INT. J. NUM. METH. HEAT FLUID FLOW, VOL. 5, 213–223 (1995)

Invited extended paper from Eighth International Conference on Numerical Methods in Laminar and Turbulent Flow, which was held July 1993 at Swansea, U.K. The guest editor is Prof. Samuel H. Makarios of Ain Shams University, Cairo, Egypt.

AN EVALUATION OF ITERATIVE METHODS IN THE SOLUTION OF A CONVECTION–DIFFUSION PROBLEM

Y. T. FENG†[1], G. J. HUANG‡, D. R. J. OWEN† AND D. PERIĆ†

†*Department of Civil Engineering, University College of Swansea, UK*
‡*Rockfield Software Ltd., Swansea, UK*

ABSTRACT

In this paper we investigate the performance of CGS, BCGSTAB and GMRES with ILU preconditioner for solving convection-diffusion problems. Numerical experiments indicate that BCGSTAB appears to be an efficient and stable method. CGS sometimes suffers from severe numerical instability. GMRES shows a higher suitability and stability but the overall convergence rate may be lower.

KEY WORDS Iterative methods GMRES The conjugate gradient squared method The stabilized biconjugate gradient method Preconditioning Convection-diffusion problem

INTRODUCTION

The finite difference or finite element discretisation of partial differential equations in many application areas, such as solid mechanics, CFD, meteorology, petroleum reservoir, neutron diffusion, etc., give rise to the following large sparse algebraic equations:

$$Ax = b \tag{1}$$

where A is a $n \times n$ nonsingular matrix, b is the known vector, and x is the solution to be found.

In general, there are two kinds of solvers for linear equations: direct and iterative. For large scale equations the number of unknowns may reach over 10000 so that the sparsity and the structure of the algebraic system have to be exploited while adopting any solution method. Although these may be incorporated within Gauss elimination-based direct methods, and sparse matrix techniques, as well as other acceleration schemes[7,8], the computational efficiency is low and the storage requirements are still extremely high, even for current supercomputers. It can be argued, however, that more powerful computers with larger space capabilities are becoming available, but at the same time there is an even increasing demand for the solution of more complicated and larger problems. In these cases, a solution can be obtained by iterative methods with reasonable computational costs in terms of time and storage. For this reason, iterative methods have been drawing increasing attention in recent years.

Iterative methods offer compelling promise over direct methods in regard to the following aspects[11]:

1 Much easier to exploit system sparsity and thus the computer memory required may be substantially reduced, especially for large problems

[1] On leave from Beijing Agricultural Engineering University, P.R. China.

0961–5539/95/030213–11$2.00

Received November 1993
Revised April 1994

2 Relatively simple in implementation
3 These methods may prove more conducive to effective implementation in emerging computer systems with vector and parallel processing facilities
4 Accuracy may be more controllable and thus computing time may be saved in the cases where only a lower level accuracy is required
5 They may lead to a more synergistic incorporation in the solution of evolving non-linear problems

However, iterative methods suffer from the following shortcomings:

1 Convergence rates may be very low; or more severely, convergence can not always be guaranteed
2 They can not efficiently process multi-right hand sides of problems
3 Performance of many iterative algorithms are problem-dependent

These severe disadvantages make iterative methods incompetitive in comparison to direct methods in solving small and medium scale linear systems.

For symmetric positive definite (SPD) problems it seems quite clear that the conjugate gradient method (CG) is a very efficient solver for many practical problems, but this is not the case for nonsymmetric systems.

A number of iterative methods have been proposed that are applicable to the nonsymmetric cases[1,6,9,14,17,20]. At the moment, however, there are no theoretical means available to estimate the performances of these methods in practice, and makes it very difficult for users to choose the best method to solve their own problems; and in particular to select a robust and efficient iterative solvers to incorporate in a commercial general-purpose FE software.

Moreover, the convergence of iterative methods depends to a great extent on the preconditioning and 'good' preconditioners suitable for a large variety of practical problems are rare.

Consequently, numerical experiments are necessarily used to obtain insight and confidence in the efficiency and robustness of a particular method or to obtain knowledge of the most appropriate method for solving a particular problem. But on the other hand, one should carefully choose the test problems, to try and make them typical of a wider class of problems and to take into account the nature of the problems. Otherwise unjust or even totally incorrect conclusions may be drawn.

For convection-diffusion type problems, several numerical investigations have been carried out. Sonneveld[16] found the conjugate gradient squared method (CGS)[17] with incomplete Cholesky decomposition (ILU)[12] and block ILU (BLU)[3] preconditioning to be more efficient than other nonsymmetric CG methods. Whereas Axelsson *et al.*[4] showed that a general least squares conjugate gradient method (CG-LS)[2] coupled with BLU methods, can work exceptionally well and that the number of iterations is sometimes significantly less than that of the corresponding nearly symmetric problem. Recently, however, CGS is found to have an irregular convergence behaviour and a 'stabilized' variant of it, called BCGSTAB, has been developed by Van der Vorst[19]. In addition, the GMRES approach proposed by Saad and Schultz[14] has been successully applied to various fields[10,15,18].

In this paper, we shall extensively investigate the performance of CGS, BCGSTAB and GMRES for solving 2-D convection-diffusion problems by varying the value of the diffusion coefficients from zero to a very large value in order to investigate the behaviour of these methods over a range of conditions. The preconditioner used is a point-wise incomplete Cholesky decomposition (ILU) with no fill-in scheme. It is shown that the BCGSTAB method exhibits a good numerical stability and efficiency, while the CGS method does have an irregular convergence characteristics although it sometimes shows a better convergence. But both approaches are superior to GMRES in terms of CPU time and storage costs in the case of the diffusion term being significant. In convection dominated cases, however, CGS and BCGSTAB may fail to converge whereas GMRES may still achieve a solution. Meanwhile, we also notice that the ILU scheme may break down for almost pure convection problems.

In the following section the convection-diffusion equation is recalled. The iterative methods and ILU preconditioner which will be tested are briefly discussed and numerical experiments and remarks are also presented.

CONVECTION-DIFFUSION PROBLEM

Consider the following convection diffusion equation:

$$\mathbf{v}\cdot\nabla u(x)-\nabla\cdot(\mathbf{c}\cdot\nabla u(x))=f(x) \qquad x\in\Omega \tag{2}$$

where $\mathbf{v}$ is the velocity vector; Ω is the fluid domain with boundary Γ; $\mathbf{c}$ is a vector of diffusion coefficients. In the 2-D case, $\mathbf{c}=\{c_x, c_y\}^T$, in which c_x, c_y are diffusion coefficients in x and y directions respectively. Let $\mathbf{n}$ (x) be the outward unit normal on the boundary Γ which is partitioned into three sets:

$$\Gamma_i=\{x\in\Gamma|\mathbf{v}(x)\cdot\mathbf{n}<0\}$$
$$\Gamma_c=\{x\in\Gamma|\mathbf{v}(x)\cdot\mathbf{n}=0\}$$
$$\Gamma_o=\{x\in\Gamma|\mathbf{v}(x)\cdot\mathbf{n}>0\}$$

corresponding to the inflow, characteristic and outflow boundaries respectively. Dirichelet boundary condition $u(x)=1$ is imposed on the inflow portion of Γ_i, while no boundary conditions are specified on the characteristic and outflow boundaries. Moreover, we set the source term $f(x)=0$. In such a case it is easy to prove that:

$$u(x)=1$$

will be the solution of (2).

The solution may be obtained by Petrov-Galerkin finite element methods, and the corresponding matrix A can be highly nonsymmetric, unless the diffusion term is dominant. In order to indicate the degree of nonsymmetry of matrix A, we define a factor nf, called the *nonsymmetric factor*, as follows:

$$nf=\frac{\|A-A^T\|_F}{\|A+A^T\|_F}$$

where A^T is the transpose of A; $\|A\|_F$ denotes the Frobenous norm of A, i.e.

$$\|A\|_F=\left(\sum_{i,j}|a_{ij}|^2\right)^{1/2}$$

It is obvious that nf will be zero if $\mathbf{v}=\mathbf{0}$ and nf reaches its maximum value when $\mathbf{c}=\mathbf{0}$. The former case corresponds to a pure diffusion problem (i.e. a SPD problem) whereas the latter is a pure convection problem.

ITERATIVE METHODS FOR NONSYMMETRIC PROBLEM

A large number of iterative methods have been developed for solving nonsymmetric problems in the last 20 years. These methods can be broadly classified into two categories: generalized conjugate gradient type algorithms and Galerkin-Krylov type approaches.

In general, there are three main approaches for the generalization of CG for handling nonsymmetric cases. One is based on the transformation of the nonsymmetric problem into a SPD problem, and then solving the resulting system, named the normalised equations, with CG. This method is termed CGN. The CGN approach is not always applicable in practice, because the condition number of the resulting system may be extremely large, and consequently it may require a large number of iterations to converge even for small problems. Another approach is

to choose an auxiliary matrix Z such that ZA becomes positive real (PR) or equivalently has a symmetric part which is positive definite. The *generalized conjugate residual method* (GCR)[5] and the *idealized generalized conjugate-gradient method* (IGCG) with three equivalent forms referred as ORTHODIR, ORTHOMIN, and ORTHORES[20] respectively, belong to this class. These methods preserve the convergence properties of CG so that the iterative process converges to the true solution after a maximum n steps in exact arithmetic. However, all previous search-directions must be kept since they are utilized to orthogonalize with the current residual vector to obtain a new search-direction. The simplified versions of the IGCG methods[9] reduce the number of previous search-directions retained to the last one or two. The *biconjugate gradient method* (BCG), first proposed by Fletcher[6], is another generalized CG type method which re-interprets the CG algorithm by using a recurrence formula so that full orthogonalization is not necessary. The BCG method also fails to converge or breaks down in many cases of practical interest. A recent extension of BCG, *the conjugate gradient squared method* (CGS)[17], however, is found to be more efficient that BCG, but suffers from numerical instability. Its stabilized version, BCGSTAB has therefore been put forward[19] to amend this shortcoming.

GMRES, *generalized minimum residual method*, is a typical Galerkin-Krylov type method. It is a residual minimization method which is basically arrived at by the introduction of an optimality property into the Arnoldi algorithm which supplies a stable way to generate Krylov subspace. Like GCR and IGCG, it utilizes all previous iteration vectors at the current iterative step.

These features cause some practical difficulties in the implementation of GMRES as well as GCR and IGCG. As the number of iterations increases, the number of vectors stored also increases. In addition, the number of multiplications performed is proportional to the square of the number of iterations. The methods thus becomes expensive in terms of both storage and computation if the number of iterations required for convergence is large. One way to deal with these difficulties is to use their restarted versions instead of the full versions. The restarted versions of GMRES, GCR and IGCG are referred to as GMRES(m), GCR(m) and IGCG(m) respectively. In these forms, the algorithms are restarted after every m steps if the convergence is not achieved. The rate of convergence will inevitably become slower and some convergence properties of the full versions may not hold in the restarted cases. Besides, how to choose an appropriate restart value m remains an unanswered question.

The GMRES algorithm is theoretically equivalent to GCR and ORTHODIR, but it is less costly in terms of both storage and arithmetic[14]. Furthermore, GMRES can not break down unless it delivers the exact solution in a sense of exact arithmetic. Therefore, GMRES(m) will be evaluated in our numerical experiments.

Unlike the GMRES method, BCG, CGS and BCGSTAB only need to keep the vectors generated at the last step which means much less storage as well as less computation is required, especially for large problems.

Because of this attractive property, BCG, CGS and BCGSTAB are selected to be fully tested and are compared with GMRES(m). For detailed descriptions of the methods, we refer to References 6, 14, 17, 19.

We denote by NZ the number of nonzero entries in A. The number of multiplications in a sparse matrix-vector product will be NZ.

If the costs of small computing are neglected and the storage of A, b and x are not included, the total multiplications and the number of vectors to be stored in GMRES(m), BCG, CGS and BCGSTAB at each step are summarized in *Table 1*, in which the multiplication of GMRES(m) means average operations at each step.

Remark 1: Compared with the BCG method, CGS and BCGSTAB need small additional costs in both floating point operations and computer memory requirement, but eliminate the need of dealing with A^T, which makes codes marginally more complicated. It is also clear that BCGSTAB is slightly more costly than CGS.

Table 1 Storage and multiplications of BCG, CGS, BCGSTAB and GMRES(m)

	BCG	CGS	BCGSTAB	GMRES(m)
Vectors stored	5	7	7	(m+2)
Multiplications	$2NZ+7n$	$2NZ+9n$	$2NZ+10n$	$NZ+(m+4)n$

Table 2 The forms of $\bar{A}, \bar{x}$ and $\bar{b}$ for different preconditioning versions

Preconditioning version	$\bar{A}$	$\bar{b}$	$\bar{x}$
Left	$(LU)^{-1}A$	$(LU)^{-1}b$	x
Right	$A(LU)^{-1}$	b	$(LU)x$
Left-right	$L^{-1}AU^{-1}$	$L^{-1}b$	Ux

Remark 2: In general, GMRES(m) may be more expensive in both computational time and storage requirements, but it needs only one matrix-vector multiplication at each step. Hence, average multiplication of GMRES(m) at each step may be less than the others in the case where the matrix-vector product is costly.

PRECONDITIONING

A suitable preconditioner is crucial to obtaining a rapid convergence of iterative methods and choosing good preconditioners for general matrices is an important research issue. An efficient preconditioner should be a close approximation of matrix A, easily computed and inverted (explicitly or implicitly) as well as reasonably sparse. Moreover, all relevant computation processes should be concurrent if the iterative methods need to be highly parallelized or vectorized. A commonly used preconditioning technique that possesses these properties (except concurrency) is the incomplete Cholesky decomposition (ILU) with no fill-in scheme, called ILU(0)[12].

Although Axelsson *et al.*[4] showed that block incomplete matrix factorization methods can work exeptionally well on convection-diffusion problems, the incomplete factorization is based on line block in a polygonal region and is not applicable for general subdomains with an unstructured mesh. Hence we select ILU(0) (the diagonal terms of L are set to be unity) as the preconditioner in our following numerical experiments.

Since no fill-in is allowed for in ILU(0) factorization, the solution of the equations such as $LUx=b$ involves the same number of floating point operations as the sparse matrix-vector product Ax.

In applying ILU preconditioning to nonsymmetric systems of equations, there are three slightly different versions to choose, which are referred as left, right and left-right preconditioning respectively. Assuming the equivalent form of the original system (1) after preconditioning is as follows:

$$\bar{A}\bar{x}=\bar{b} \tag{3}$$

then $\bar{A}$, $\bar{x}$ and $\bar{b}$ will show different forms with the above different preconditioning versions, as listed in *Table 2*.

Different forms of $\bar{A}$ have different condition numbers and different distributions of eigenpairs, and consequently have different effects on the convergence of iterative methods. No theoretical results, however, seem available to compare the efficiencies of these three preconditioning versions, and often contradicting opinions are presented in References 10, 18. Hence all three preconditioning patterns will be examined in our test problems. We note that ILU may break down in practical problems as A can not be guaranteed to be a M-matrix[12].

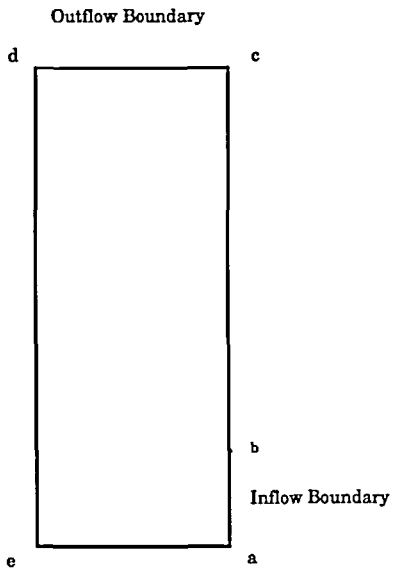

Figure 1 The flow region of the test problem

Table 3 The details of finite element models

Mesh	Nodes	Elements	Orders	Nonzero entries
Low resolution	10201	10000	10195	90545
High resolution	40401	40000	40390	361485

NUMERICAL EXPERIMENTS

In this section, we undertake numerical experiments to compare the performance of GMRES(m) with other conjugate gradient-like methods, BCG, CGS and BCGSTAB. The tests are performed on a HP-9000 series 730 workstation, using double precision. All Fortran codes are compiled and linked optimally to level 2.

The test problem selected is a 2-dimensional 10m × 50m rectangular region as shown in *Figure 1*, with a 2.5m inflow boundary Γ_i at the right bottom corner (line *a-b*), and an outflow boundary Γ_o at the top (line *c-d*). Two different finite element mesh resolutions are used to discretize this region; one low resolution with 100 × 100 regular mesh and the other a high resolution with 200 × 200 regular mesh. More detailed descriptions are listed in *Table 3*, including the number of nodes, elements and order of equations, as well as the nonzero entries in A.

The velocity field $V(x)$ is firstly determined by the following model[13]:

$$\nabla \cdot V = 0 \tag{4}$$

$$V = -G\nabla p(x) \tag{5}$$

where $p(x)$ is the pressure in the region and G is defined by:

$$G = \alpha |\nabla p|^{\beta}$$

in which α and β are constants, and $\beta = -0.1$ is used in our computations.

Table 4 The costs of iterative methods with left preconditioning for low resolution

	Iterations			CPU time		
nf	CGS	BCGSTAB	GMRES	CGS	BCGSTAB	GMRES
0.0	144	143	276@	80.34	81.30	475.7
.184e-1	69	71	105	38.60	40.44	178.7
.245	36	42	56	20.38	24.23	57.89
.735	35	38	46	19.79	21.81	41.28
.992	35	36	44	19.79	20.85	38.28
1.82	35	39	48	19.79	22.44	44.35
3.58	57	60	72	32.19	34.65	90.34
6.73	447	650	128	251.6	371.4	259.7

Note @: computed by restarted version m=144; ×: not converged

Substituting (5) into (4) results in:

$$\nabla\cdot(G\nabla p)=0$$

which is a quasi-linear elliptic partial differential equation and can be solved numerically to obtain nodal pressure values by the finite element method, combined with the following boundary conditions:

$$p|_{\Gamma_i}=p_o, \qquad p|_{\Gamma_o}=0$$

The velocity fields in every element may then be easily computed based on the element shape functions and the relationship (5). The same meshes are used in both velocity field and convection-diffusion computation.

In all cases we set $c_x=c_y=c$. With the Petrov-Galerkin procedure[21], the element stiffness of the convection-diffusion problem is represented in the following form:

$$K_e=\int_\Omega W^T v_i\frac{\partial N}{\partial x_i}d\Omega+c\int_\Omega\frac{\partial W^T}{\partial x_i}\frac{\partial N}{\partial x_i}d\Omega+b.c.\ term$$

where W is the weighting function, the form of which could be found in Reference 21. Therefore, the global stiffness matrix A will be nonsymmetric unless $V=0$. If we choose different values of c, A will have different degrees of nonsymmetry, which can be measured quantitatively by the nonsymmetric factor *nf* defined previously. In addition, diagonal dominance in our cases is violated slightly.

All three preconditioning versions, left, right and left-right are examined in our tests. Here we directly solve equations $\bar{A}\bar{x}=\bar{b}$ rather than $Ax=b$. The terminating criterion is selected by requiring the relative residual norm to be less than $\varepsilon=10^{-10}$, i.e.:

$$\frac{\|\bar{A}\bar{x}-\bar{b}\|}{\|\bar{r}_o\|}<\varepsilon$$

The total number of iterations and the solution CPU times (in seconds) of CGS, BCGSTAB and GMRES with three preconditioning versions for low and high resolutions are listed in *Table 4–6* and *Table 7–9* respectively. We find that BCG is much inferior to the other methods, and thus the corresponding results are not included in the tables. For the low resolution mesh, the full version of GMRES is used except for the case that *nf*=0.0 where GMRES(144) is applied. The restarted version (m=50) is used for high resolution mesh.

The computations show that when convection term becomes significant, the incomplete Cholesky factorization may break down or *LU* may become nearly singular, whch causes the iterative procedure barely operable in the former case or the accuracy of the final solutions to

Table 5 The costs of iterative methods with right preconditioning for low resolution

	Iterations			CPU time		
nf	CGS	BCGSTAB	GMRES	CGS	BCGSTAB	GMRES
0.0	147	136	276	81.23	78.50	475.7
.184e-1	71	77	103	39.72	43.91	172.6
.245	39	38	55	22.02	21.81	56.08
.735	29	39	45	16.47	22.44	39.77
.992	26	35	44	14.82	20.20	38.28
1.82	46*	42	50	25.88	24.23	46.57
3.58	67	73	77	37.41	41.58	101.5
6.73	776	317	144	480.4	180.2	323.1

Note: the accuracy of final solution is deteriorated (*) or severely deteriorated (**)

Table 6 The costs of iterative methods with left-right preconditioning for low resolution

	Iterations			CPU time		
nf	CGS	BCGSTAB	GMRES	CGS	BCGSTAB	GMRES
0.0	145	143	276	80.62	81.00	475.7
.184e-1	70	71	104	39.08	40.44	175.7
.245	39	40	56	22.02	22.96	57.89
.735	30	40	45	17.00	22.96	39.77
.992	29	40	44	16.47	22.96	38.28
1.82	41*	42	50	23.06	24.23	46.57
3.58	61	74	76	34.24	42.08	99.23
6.73	654	337	145	366.7	195.7	325.0

Table 7 The costs of iterative methods with left preconditioning for high resolution

	Iterations			CPU time		
nf	CGS	BCGSTAB	GMRES	CSG	BCGSTAB	GMRES
0.00	293	273		509.4	481.5	
.094	107	100	596	186.7	177.1	1823.2
.187	69	84	449	120.7	140.8	1378.5
.649	66	48	84	115.5	103.4	232.8
.936	84**	65	82	147.7	117.2	226.8
1.87	129	60*	93	224.4	106.6	271.6
3.70	128**	241**	231**	294.5	426.2	683.4

Table 8 The costs of iterative methods with right preconditioning for high resolution

	Iterations			CPU time		
nf	CGS	BCGSTAB	GMRES	CGS	BCGSTAB	GMRES
0.0	293	263		509.4	467.6	
.094	88	90	478	154.0	159.6	1436.5
.187	70	75	298	122.8	133.4	912.5
.749	49*	59	78	86.24	104.2	212.4
.936	49**	61	85	86.24	109.0	239.2
1.87	101*	84	122	176.5	148.2	347.0
3.70	210**	172**	341**	367.2	304.7	1125.9

Table 9 The costs of iterative methods with left-right preconditioning for high resolution

	Iterations			CPU time		
nf	CGS	BCGSTAB	GMRES	CGS	BCGSTAB	GMRES
0.0	297	269		517.5	476.1	
.094	86	104	606	150.2	184.1	849.6
.187	75*	93	512	131.0	165.0	1553.
.749	48**	65	91	84.33	115.4	262.6
.936	50**	69	86	89.23	122.7	241.2
1.87	97*	84	110	169.2	148.9	320.5
3.70	210**	×	394**	367.2		1200.

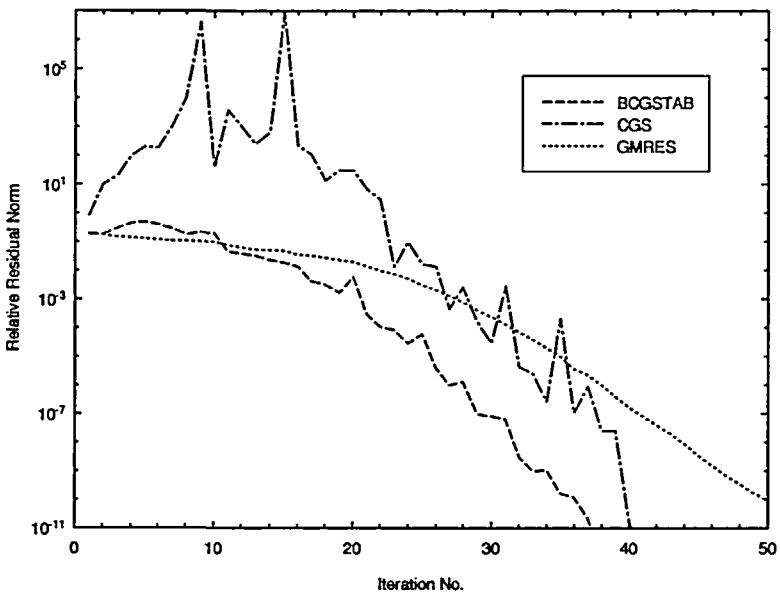

Figure 2 Convergence histories of GMRES, CGS and BCGSTAB (low resolution)

be very low in the latter. Therefore, for convection dominated problems, more reliable preconditioning techniques may be needed.

From the results we can not directly compare the performances of the different precondition patterns since the terminating criterion is not the same in these cases. However, no consistent patterns emerges when comparing the various methods for different *nf* values. We observe, however, that with right and left-right preconditioning the CGS method sometimes exhibits high oscillation in the convergence histories, as shown in *Figures 2* and *3* (both with right preconditioning), whereas the situation appears to be much better with left preconditioning.

We also notice that CGS is slightly more efficient than BCGSTAB in almost all cases, but it sometimes suffers from numerical instability as mentioned above, which may also severely deteriorate the accuracy of the final solutions. For example, in the case shown in *Figure 3*, the final solution has only an accuracy of approximately 10^{-3}, although the terminating criterion has been satisfied. However, BCGSTAB shows a better numerical stability in all cases. On the other hand, the convergence rate of GMRES exhibits no oscillatory behaviour but is slower

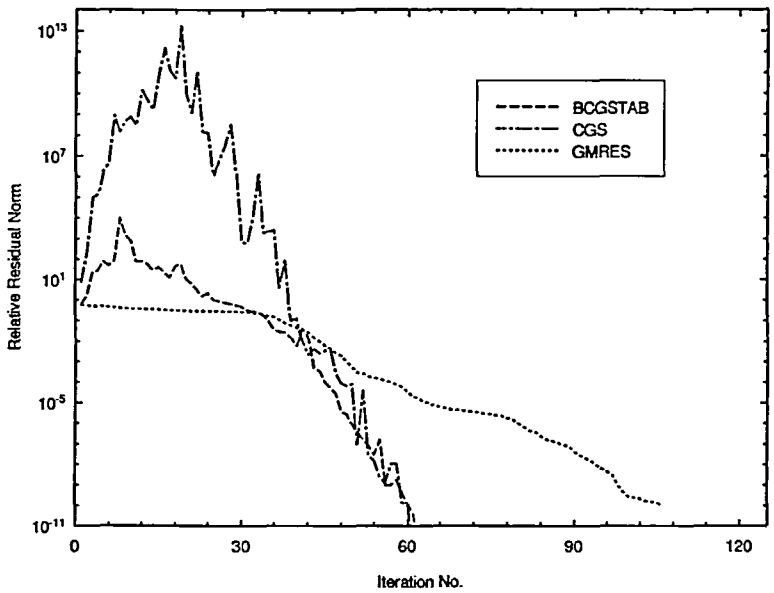

Figure 3 Convergence histories of GMRES, CGS and BCGSTAB (high resolution)

compared with BCG and BCGSTAB. If nf increases to a large value, the convergence rates of CGS and BCGSTAB begin to decrease, while GMRES still performs reasonably well.

It is interesting to note that when $nf \approx 0.1-2.0$ the costs of the iterative methods are only approximately one third of the symmetric case ($nf=0$). This observation is consistent with the work of Axelsson[4] where the number of iterations for convection-diffusion problems is sometimes significantly smaller than that for the corresponding (nearly) symmetric problem.

For pure diffusion problems, A becomes SPD in which case CG needs 178 iterations to converge for the low resolution case, about 40 iterations more than CGS and BCGSTAB. But the CPU time is nearly half, since CGS and BCGSTAB need much more work at each step. As for the high resolution case, CG is much inferior to CGS and BCGSTAB in terms of iterations and CPU times.

FINAL REMARKS

The numerical results shown in the previous section indicate that BCGSTAB appears to be an efficient and stable iterative method for the solution of convection-diffusion problems for the time being. Although CGS appears to be slightly more efficient than BCGSTAB, it sometimes suffers from severe numerical instability which might deteriorate the computational results. On the other hand, GMRES shows a higher suitability but the overall convergence rate is lower.

It should be noted that the test problem in our work is 2-D and GMRES may be less expensive for 3-D problems. Therefore, further 3-D test problems are needed to confirm the conclusions presented here.

It is also important to observe that present results are related to the global (ILU) type preconditioning, while various other preconditioning techniques are currently being explored. A promising prospect, for instance, is offered within a multigrid preconditioning technique. In addition, a combination of a direct and iterative solver within a domain decomposition method, where the local problem is solved by a direct method and the interface problem is solved by an iterative solution is expected to significantly improve the overall conditioning.

ACKNOWLEDGEMENT

The work of the first author is funded in part by the *FOK YING TUNG* Education Foundation, and this support is gratefully acknowledged.

REFERENCES

1 Axelsson, O. Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations, *Lin. Alg. Appl.*, **29**, 1–16 (1980)
2 Axelsson, O. A generalized conjugate gradient, least square method, *Numer. Math.*, **51**, 209–227 (1987)
3 Axelsson, O., Brinkeuper, S. and Il'in, V. P. On some versions of incomplete block-matrix factorization, *Lin. Alg. Appl.*, **58**, 3–15 (1984)
4 Axelsson, O., Eijkhout, V., Polman, B. and Vassilevski, P. Incomplete block-matrix factorization iterative methods for convection-diffusion problems, *BIT*, **29**, 867–889 (189)
5 Eisenstat, S., Elman, H. and Schultz, M. Variational iterative methods for non-symmetric systems of linear equations, *SIAM J. Numer. Anal.*, **20**, 345–357 (1983)
6 Fletcher, R., Conjugate gradient methods for indefinite systems, *Lecture Notes in Mathematics*, 73–89, Springer, New York (1976)
7 George, A. and Lin, J. W. H. *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ (1981)
8 Habashi, W. G., Nguyen, V. N. and Bhat, M. V. Efficient direct solvers for larger-scale computational fluid dynamic problems, *Comp. Meth. Appl. Mech. Engng.*, **89**, 253–265 (1991)
9 Jea, K. and Young, D. On the simplification of generalized conjugated-gradient methods for nonsymmetrizable linear systems, *Lin. Alg. Appl.*, **52, 53**, 399–417 (1983)
10 Kadioglu, M. and Mudrick, S. On the implementation of the GMRES(m) method to elliptic equations in meterology, *J. Comp. Phys.*, **102**, 348–359 (1992)
11 Kane, J. H. and Prasad, D. K. K. G. Iterative solution techniques in boundary element analysis, *Int. J. Num. Meth. Engng.*, **31**, 1511–1536 (1991)
12 Meijerink, J. and van der Vorst,H. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, *Math. Comp.*, **31**, 148–162 (1977)
13 Miketinac, M. J. and Sokhansanj, S. Velocity-pressure distribution in grain bins – Brooker's model, *Int. J. Num. Meth. Engng.*, **21**, 1067–1075 (1985)
14 Saad, Y. and Schultz, M. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comp.*, **7**, 856–869 (1986)
15 Shakib, F., Hughes, T. J. R. and Johan, Z. A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis, *Comp. Meth. Appl. Mech. Engng.*, **75**, 415–456 (1989)
16 Sonneveld, P. CGS: a fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Stat. Comp.*, **10**, 36–52 (1989)
17 Sonneveld, P., Wesseling, P. and de Zeeuw, P. Multigrid and conjugate gradient methods as convergence acceleration techniques, D. Paddon and H. Holstein, eds, *Multigrid Methods for Integral and Differential Equations*, 117–167, Clarendon Press, Oxford (1985)
18 Tan, L. and Bathe, K. Studies of finite element procedures – the conjugate gradient and GMRES methods in ADINA and ADINA-F, *Comp. Struct.*, **40**, 441–449 (1991)
19 van der Vorst, H. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear equations, *SIAM J. Sci. Stat. Comp.*, **13**, 631–644 (1992)
20 Young, D. and Jea, K. Generalized conjutate-gradient acceleration of nonsymmetrizable iterative methods, *Lin. Alg. Appl.*, **34**, 159–194 (1980)
21 Zienkiewicz, O. C. and Taylor, R. L. *The Finite Element Method: Solid and Fluid Mechanics, Dynamic and Non-linearity (Fourth Edition)*, **2**, McGRAW-Hall Book Company, London (1991)