} = x^k + \alpha^k p^k$$
$$r^{k+1} = r^k - \alpha^k A p^k$$
$$s^{k+1} = s^k - \alpha^k A^T q^k$$
$$\beta^{k+1} = (s^{k+1}, r^{k+1})/(s^k, r^k)$$
$$p^{k+1} = r^{k+1} + \beta^{k+1} p^k$$
$$q^{k+1} = s^{k+1} + \beta^{k+1} q^k$$

In this formulation the adjoint vector y^k has been eliminated and the associated starting residual $s^0 = c - A^T y^0$ is set equal to $r^0 = b - Ax^0$. The vectors r^k, p^k, s^k, and q^k satisfy

$$(s^k, r^l) = 0 \qquad \text{for} \quad k \neq l$$
$$(q^k, Ap^l) = 0 \qquad \text{for} \quad k \neq l$$

with $(\cdot, \cdot)$ being the natural scalar product in R^n.

Furthermore, we have

$$r^k = \phi_k(A)\, r^0$$
$$s^k = \phi_k(A^T)\, r^0$$
$$p^k = \theta_k(A)\, r^0$$
$$q^k = \theta_k(A^T)\, r^0,$$

where ϕ_k and θ_k are polynomials of degree k.

In order to accelerate the biconjugate gradient method, Sonneveld [14] has constructed an algorithm for which the residual after k iterations is $\phi_k^2(A)\, r^0$ instead of $\phi_k(A)\, r^0$. So if Bcg converges, $\phi_k(A)$ represents a contraction for large values of k, and then $\phi_k^2(A)$ represents a contraction of smaller norm, leading to a faster convergence.

By squaring the induction relations

$$\phi_{k+1}(A) = \phi_k(A) - \alpha^k A \theta_k(A)$$
$$\theta_{k+1}(A) = \phi_{k+1}(A) + \beta^{k+1} \theta_k(A),$$

he obtains the so-called conjugate gradient squared method (Cgs), that we can summarize in the following way:

— Choose an arbitrary x^0

— Set

$$r^0 = b - Ax^0$$

and

$$q^0 = p^0 = r^0$$

—until convergence Do

$$\begin{aligned}
\alpha^k &= (r^0, r^k)/(r^0, Aq^k) \\
x^{k+1} &= x^k + \alpha^k(p^k + p^k - \alpha^k Aq^k) \\
r^{k+1} &= r^k - \alpha^k A(p^k + p^k - \alpha^k Aq^k) \\
\beta^{k+1} &= (r^0, r^{k+1})/(r^0, r^k) \\
p^{k+1} &= r^{k+1} + \beta^{k+1}(p^k - \alpha^k Aq^k) \\
q^{k+1} &= p^{k+1} + \beta^{k+1}(\beta^{k+1} q^k + p^k - \alpha^k Aq^k).
\end{aligned}$$

The computational work per iteration is about the same as in the initial formulation (Bcg), but multiplications by A^T are avoided, so a vectorized version of Cgs refers only to standard techniques. The vector $v = p^k - \alpha^k Aq^k$, which appears many times in each iteration, has to be stored to save computational time.

3. Incomplete Factorization

The use of a preconditioner is now a classical way to accelerate the convergence of iterative methods, a lot of papers have improved this technique: [1, 11] and many others...

In this paper, we use the incomplete factorization developed by [7, 9, 4], which works in two steps. The first step is a symbolic incomplete factorization to determine the non-zero locations in the $L.U$ factors through the level of fill-in idea. The second step computes the coefficients of the associated incomplete factors.

We summarize the symbolize factorization in defining for each fill-in term a level by an induction argument

— the level is set equal to zero for all non-zero coefficients in A.

— the fill-in terms which arise due to the elimination of a level k term are set to level $k+1$.

Clearly after a few steps of this process, we obtain the complete factorization of A, for the deepest level, say $l(A)$. For incomplete factorizations some level l, $0 \leqslant l < l(A)$, has to be chosen in order to accelerate the iterative methods. This method, which deals with any matrix, is a generalization of the incomplete factorization described by [6] for matrices of very regular structure arising in finite difference regular grids.

4. Reordering

As $l(A)$ depends on the ordering of the unknowns a permutation which produces a minimum fill-in or a minimum level $l(A)$ must be found. The minimum degree algorithm [5] and the nested dissection algorithm [5] are well known to provide less fill-in than the natural ordering (see for reservoir simulation improvements [7, 9], for example). Here we used first the natural ordering, then the D4 ordering (equivalent to the red-and-black ordering), which is well adapted to regular finite difference grids.

5. Test Problems

The test problems arise from a three-dimensional reservoir simulation model using a seven points finite difference scheme on a $n* ny * nz$ grid with nc equations and unknowns per grid block. For a complete description of the physics, we refer to [10]. Table I gives the values of the different parameters. These problems are often used as benchmarks to test the performance of iterative methods, for example, [13] presents numerical experiments of the Itpack code on vector computers to solve these linear systems.

6. Numerical Results

Numerical experiments have been performed on a NAS 9080 computer using double precision variables (64 bits per real*8 word). Convergence is obtained when

TABLE I

Problem	nx	ny	nz	nc	n
1	10	10	10	1	1000
2	6	6	5	6	1080
3	35	11	13	1	5005
4	16	23	3	1	1104
5	16	23	3	3	3312

TABLE II

Results for Problem 1 with $ILU(0)$ Preconditioner

Algorithm		Number of iterations	Time
Orthomin	5	42	0.6696
	10	46	0.9105
Gmres	5	15[a]	1.1558
	10	6[a]	0.9709
Bcg		41	0.8449
Cgs		34	0.7395

[a] Number of least squares problems to solve.

$\|r^k\|/\|r^0\|$ is less than 10^{-6} and the number of iterations less than $n/5$, where $n = nx * ny * nz * nc$ is the total number of unknowns. The initial guess $x^0 = 0$ is fixed for all the runs presented. Execution times, including factorization and iterations times, are given in seconds.

6.1. *Iterative Methods*

We first compare the four algorithms using the same preconditioner on the five test problems with natural ordering: the incomplete factorization of level 0 is used.

Orthomin and Gmres are used with 5 directions and 10 directions. Results are given in Tables II to VI. We note that an expected breakdown of Bcg does not occur, thus this algorithm converges faster than both Orthomin and Gmres on Problems 3, 4, and 5. The accelerated version Cgs gives good convergence rate and total time cost (in fact, the best ones if excluding the first problem).

TABLE III

Results for Problem 2 with $ILU(0)$ Preconditioner

Algorithm		Number of iterations	Time
Orthomin	5	20	0.9501
	10	20	0.9797
Gmres	5	4[a]	0.9570
	10	2[a]	0.9315
Bcg		19	1.4468
Cgs		10	0.7715

[a] Number of least squares problems to solve.

TABLE IV

Results for Problem 3 with *ILU*(0) Preconditioner

Algorithm		Number of iterations	Time
Orthomin	5	150	13.7952
	10	151	17.6515
Gmres	5	240[a]	21.8188
	10	180[a]	15.3783
Bcg		97	12.1372
Cgs		66	8.0467

[a] Number of least squares problems to solve.

TABLE V

Results for Problem 4 with *ILU*(0) Preconditioner

Algorithm		Number of iterations	Time
Orthomin	5	72	1.1728
	10	52	1.1085
Gmres	5	23[a]	1.6776
	10	9[a]	1.4307
Bcg		32	0.8000
Cgs		23	0.4979

[a] Number of least squares problems to solve.

TABLE VI

Results for Problem 5 with *ILU*(0) Preconditioner

Algorithm		Number of iterations	Time
Orthomin	5	Stagnation	—
	10	Stagnation	—
Gmres	5	33[a]	8.7848
	10	10[a]	6.6467
Bcg		41	4.2082
Cgs		25	2.4743

[a] Number of least squares problems to solve.

TABLE VII

Incomplete Factorization (Natural Ordering)

Level	NZ	Number of Iterations	Residual Norm	Factorization Time	Iteration Time	Total Time
0	2750	34	10^{-6}	.0198	.7395	.7593
1	4436	16	10^{-7}	.0517	.3589	.4106
2	6814	11	10^{-6}	.0804	.3174	.3978
3	11678	8	10^{-6}	.1594	.3013	.4607
4	20408	7	10^{-8}	.3703	.3727	.7430
5	32794	4	10^{-7}	.8248	.3031	1.1279
6	42636	3	10^{-7}	1.3403	.2902	1.6305
7	46474	2	10^{-8}	1.6508	.2223	1.8731
8	47494	1	10^{-6}	1.8678	.1541	2.0219
9	47776	1	10^{-8}	1.8661	.1579	2.0240
10	47900	1	10^{-9}	1.8662	.1474	2.0136
11	47952	1	10^{-9}	1.8231	.1468	1.9691
12	47968	1	10^{-9}	1.8521	.1487	2.0008
13	47970	1	10^{-9}	1.8192	.1451	1.9642

TABLE VIII

Incomplete Factorization (D4 Ordering)

Level	NZ	Number of Iterations	Residual Norm	Factorization Time	Iteration Time	Total Time
0	2750	54	10^{-6}	.0226	1.1732	1.1958
1	5862	11	10^{-7}	.0693	.2832	.3525
2	8208	7	10^{-6}	.1041	.2318	.3359
3	13012	4	10^{-6}	.2156	.1726	.3882
4	20978	3	10^{-6}	.5159	.1701	.6860
5	27316	2	10^{-7}	.9388	.1505	1.0891
6	28862	1	10^{-8}	1.1527	.1001	1.2528
7	28938	1	10^{-10}	1.1749	.0980	1.2729

6.2. *Incomplete Factorization*

The incomplete factorization using the level idea has been used, we give in Table VII the results for the Cgs method on Problem 1 (natural ordering), for which the complete series from level 0 to level $l(A)$ can be computed. NZ is the number of non-zero coefficients in the $L.U$ factors. We observe that the number of iteration steps decreases as the level of fill-in increases, but the gain of time spent in the iterations is progressively overcome by the increasing factorization time. Note that from level 8 to 13 (13 corresponding to the complete factorization), just one iteration is sufficient to satisfy the convergence criterion, but the residual norm continues to decrease as the level increases. The best total computing time is obtained for level 2.

The same experiment on Problems 2 to 5, leads to an analogous behaviour of the Cgs method. For these problems, the optimal level in regard with the computing time is 1.

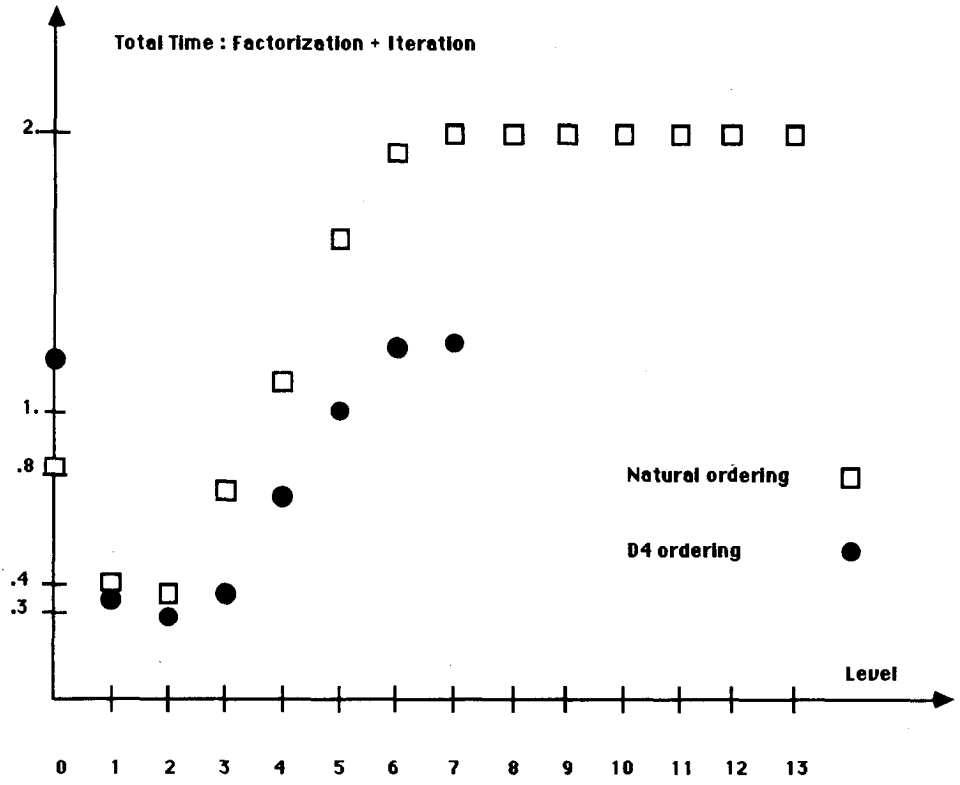

FIG. 1. Total time versus level, comparison of natural, and D4 orderings.

6.3. *Reordering*

Using the D4 (or red-and-black) ordering, we solve again the first problem with the Cgs algorithm. We obtain the same behaviour of the total computing time versus the level of fill-in, but the value of $l(A)$ is then 7, and the best total time corresponds to level 2 (see Table VIII). Figure 1 summarizes the comparison of the results of Tables VII and VIII. Problems 2 to 5 provide the same results. So we get some performance gain by using a reordering algorihtm, and a level of fill-in >0.

Another point of interest is that the level idea leads to significant improvement of the convergence. Figure 2 summarizes the evolution of the residual norm for different values of the level (Problem 1, Cgs method, D4 ordering). For level 0, we observe drastic oscillations which are the natural convergence behaviour of the Cgs method (we recall that this algorithm does not minimize the residual norm). For level 1, oscillations have disappeared and the convergence is monotone. So we see that the use of an incomplete factorization can enhance the properties of an iterative method.

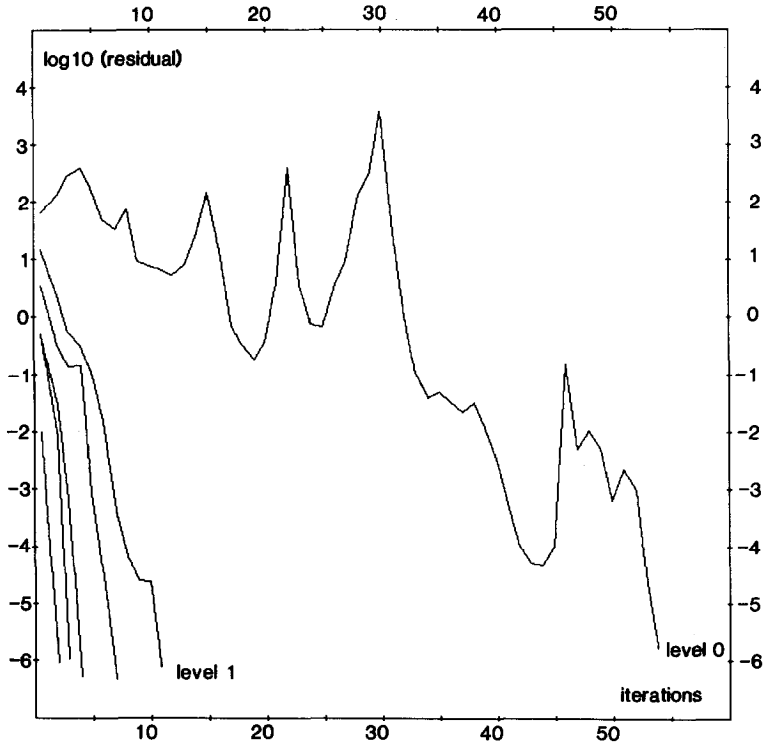

FIG. 2. Cgs convergence versus level.

7. Conclusion

(1) We have compared four iterative methods to solve reservoir simulation problems, which are considered to be very difficult. The Cgs algorithm gives the best total execution time and never breaks down in our experience.

(2) We have used the incomplete factorization with the level idea; we obtain the bests results for a value of 1 or 2.

(3) The use of the D4 reordering method yields an expected acceleration of the convergence rate and some gain on the total computing time, if used with a level > 0.

References

1. T. Dupont, R. P. Kendall, and H. H. Rachford, *SIAM J. Numer. Anal.* **3** (1968).
2. R. Fletcher, in *Proceedings, Conference in Numerical Analysis, Dundee, Eire*, 1975 edited by G. A. Watson, Lecture Notes in Math., Vol. 506 (Springer-Verlag, Berlin, 1976).
3. P. K. W. Vinsome, in *Proceedings, Fourth Symposium on Numerical Simulation of Reservoir Performance of the Society of Petroleum Engineers of AIMe, Los Angeles*, 1976.
4. J. W. Watts, *Soc. Petrol. Eng. J.* **21** (1981).
5. A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems* (Prentice–Hall, Englewood Cliffs, NJ, 1981).
6. J. A. Meijerink and H. K. Van der Vorst, *J. Comput. Phys.* **44** (1981).
7. A. Behie and P. A. Forsyth, Jr., in *Proceedings, Seventh Symposium on Reservoir Simulation of the Society of Petroleum Engineers of AIME, San Francisco*, 1983.
8. K. C. Jea and D. M. Young, *Linear Algebra Appl.* **52/53** (1983).
9. J. R. Wallis, in *Proceedings, Seventh Symposium on Numerical Simulation of Reservoir Performance of the Society of Petroleum Engineers of AIME, San Francisco*, 1983.
10. A. H. Sherman, "Linear Algebra for Reservoir Simulation. Comparison Study of Numerical Algorithms," J. S. Nolen Associates, Inc., 1984 (unpublished).
11. P. Concus, G. Golub, and G. Meurant, *SIAM J. Sci. Stat. Comput.* **6** (1985).
12. Y. Saad and M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **7** (1986).
13. T. C. Oppe and D. R. Kincaid, *Commun. Appl. Numer. Meth.* **3** (1987).
14. P. Sonneveld, *SIAM J. Sci. Stat. Comput.* **10** (1989).