JOURNAL OF COMPUTATIONAL PHYSICS **102**, 348–359 (1992)

On the Implementation of the GMRES(m) Method to Elliptic Equations in Meteorology[1]

MIKDAT KADIOĞLU AND STEPHEN MUDRICK

Department of Atmospheric Science, University of Missouri, Columbia, Missouri 65211

Received July 5, 1990; revised June 17, 1991

In this paper we consider the relatively new preconditioned generalized minimal residual method, restarted every m iterations (GMRES(m)), for the solution of three-dimensional elliptic equations. Large, sparse, non-symmetric matrices are involved. The particular equation of interest is the quasi-geostrophic "omega" equation, often used in meteorology to compute vertical motion. The GMRES(m) method is tested with different preconditioners for the solution of two- and three-dimensional elliptic equations. The method requires no relaxation parameters and has no restrictions on the size of the 3D grid. GMRES can be used for more types of matrices than other methods such as SOR. Numerical results show that Jacobi preconditioned GMRES(m) performs best for 3D and high resolution problems among five different preconditioners tested, while ILU factorization of the partial or whole matrix A, as a preconditioner, is good for 2D and low resolution problems. The SOR preconditioners for the GMRES(m) method, with optimal relaxation parameters, are not as efficient, and the best choice for the relaxation parameter in SOR preconditioning is not the same as the best choice for the simple SOR method. An algorithm for using the preconditioned GMRES(m) method is presented. © 1992 Academic Press, Inc.

1. INTRODUCTION

The rapid advances in all phases of dynamical–numerical weather forecasting as well as in other areas of computational fluid dynamics have created more complex and larger systems of equations. Therefore, more efficient techniques for solving them are necessary.

In this paper, we consider a relatively new, efficient, and robust general iterative procedure for solving large, sparse systems of linear algebraic equations arising from an old meteorological problem. Our purpose is to demonstrate that the preconditioned generalized minimal residual method (GMRES) can be useful as a general-purpose solver for linear (elliptic) partial differential equations (PDEs).

Consider the system of equations

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

where A is a real, nonsingular $n \times n$ matrix, b is a given vector of order n, and the value of the vector x of order n is sought. Apart from knowing the general method used to simulate these problems, we will assume here that almost nothing is known a priori about the mathematical properties of system (1). We do know, however, that our system is large, sparse, and, in general, nonsymmetric.

Many scientific and engineering problems involve the solution of such a set of linear equations at some stage of analysis. The large, sparse system of linear equations, in our case, arises after a second-order finite-difference discretization of the quasi-geostrophic omega equation, which is used for determining the three-dimensional (3D) distribution of atmospheric vertical motion.

In general, there are two categories of linear equations solvers: direct and iterative methods. For problems arising from PDEs in 3D domains, direct methods can be too costly. Advantages of iterative methods for solving large, sparse systems lie in the fact that the matrix factorization of direct methods is avoided, with consequent substantial saving in storage, and that an approximate solution of (1) can be extracted in less than n (order of matrix) iterations, thus saving computing time (e.g., [29]).

For systems such as (1), however, very efficient direct methods also have been developed, which are called fast or fast direct methods. They are usually based on the fast Fourier transformations or the method of cyclic reduction (e.g., [5, p. 225]). Some of these algorithms are utilized in the software package FISHPAK [5, 31]. However, these techniques are limited primarily to systems which arise from separable self-adjoint boundary value problems, and the grid intervals must be determined according to the products of a small number of prime numbers [5].

Most iterative methods for solving (1) are developments of one of two basic ideas, namely Jacobi's method, and the conjugate gradient (CG) algorithm. For recent reviews, see [9, 17, 35]. According to Young [35], Wachspress [37], Roache [27], and many others, elliptic PDEs have been solved numerically by the very popular *successive overrelaxation* method (SOR), based on Jacobi's method. According to Haltiner and Williams [13, p. 157] and Boisvert and Sweet [5, p. 225], SOR is also the most

[1] Contribution from the Missouri Agricultural Experiment Station. Journal Series Number 11221.

common method of solution used by meteorologists or fluid dynamicists, because of its easy implementation.

The traditional SOR, however, is not an effective general solution technique for a linear equations system [35, p. 36]. First, it requires the choice of the best over-relaxation coefficient γ_{opt} to optimize the rate of convergence of the iteration. In general, this is a very difficult problem. Theory for finding γ_{opt} only exists for some special classes of coefficient matrices [3, 17, 26, 36]. Even for special matrices, computation of the extreme eigenvalues of the matrix for optimal convergence is usually as expensive as solving the linear equation system. For the relatively well-defined problem of the discretized elliptic equations, convergence can be affected greatly by an obscure "optimum" parameter value (e.g., [5, 13, 27, 33]). Second, it needs a good initial approximation to the solution of (1), which is not always readily available. In addition to these, SOR is not guaranteed to converge when A is a nonsymmetric matrix (e.g., [12, p. 332; 26]). Also, if diagonal elements of the matrix are sufficiently negative, SOR will not converge [13, p. 166]. Thus, it is apparent that SOR may not be the most appropriate technique to use in the solution of nonsymmetric, large, sparse linear systems. Moreover, according to Navarra [22], standard techniques similar to SOR that have been developed and used in meteorology cannot be applied to large linear problems [18]. Some of the other classical techniques used for elliptic boundary value problems, such as Guassian elimination, point iterative methods, are reviewed by Fulton *et al.* [7].

An extremely fast solution for (1), however, can be extracted by multigrid methods, which combine some of the classical iterative techniques with a subgrid refinement procedure. Multigrid and related methods form a new class of techniques which give the fastest known solution for a significant category of matrix problems. For a recent review on the multigrid methods see [7]. Fortran subroutines that solve linear elliptic PDEs using multigrid iteration techniques are utilized in a multigrid package MUDPACK by Adams [1]. Both FISHPAK and MUDPACK are readily available from the National Center for Atmospheric Research. However, not only do the properties of the physical problem, the discretization, and the choice of the appropriate relaxation in some cases lead to a degradation of the performance of the multigrid methods, but also the usage of MUDPACK is possible if and only if the data satisfy the discrete compatibility condition. In other words, according to MUDPACK Version 1.2, February 1989 documentation, the 2D or 3D domains data must be provided on uniform grids that have the form $nx = p \cdot 2^k + 1$, $ny = q \cdot 2^k + 1$, and $nz = r \cdot 2^k + 1$, where p, q, r, and k are positive integers (p, q, and r preferably smaller than 5). For the problem we are considering here, our data are not able to satisfy these conditions. We, therefore, still need an algorithm which solves our problems to a desired accuracy by a "reasonable" amount of work and which has few restrictions.

Although the CG methods do not require either estimates for the eigenvalues or an optimization parameter, the methods have not been used extensively on systems arising from the discretization of PDEs until recently. This is presumably because they required more storage and they were not significantly faster than SOR. Rapid improvements in CG algorithms and sparse matrix techniques, however, have made them more practical for use. (For more discussion see, e.g., [6, 14].)

Although powerful CG algorithms exist for solving symmetric systems of linear equations, efficient methods for solving large nonsymmetric systems are still rare [6, 29]. There are three main approaches for the generalization of CG for nonsymmetric problems. One approach is based on transforming the nonsymmetric problem into a symmetric positive definite one, then using the CG method to solve the resulting system, called the normal equations (the CG iteration applied to the normal equations (CGN)). The CGN method is not always applicable in practice because it may require a large number of iterations to converge [15, 31]. The biconjugate gradient method (BCG) is another one which re-interprets the CG algorithm, using a recurrence formula so that full orthogonalization is not necessary. The BCG method also may fail to converge or break down in many cases of practical interest [15, 31, 35]. A recent extension of BCG, the conjugate gradient squared method (CGS), however, is found to be more efficient than BCG [31]. The other generalization is GMRES. It is a residual minimization method which basically results by the introduction of an optimality property into the Arnoldi algorithm. Like BCG it uses a Krylov subspace generated by A, but with full orthogonalization [28–30].

According to Nachtigal *et al.* [21], among the many parameter-free matrix iteration techniques proposed for the solution of nonsymmetric systems of linear equations, the GMRES method is the most robust one. They test what they call the three leading methods, CGN, CGS, and GMRES, on eight different classes of matrices, and they found that GMRES always converged with a reasonable number of iterations. An empirical comparison with realistic computations is also given by Radicati *et al.* [25]. Their results also support those of Nachtigal *et al.*; GMRES is very robust and more efficient than the other methods for their physical problem. Gómes and Morales [10] have shown that GMRES is a robust and successful technique for the solution of problems arising from oil-reservoir simulations. In addition, Navarra [22, 23] also pointed out that Krylov subspace techniques, such as Arnoldi and GMRES can be useful for solving large linear systems resulting from the discretization of geophysical fluid dynamic problems. For more comparative performances of GMRES with other CG-like methods, see [28, 29].

Therefore, we have chosen the recently introduced iterative preconditioned GMRES technique in order to evaluate its utility in a meteorological problem. Moreover, since the theory and practice of GMRES and preconditioned GMRES are not well understood, because only limited numerical experiments have been done using them, particularly for real problems modelled by PDEs [6, 10], we will present and clarify some of the computational aspects of this method. Also, a comparison between some of the common preconditioners will be presented using GMRES, in order to shed some light on choosing a "good" preconditioner in real world applications.

In Section 2, the basis for our system of linear equations is described. Some computational aspects of GMRES are given in Section 3. In the following sections, tests for preconditioned GMRES will be given to detect weaknesses, display strengths, and explore robustness. Conclusions are drawn for the choice between solvers in 2D and 3D in Section 5.

2. MODEL PROBLEM AND METEOROLOGICAL CASES

For weather forecasting, it is vital to be able to make deductions about the vertical motion, which typically is of the order of a few centimeters per second and which cannot be measured directly. Upward motion produces precipitation and plays an important role in the development of fronts and storms, etc. One method of getting the vertical motion involves inverting PDE using the output from a 3D forecast model.

Our 3D, non-linear model employs 100 km (200 km) and 0.5 km (1.5 km) horizontal and vertical grid lengths, respectively; an integration uses $54 \times 62 \times 30$ ($26 \times 30 \times 10$) grid points in the east–west, north–south, and vertical directions, respectively, in high resolution runs (and as numbers in parenthesis indicate, in low resolution runs). The domain of integration is a west–east re-entrant channel with rigid horizontal and vertical boundaries:

$$0 \leqslant x \leqslant l, 0 \leqslant y \leqslant 1, 0 \leqslant z \leqslant 1,$$

where l is the nondimensional channel length.

We are required to solve elliptic, 3D partial differential equations of the form

$$\left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{1}{N(z)^2}\frac{\partial^2}{\partial z^2}\right] w(x, y, z) = F(x, y, z), \qquad (2)$$

where F is a known forcing function, N^2 is a known stability parameter, and the appropriate boundary conditions (BCs) are

$$w(x, y, 0) = w(x, y, 1) = 0$$

$$\alpha(x \pm l, y, z) = \alpha(x, y, z),$$

where α is any variable.

We solve for w, the vertical motion. The vertical direction is z, the height. An example of such an equation is the so-called "quasi-geostrophic (QG) omega equation," used to obtain the vertical movement of air [13]. Knowing the 3D distribution of pressure at a given time, one can evaluate F. The distribution of temperature with height, averaged horizontally, is related to the stability of the atmosphere with respect to vertical overturning, and is the basis for N^2. The greater the decrease in temperature per given height interval, the more prone is that atmospheric region to "instability" and the smaller is N^2. Knowing F and N^2, one solves the equation for the three-dimensional distribution of w. Regions of significant upward motion ($w > 0$) can lead to cloud formation and precipitation. Equation (2) is approximated by a system of linear equations evaluated on a three-dimensional grid. For an internal grid point ijk, away from any boundaries, our finite-difference approximation for (2), to second-order accuracy, is given by

$$\begin{aligned} w_{i+1jk} + w_{i-1jk} &+ \frac{\Delta x^2}{\Delta y^2}(w_{ij+1k} + w_{ij-1k}) \\ &+ \frac{\Delta x^2}{N_k^2 \Delta z^2}(w_{ijk+1} + w_{ijk-1}) \\ &- 2\left(1 + \frac{\Delta x^2}{\Delta y^2} + \frac{\Delta x^2}{N_k^2 \Delta z^2}\right) w_{ijk} = \Delta x^2 F_{ijk}. \qquad (3) \end{aligned}$$

Equation (3), applied at each grid point, plus modified equations incorporating the boundary conditions, are to be inverted to obtain all w_{ijk}'s.

The forcing function F_{ijk} can be written as the sum of two terms, a "vorticity advection" and a "thickness advection" term. We solve Eq. (3) twice, once with each term of the forcing function; the two vertical motion solutions w are added to get the total w. Each component of w provides different information concerning the dynamics of the weather system. Here we discuss only results from the "vorticity advection" forcing term; the other forcing term gives similar results.

Equation (3) requires BCs for w at all boundaries; we have not discussed the northern and southern channel walls. If we write $w(x, y, z) = \bar{w}(y, z) + w'(x, y, z)$, where $\overline{(\ )}$ is the east–west average and $(\ )'$ is the deviation, we can split Eq. (3) into two elliptic equations, a 3D equation for $w'(x, y, z)$, which looks like (3), and a 2D equation for $\bar{w}$, in y and z. It turns out, from the BCs associated with the complete set of atmosphere simulation equations used to derive (3), the so called QG equations, that, at the northern and southern walls, $w' = 0$. Thus the 3D w' equation has complete BCs. For the 2D $\bar{w}$ equation we rederive an equation for the east–west mean streamfunction, for which the BCs are zero on the y and z boundaries. Solving this 2D elliptic streamfunction equation allows us to obtain $\bar{w}$; we then

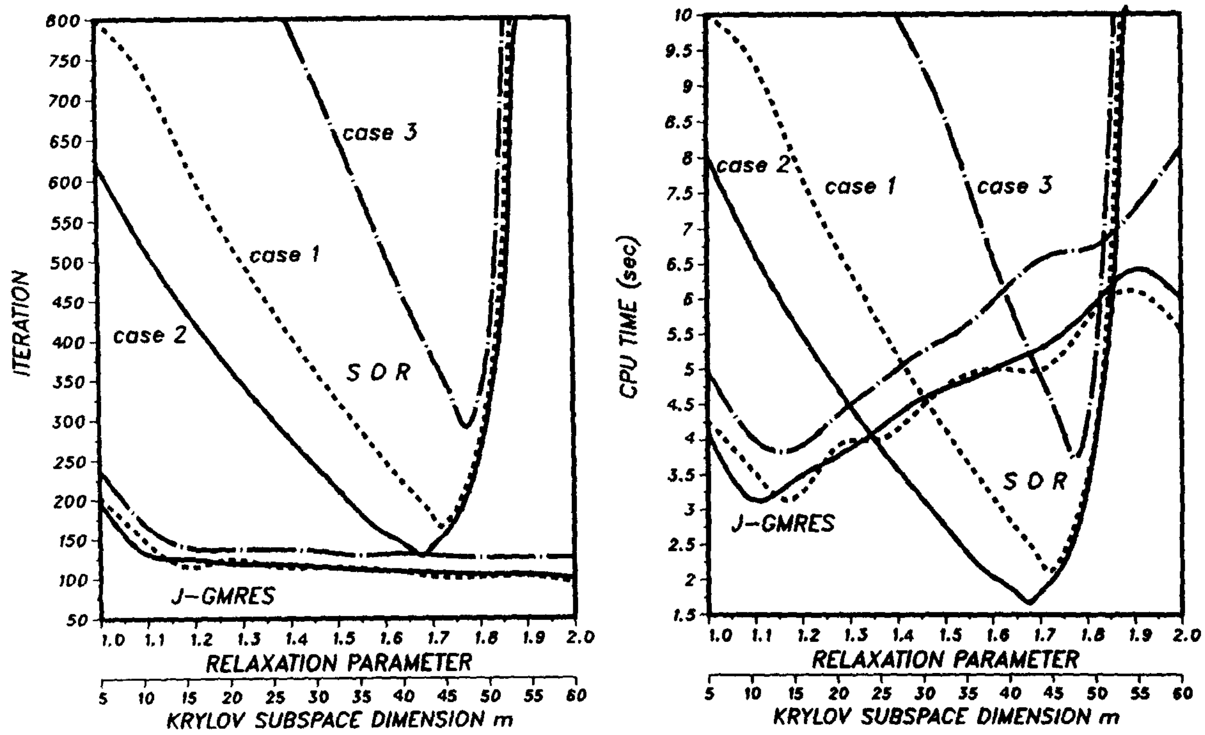


FIG. 1. Performance of the J-GMRES and the SOR methods as a function of the relaxation parameter γ and Krylov subspace dimension m.

reconstruct w. In what follows, the "3D" and "2D" refer to these two elliptic equations, rather than to Eq. (3).

The forcing functions were obtained from three different cases, representing three different types of atmospheric situations. Cases 1 and 2 used typical mid-latitude values of tropospheric and lower stratospheric stability for N^2. (The ranges of nondimensional values were 0.166 to 0.587 and 0.056 to 0.331, respectively.) Case 1 was very simple, with one high pressure and one low pressure region superimposed on an idealized west to east mid-latitude flow. (Details of the "QG" model used to generate the pressure data which was used as input for the forcing functions can be found in Mudrick [20]; case 1 is labelled 2QG-INT in that paper.) Case 2 used a similar west to east "basic state" flow, but two "highs" and two "lows" were superimposed. Their interaction produced a more realistic and complicated evolution and hence a more complicated pattern of w. Case 3 was similar in appearance (initially) to case 1 but a lower value of tropospheric stability was specified; i.e., N^2 in Eq. (3) was reduced. The result was that the number of iterations required for convergence of case 3 increased significantly. This was due to wider range of values of N^2, increasing the asymmetry in the coefficient matrix (the range of N^2 values were 0.053 to 2.418), compared to the less nonsymmetric cases 1 and 2 as presented in Figs. 1 and 2 for both GMRES and SOR. The increase was much more pronounced for SOR; see Fig. 1. In what follows, case 2 is emphasized; it represents more typical and realistic atmospheric conditions than the other two cases.

3. THE GMRES(m) METHOD AND ITS IMPLEMENTATION

Here we consider only the implementation of the GMRES method. Following [8, 28–30], we try to sketch the essential points of the algorithm. Theoretical details can be found in these references.

The GMRES method basically minimizes a norm of the residual at each step over a subspace. The subspace increases with the number of iterations and, therefore, the number of vectors requiring storage and operations will be increased. Thus, we use the algorithm iteratively by restarting it every m iterations. This restarted version of GMRES is denoted as GMRES(m) by Saad and Schultz [29]. For brevity, we mostly drop (m)'s; from here on GMRES represents the restarted version GMRES(m).

First we establish some notation. Matrices are denoted by capital letters; vectors and scalars are denoted by lower case letters. Column vectors of matrices, iteration steps, and dimensions of vectors are indicated by subscripts, and vector elements are shown in parentheses.

In brief, the GMRES method begins with an initial approximate solution to Eq. (1) of x_0 and initial residual $r_0 = b - Ax_0$. It then computes an approximate solution $x_j = x_0 + z_j$, at the jth iteration, where z_j belongs to a Krylov subspace $\mathcal{K}_j = \text{span}\{r_0, Ar_0, ..., A^{j-1}r_0\}$ whose residual norm $\|b - Ax_j\|$ is a minimum. Here $\|\cdot\|$ denotes the Euclidian norm (i.e., $\|x\| = (x_1^2 + x_2^2 + \cdots + x_n^2)^{1/2}$).

The method uses a modified Gram–Schmidt process (see

[8]) in the Arnoldi iteration. An orthonormal basis $V_{n\times m}=[v_1, ..., v_m]$ (i.e., $v_i^T v_j=0$ for $i\neq j$ and $v_i^T v_j=1$ for $i=j$) for the Krylov subspace is generated. The matrix A can then be transformed into an upper-Hessenberg matrix $H_{m\times m}$ (i.e., $h_{ij}=0$ if $i>j+1$) through the relation

$$V^T_{n\times m}A_{n\times n}V_{n\times m}=H_{m\times m}.$$

If we let $v_1=r_0/\|r_0\|$, $\beta=\|r_0\|$, and let $\mathscr{H}$ denote the $m\times m+1$ matrix obtained by appending to $H_{m\times m}$ a row with a single nonzero entry $h_{m+1,m}$ in column m, then the Arnoldi basis matrices $V_{n\times m}$ and $\mathscr{H}_{m+1\times m}$ would satisfy

$$A(v_1, ..., v_m)=(v_1, ..., v_{m+1})\,\mathscr{H}_m$$

or

$$Av_j=\sum_{i=1}^{j+1} h_{ij}v_i \qquad \text{for} \quad 1\leqslant j\leqslant m.$$

Since v_1 is known, we can start to generate nonzero elements of $\mathscr{H}_{kj}$ and $V_{n\times k}$, where $k=j+1$. For $j=1$ to m,

$$h_{ij}=v_i^T Av_j \qquad \text{for} \quad i=1 \text{ to } j,$$

then

$$v_{j+1}=r_j/\|r_j\|,$$

where

$$r_j=Av_j-\sum_{i=1}^{j} h_{ij}v_i.$$

With these orthonormal bases and upper-Hessenberg matrices an approximate solution $x_j=x_0+z_j$ is extracted from the solution of the least squares problem

$$\min \|b-A(x_0+z_j)\|=\min \|\beta e_1-\mathscr{H}_{kj}\,y_j\| \text{ for } y_j,$$

where e_1 is the unit vector $e_1\equiv(1, 0, ..., 0)^T$, and $z_j=V_{n\times j}\,y_j$ at the jth iteration.

Hence, the GMRES iterate is given by $x_0+V_{n\times j}\,y_j$, and y_j is the solution to the upper-Hessenberg least squares problem. This problem is easily solved by factoring $\mathscr{H}_{kj}=Q_{kk}R_{kj}$, where Q_{kk} is a product of Givens rotations (see [8]) and R_{kj} is an upper-triangular matrix, whose last row is zero.

In this case, since Q_{kk} is unitary (i.e., $Q^TQ=I$), we have $\min \|g_k-R_{kj}\,y_j\|$ for y_j. This minimization is achieved by back-solving the triangular system

$$R_{kj}\,y_j=Q_{kk}\beta e_1=g_k,$$

where g is the transformed right-hand side, and here we have removed the last row $(j+1)$ of R_{kj} and the last component of g. This provides y_j, and then an approximate solution x_j for the linear equation system (1), and thus Eq. (3).

Note that Givens rotations also allow a very important feature for practical GMRES implementation; the absolute value of the last component of g, $g(k)$, is just the norm of the residual vector r_j. The residual at every iteration, therefore, can be determined without actually having to compute x_j [29].

Another important factor in the success of preconditioned GMRES is the application of a preconditioning technique. This transforms the original linear system into one which has a better eigenvalue spectrum and thus requires fewer iterations without greatly increasing the cost of each iteration.

We thus solve the preconditioned linear system

$$M^{-1}Ax=M^{-1}b, \tag{4}$$

instead of solving (1). This is discussed further in Section 4.3.

The above discussion suggests the following algorithm for general preconditioned GMRES implementations.

Algorithm. Iterative preconditioned GMRES(m).

```
1. Start:
   (i) Choose $x_0$ and a dimension $m$ of the Krylov subspace.
   (ii) Set-up a preconditioner matrix $M\approx A$ and factorize it, if necessary (see Section 4.3).
2. Arnoldi process:
   (i) Initialize; $r_0 \leftarrow M^{-1}(b-Ax_0)$, $\beta \leftarrow \|r_0\|$, and $c(1)\leftarrow\beta$.
   (ii) Gram–Schmidt orthogonalization:
        For $j=1$ to $m$
          if $\beta\neq 0$, then $v_j\leftarrow r_{j-1}/\beta$; otherwise STOP.
          $w\leftarrow M^{-1}Av_j$, and $r_j\leftarrow w$
          for $i=1$ to $j$, $h_{ij}\leftarrow v_i^T w$, and $r_j\leftarrow r_j-h_{ij}v_i$.
          $\beta\leftarrow\|r_j\|$, and $k=j+1$
          $h_{kj}\leftarrow\beta$, and $c(k)\leftarrow 0$.
3. Factor the upper-Hessenberg matrix $\mathscr{H}_{kj}$; $Q_{kk}\mathscr{H}_{kj}=R_{kj}$.
4. Obtain residual norm of the approximate solution $x_j$; $g(k)=Q_{kk}c_k$.
5. Make decision to form the approximate solution and restart the algorithm:
   if $|g(k)|>\varepsilon$ and $j<m$, then (go to 2(ii)) next $j$.
   if $|g(k)|\leqslant\varepsilon$ or $j=m$, then first solve the least
      squares problem $\min\|g_k-R_{kj}\,y_j\|$ for $y_j$,
      and then $x_j\leftarrow x_0+V_{kj}\,y_j$.
   if $|g(k)|\leqslant\varepsilon$, then STOP; otherwise $x_0\leftarrow x_j$
      and (go to step (2)) RESTART.
```

In summary, the above iterative GMRES algorithm computes a new direction vector v_j in the Krylov subspace spanned by $v_1, ..., A^{j-1}v_1$ and orthogonalizes it against all the previous ones which have to be stored. After several search vectors are computed by GMRES, a global minimization problem is solved.

4. COMPUTATIONAL PROCEDURE AND NUMERICAL RESULTS

In this section, we describe results that illustrate the behavior and effectiveness of the iterative methods. The numerical experiments described in this section were coded in FORTRAN and were carried out on an IBM 3090-170J scientific vector computer, using double precision. No out-of-core memory was used. We optimized every loop that could be vectorized by IBM 3090 vector facility (VF), a hardware feature that provides significantly faster run time for eligible code. No efforts, however, were made to start iterations with a good initial guess vector; we used the initial guess vector $x_0 = 0$.

4.1. *Storage*

In the above iterative GMRES algorithm, three computational kernels may be easily identified: dot products and vector updates, sparse matrix-vector products, and the application of M^{-1}. Potentially time consuming operations, such as the sparse matrix-vector product and the implementation of the preconditioner matrix M, deserve particular attention.

We first observe that these operations can be performed by diagonals, since our matrices are regularly structured. In general, matrices with regular sparseness patterns are stored by their diagonals so that the matrix–vector product involves contiguous memory locations and no indirect addressing is necessary. Thus, instead of regenerating nonzero elements of matrix A, only the nonzero diagonals of matrix A are stored to allow efficient vectorization of computational kernels [24]. The matrices generated by the 2D and 3D discretizations of a PDE on a regular grid have sparseness patterns of this sort.

We therefore applied ITPACK/ELLPACK's general storage approach to store the entries of our matrices in the *coef-jcoef* sparse nonsymmetric diagonal format. This allows us to exploit the sparsity in the computations and it allows some of the computational kernels to be efficiently vectorized [24, 25]. Here *coef* is a real array of size n-by-*maxj*, which contains the nonzero diagonals of A in its columns. The *maxj* is the maximum number of nonzeros per row of A. Upper diagonals are top-justified and lower diagonals are bottom-justified so that all rows have the same length. The *jcoef* is an integer, $n \times maxj$ array containing integers giving the distance (positive for upper diagonals, negative for lower diagonals) of each diagonal from the main diagonal.

We used five-point (for 2D) and seven-point (for 3D) finite difference schemes to discretize our elliptic problems, and the resulting matrices were five-banded and mainly nine-banded. (In addition to the seven usual diagonals in the 3D matrix, we have two extra bands due to the cyclic boundary conditions; these bands, however, consist of a few ones.) Doing this, the storage requirement for the coefficient matrices turns out to be $10 \times n$ instead of $n \times n$, where for our 3D problems, n is 5832 for 7800 unknowns in low resolution, and n is 88,972 for 100,440 unknowns in high resolution. (For more detail on storage and cost of computations in GMRES see [29].) Thus, storing those few nonzero entries of the matrix saves us from a large storage requirement and I/O costs. The storage requirements for GMRES methods, however, are substantially larger than those of the SOR method, and it increases as m increases; e.g., [10, 29]. But, as will be shown below, a reasonably small value of $m = 15$ turns out to be adequate for our purposes; this reduces the storage requirements, even for our high resolution problem, to reasonable values.

The code for sparse matrix-vector multiplication with the above storage scheme is given by Oppe *et al.* [24, p. 290].

4.2. *Efficiency and Robustness*

GMRES is effective for solving nonsymmetric linear systems arising from the discretization of elliptic problems, but little of convergence theory carries over to the nonsymmetric case (see [illegible], and Saad's papers for more discussion).

From widely varying and somewhat arbitrary test procedures for stopping criteria, we chose to monitor the Euclidian norm of the true residual vectors, $\|r_j\| = \|M^{-1}(b - Ax_j)\|$. Each step of an iterative method is subject to rounding error as well as the initial values of the elements of A and b being subject to several other types of errors, so it is necessary to check the accuracy of the final solution by insertion into the original equation, e.g., [12]. Moreover, since the norm of the residual vector gives the same weight to large and small error components over the grid points, e.g., [22], the small norms of residual vectors are not always the best indicators of small errors. We therefore conducted a few tests with different stopping criteria ε in order to extract a satisfactory approximate solution without being affected by the limits of the machine precision. A stopping criterion of $\varepsilon = 10^{-14}$ was thus chosen and used for computations. The typical magnitude of the vertical motion was 10^{-2} or 10^{-3}. In all cases, the iteration process was terminated either when the norm of the residual vector was equal to or less than ε, or after n steps for GMRES, and 1500 steps for SOR. Like the GMRES cases where we monitored $\|r_j\|$, which was estimated without explicitly computing x_j at every iteration step, for the convergence criterion at each iteration of SOR we checked the norm of the true residuals, $\|r_j\| = \|(b - Ax_j)\|$, which was obtained dynamically (i.e., computed during the relaxation sweep, e.g., [7]).

We will now discuss the results. Some aspects of the performance of GMRES without any preconditioning, as well as SOR, will be included as benchmarks. Results from the three meteorological cases are displayed in Fig. 1 for the SOR solutions (the "V" shaped curves) and for the Jacobi preconditioned GMRES (J-GMRES, see Section 4.3). These results are for the 3D equation from the low resolution problem; they show the number of iterations needed for convergence and the CPU time required. The upper x axis of Figs. 1 and 2 shows the relaxation parameter γ for the SOR method while the lower x axis shows the Krylov subspace dimension m for the J-GMRES method. Figure 1 thus compares the performance of the J-GMRES and the SOR methods for our three cases as a function of γ and m for the 3D low resolution problem. Figure 2 presents similar results, but only for case 2, for both the 2D and 3D equations, for the high resolution problem. Note that the dashed lines refer to CPU time; the solid ones refer to the iterations required for convergence.

According to Ashkenazi [3, 5] and many others, the SOR method is one of the simplest to use of the iterative methods. The simplicity of the concept and the ease of programming and operation are, for many users, the essence of SOR. However, as seen from Figs. 1 and 2, the automatic application of a random γ, without due precaution, could greatly worsen the rate of convergence. The figures also show that γ has to be known to at least three digits to guarantee optimum performance, a knowledge which is difficult to obtain in real applications. This sensitivity to γ is a well-known feature of SOR, as pointed out by Wachspress [37, p. 282]. Thus, it is clear that the effectiveness of the SOR method depends strongly upon the selection of an optimization parameter which is not readily available.

The GMRES algorithm can be seen from Figs. 1 and 2 to be far less sensitive to the choice of m than is SOR to the choice of γ. Figure 1 makes it clear that SOR shows sensitivity to γ and to the varying structure of the coefficient matrix A, while the number of iterations for J-GMRES, both with varying m, and A, is relatively constant. This was mentioned in Section 2.

For the high resolution problem, Fig. 2 shows that the SOR method requires less CPU time only when γ is close to γ_{opt}. With m chosen ~ 15, for both the 3D high resolution problem (Fig. 2) and for all three cases for the 3D low resolution problem (Fig. 1), J-GMRES is seen to be competitive with SOR, unless extra effort is expended to find γ_{opt}.

The rate of convergence of the classical iterative methods, such as SOR, depends on the resolution, with higher resolution resulting in slower convergence [7]. We thus checked the convergence rates for the GMRES methods. For $m = 15$, GMRES without preconditioning and for the low resolution 3D problem required 5.97 s to converge, using 168 iterations, or 0.0355 s/iteration (Fig. 5). The 3D high

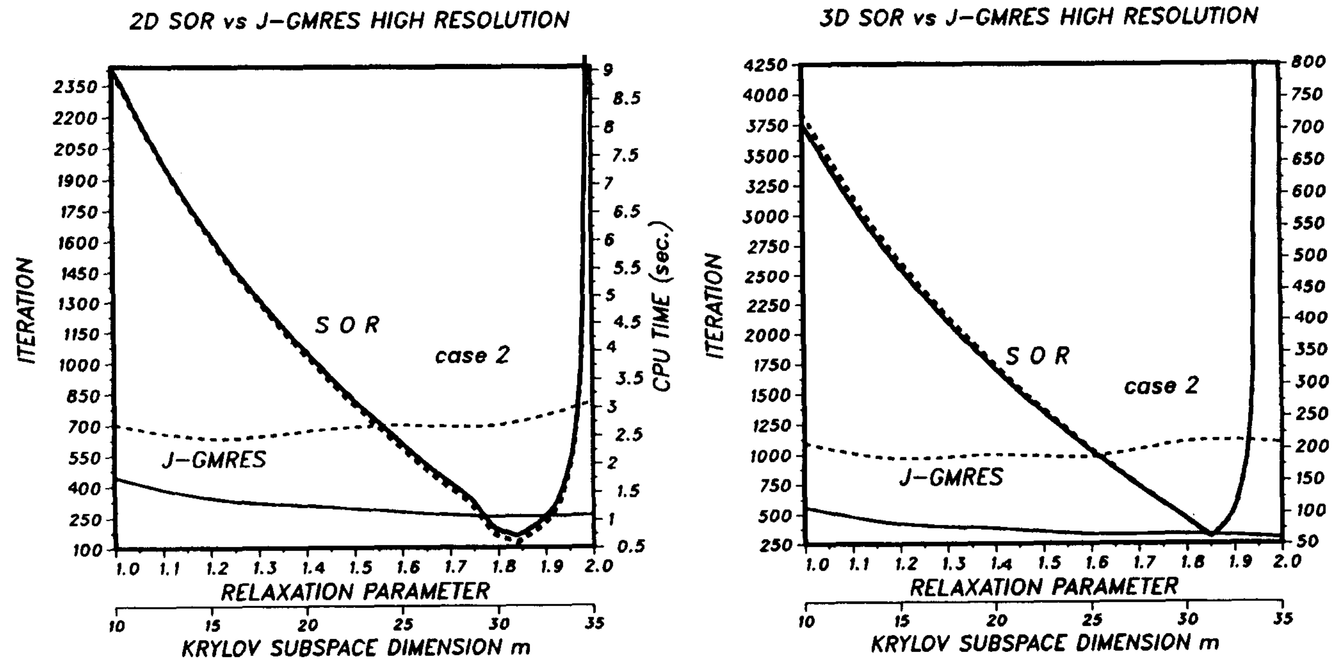


FIG. 2. Performance of the J-GMRES and the SOR methods as a function of γ and m. Dashed lines show the total CPU time, and solid lines show the number of iterations.

resolution problem required 244.16 s to converge, using 608 iterations, or 0.402 s/iterations (Fig. 7), more than a tenfold increase in time/iteration, similar to the ratio of the increase in resolution. The J-GMRES method, also with $m=15$, gives better times, but shows a similar increase in time/iteration for the 3D versus the 2D problem (Figs. 5 and 7). In addition, the convergence rate in general decreases as the Krylov subspace dimension m increases.

4.3. *Preconditioning and Preconditioners*

A suitable preconditioner is crucial in obtaining a rapid convergence of CG-type methods, and choosing good preconditioners for general matrices is an important research issue [10, 11]. According to Nachtigal *et al.*, [21], the convergence rate of GMRES depends on its eigenvalues; thus, by applying a preconditioner to GMRES we wish to cluster the eigenvalues in the right half of the complex plane and/or to improve the distribution of eigenvalues, thereby significantly accelerating the convergence of the method [31, 34].

Since the desired accuracy can often be obtained by more than one method, a major factor in deciding upon an appropriate iterative procedure is the cost of computation, as pointed out by Wachspress [37, p. 12]. In general, an approximate preconditioner M for Eq. (1) is any "simple" matrix that approximates the "essential structure" of A. Thus, one effective class of preconditioner is based on the matrix M^{-1} being a good approximate inverse of A in the sense that $M^{-1}A \approx I$.

In this section, we briefly describe our choices for a preconditioning operator M, and their effects on the cost of computation. We did not want to spend too much effort in finding ways to construct an effective preconditioner. Based on previous studies [10, 24], we used left preconditioning in all numerical experiments. (For more discussion on preconditioning of iterative methods see [4] and its references.)

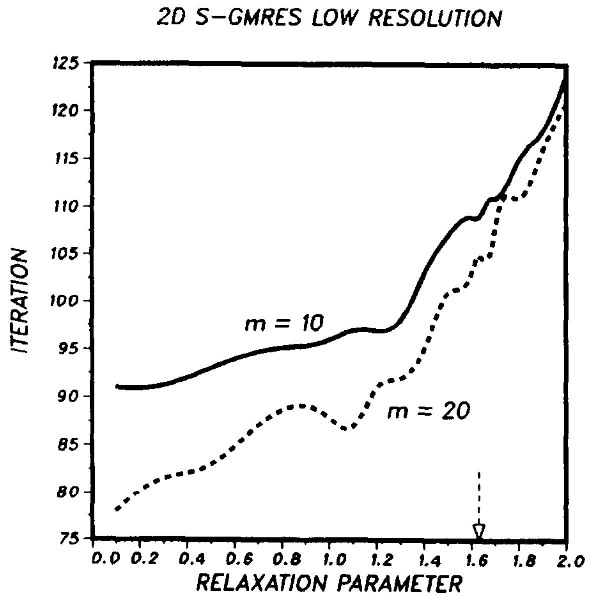
Preconditioning will increase the amount of computation. We have to compute matrix vector products $M^{-1}Av_j$ besides Av_j for each iteration step. Thus, in choosing a preconditioner we must select between methods which usually perform a large number of *cheap* iterations or a small number of *expensive* iterations [24]. The CPU computation time in seconds includes both the time required for the implementation of the preconditioner and the total iterations required for convergence.

We will try some of the basic preconditioners [11], and a relatively new class, incomplete LU factorization (ILU) of the matrix A [19]. The ILU factorization of A, $M=LU$, where L and U are lower and upper triangular matrices, respectively, is based on a modified Gaussian elimination procedure without any pivoting. For the 2D and 3D coefficient matrix A, the preconditioner M for the GMRES is chosen to be:

ILU factorization of the entire matrix A (ILU or ILU-GMRES), and

LU factorization of only the tridiagonal of matrix A (3B or 3B-GMRES).

The basic preconditioners are, however, mostly based on the matrix splittings of A, which is based upon writing $A=D+L+U$, where D is the diagonal, and L and U are strictly lower and upper triangular matrices, respectively.



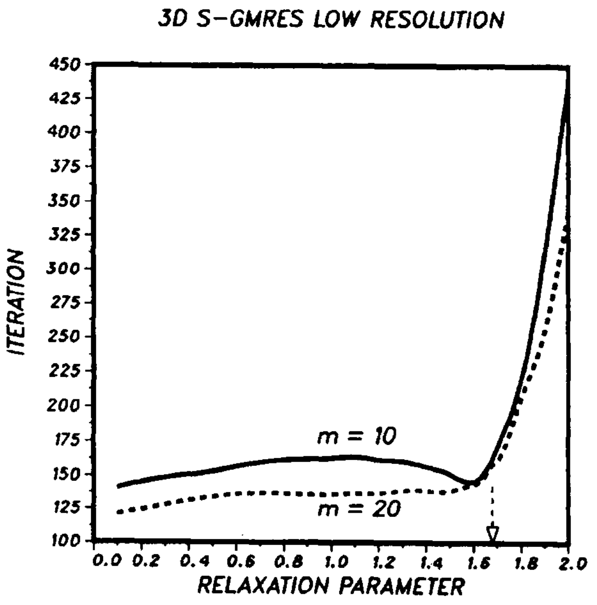


FIG. 3. Performance of the SOR preconditioned S-GMRES method as a function of m and γ. Arrows show γ_{opt} values.

They are chosen to be:

Jacobi (J-GMRES): $M = D$

Gauss-Seidel (G or G-GMRES): $M = D + L$

Successive over-relaxation (S or S-GMRES):

$M = \gamma^{-1}D + L.$

With the choice $\gamma = 1$, the S-GMRES reduces to G-GMRES.

In using SOR as a PDE solver, one needs an estimate of the optimal relaxation parameter γ_{opt}. One cannot, however, calculate γ_{opt} in advance, as discussed in Section 1; in general, we have to rely on "trial and error" to find a good value (see Figs. 1 and 2). Computing of γ_{opt} for each different problem can require significant extra work. Using the SOR method to solve the equations similar to (3) on naturally ordered 2D and 3D grids (i.e., grid points were numbered from left to right and bottom to top), we mostly tried over-relaxation values; $1 \leqslant \gamma < 2$.

We find, however, that the convergence rates as a function of γ are quite different depending on whether SOR is used as a PDE solver or whether γ is used in S-GMRES, for the same problem. In Fig. 3, compare γ_{opt} (shown by the arrows) with the performance of S-GMRES for both 2D and 3D low resolution problems and with two different Krylov subspace dimensions. Figures 4 and 5 also show that G-GMRES (S-GMRES, with $\gamma = 1$) performs better than S-GMRES with γ_{opt} for those problems. For the high resolution problems, Figs. 6 and 7 show that γ_{opt} as well as near γ_{opt} relaxation values degrade the performance of S-GMRES. Similar to Figs. 3–5, Figs. 6 and 7 show that the efficiency of S-GMRES increases as γ approaches 1. The number of required iterations increases as γ approaches 2 and D is effectively reduced in magnitude, see, e.g., Fig. 3. Apparently, the more "diagonally dominant" D is, the better S-GMRES performs. Thus, even if γ_{opt} is known in advance, it will not be useful as a preconditioner.

The SOR preconditioner should therefore be used with care; in general, the best choice for γ may not be γ_{opt}. Our results suggest the use of other preconditioners. (See [2] for a different SOR preconditioner used in a different manner.)

In the implementation of the ILU preconditioner, the explicit calculation of the matrix inverse of M is avoided by solving linear equations in the general form

$$(LU)\, r_j = (b - Ax_j) \quad \text{and} \quad (LU)w = Av_j, \tag{5}$$

instead of inverting and multiplying M. Since no fill-in is allowed in ILU factorization, the solution involves the same number of floating point operations as the sparse matrix–vector product.

Figure 6 shows that the 3B and Jacobi preconditioners have similar efficiencies for the 2D high resolution problem.

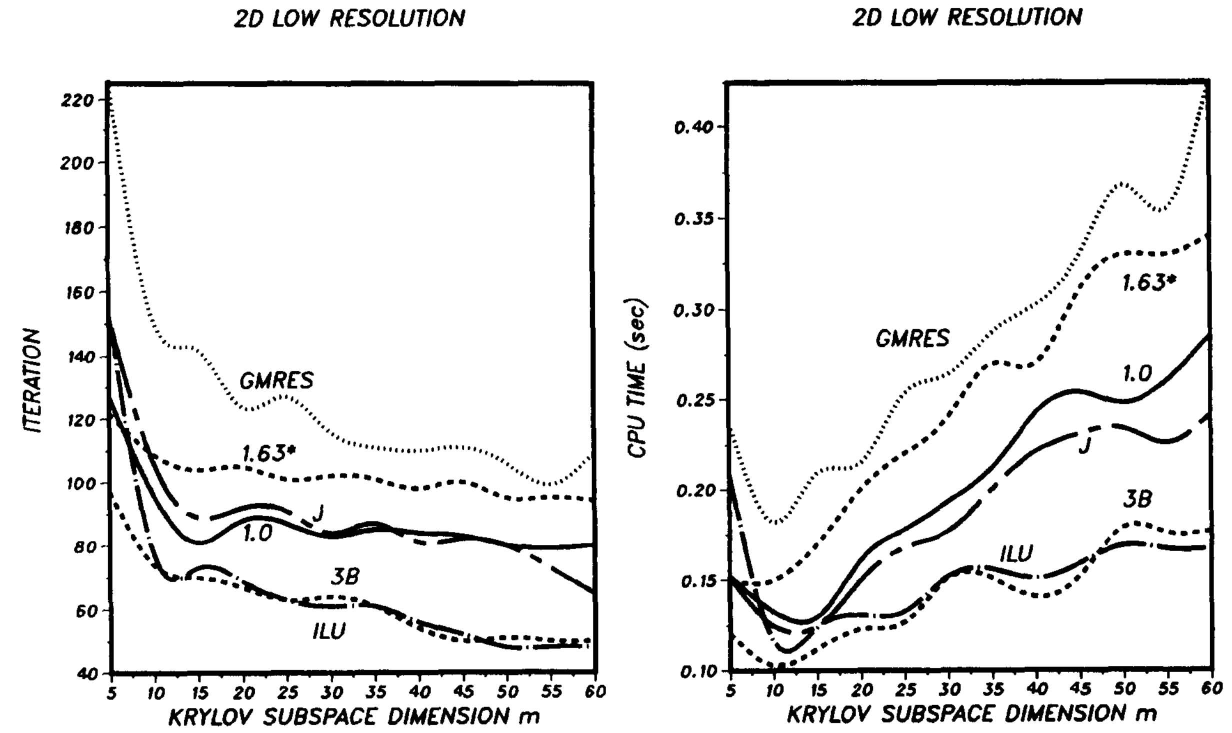


FIG. 4. Performance of the preconditioned GMRES methods as a function of m and γ. Here GMRES represents the GMRES method without any preconditioning. The numbers on the curves show the S-GMRES with different γ's, and * indicates the γ_{opt}.

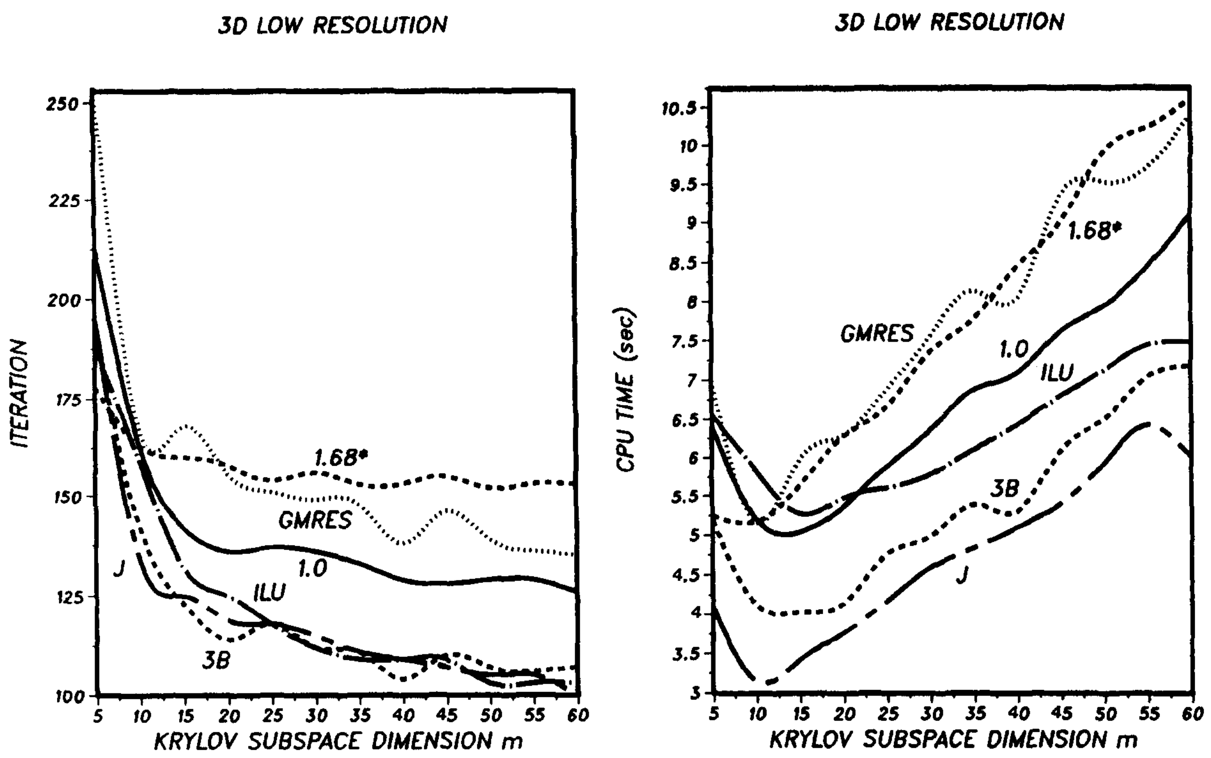


FIG. 5. Same as Fig. 4, but for 3D low resolution.

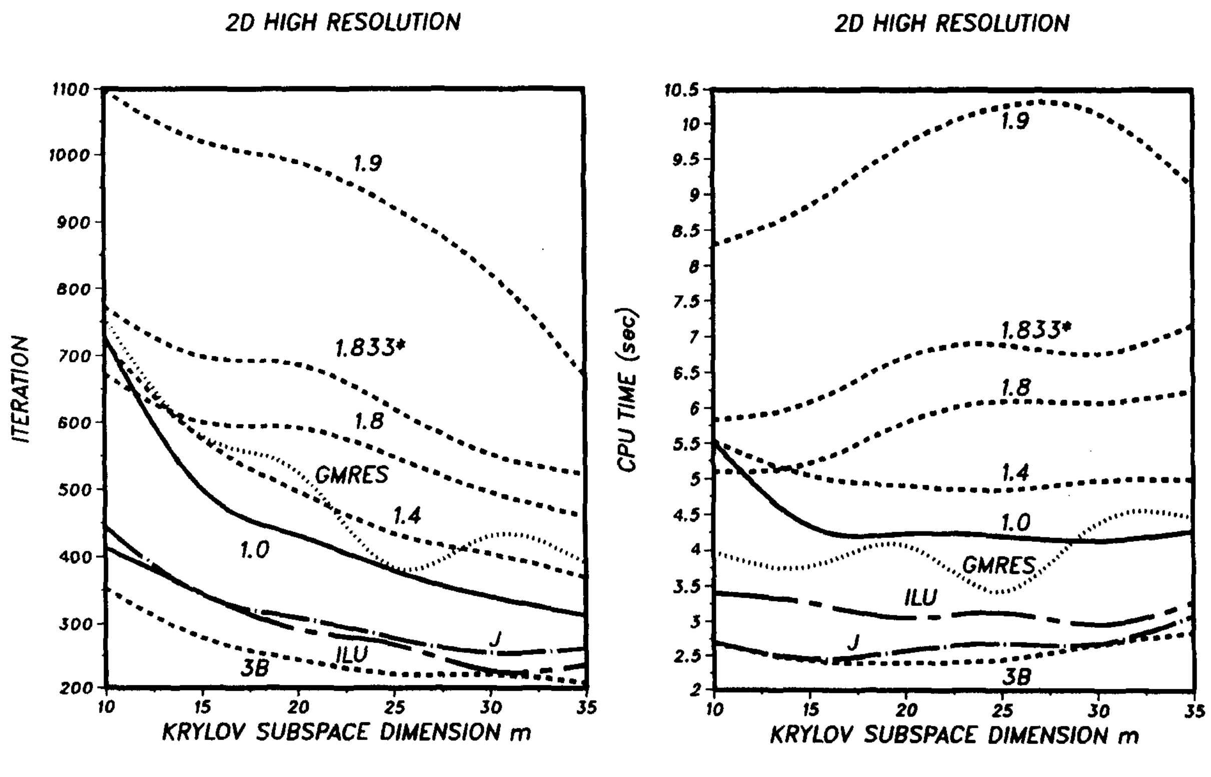


FIG. 6. Same as Fig. 4, but for 2D high resolution.

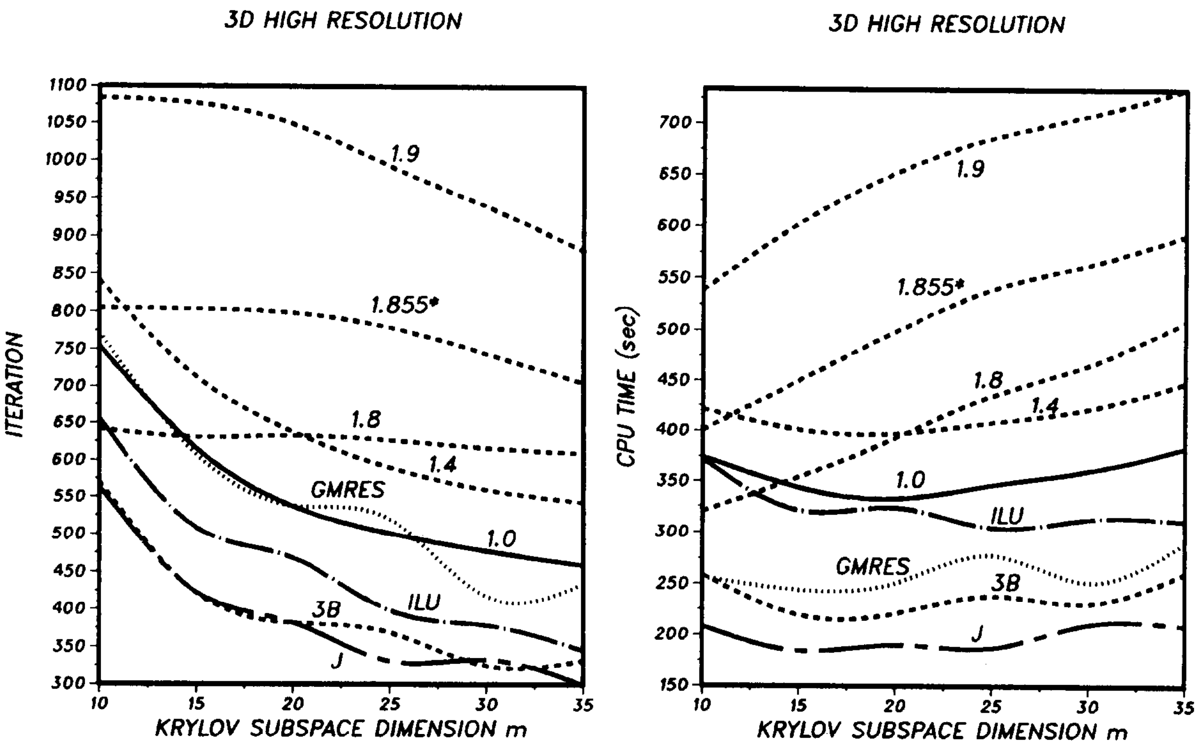


FIG. 7. Same as Fig. 4, but for 3D high resolution.

This is different from the 2D low resolution problem; compare Figs. 4 and 6. In the 3D high resolution problem, the superiority of the Jacobi preconditioner becomes more clear; see Fig. 7. Since J-GMRES appeared to be the best all-around preconditioner we tested, it was used as a comparison to SOR in Figs. 1 and 2. Figure 7 also shows that, although ILU-GMRES requires less iteration steps to converge, it requires more CPU time per iteration than does the GMRES method without preconditioning.

In the low resolution 2D and 3D problems, the required CPU time shows a definite increase with the subspace dimension (Figs. 4 and 5). For the high resolution 2D and 3D problem, the S-GMRES cases tend to behave in a similar manner, while the other cases are not as subspace dimension dependent (Figs. 6 and 7). These results seem inconsistent. We also note from the left-hand portion of these figures, that as the matrix dimension and the grid size increases different preconditioners require almost the same number of iterations but different CPU time per iteration.

The reason behind these may be given by two factors, as indicated in the IBM 3090 VF compiler output from our runs: First, the length of the vectors in the 2D low resolution problem is too short for effective vectorization, and second, in the preconditioner algorithms (except Jacobi), dependence of successive vectorial operations on the same vector elements somewhat inhibits the vectorization of the loop operations.

While most of the kernels of the preconditioned GMRES algorithms are vectorizable, the incomplete factorization and the forward and back solvers in the sparse systems of the preconditioners are recursive operations that cannot be vectorized efficiently. Many alternative types of preconditioners could therefore be considered. Among the others, Aschraft and Grimes [2] developed the *wavefront* technique to vectorize ILU and symmetric SOR preconditioners for the CG method. It remains to be seen if the application of a wavefront technique to our ILU preconditioner would produce a trade-off of its high iteration number with very fast operations, and thus a reduction in CPU time for our 3D high resolution problem.

Our results indicate that the vectorization is more effective and efficient only if the vectors are relatively large, as in the 3D problems. Otherwise, vectorial operations are done only in a scaler manner on the IBM 3090-170J.

5. CONCLUSIONS

In this paper we consider a relatively new and general iterative procedure for solving a large, sparse linear system of equations, specifically, 3D elliptic equations. Although it is difficult to make any definite statements as to an overall "best" method, we proceed by checking the relative merits of the various iterative methods.

First, it is well known that the SOR method requires properties such as diagonal dominance or positive definiteness of the systems of equations, and sometimes symmetry, for the theoretical properties of the method to be valid. Second, while a Krylov subspace dimension is used in GMRES(m), it has been found that its value is far less critical than the relaxation parameter for SOR. Indeed a large enough Krylov subspace dimension, such as $m = 15$, seems adequate, if not ideal, for a range of different problems. Third, the rate of convergence obtained has generally been found to be very good with preconditioned GMRES(m). Like SOR, for GMRES(m) methods the CPU time per grid point required to converge to an approximate solution is [illegible]

GMRES(m), a method not yet widely used, at least in the meteorological community, is better than the SOR, especially for a large system of equations. It is simple to implement, and is virtually parameter free, as well as robust.

Other techniques have limitations. For example, MUDPACK and FISHPACK are limited by the requirement of "quantized" grid numbers in some or all directions. Such restrictions have not been found to be necessary for GMRES(m). For problems which can fit the required grids, these other techniques may be superior to GMRES.

With respect to the choice of a good preconditioner, it seems that the simplest gives the cheapest and best results, especially for large systems. For 2D and low resolution problems, the ILU preconditioners may be preferred to the others. The S-GMRES preconditioner, however, is not recommended. For small and easier 2D problems the low storage methods, such as the optimal SOR method, are still worth using. The iterative preconditioned GMRES(m) algorithm, as a general PDE solver, is a viable alternative for large, sparse, and nonsymmetric linear systems arising from fluid dynamics problems.

Further research for improving the efficiency of preconditioned GMRES(m) could involve choosing a better initial guess vector and a good restart procedure and improving the preconditioners by adapting wavefront or similar techniques.

ACKNOWLEDGMENTS

The authors express their gratitude to Dr. Ronald B. Morgan of the Department of Mathematics for his continuous and enthusiastic support and to Wayne L. Churchill of Computing Services for his excellent assistance throughout the development of this work at the University of Missouri-Columbia. Our thanks also go to the reviewers for their many invaluable suggestions.

REFERENCES

1. J. C. Adams, *Appl. Math. Comput.* **34**, 113 (1989).
2. C. C. Ashcraft and R. G. Grimes, *SIAM J. Sci. Stat. Comput.* **9**, 122 (1988).
3. V. Ashkenazi, in *Large Sparse Sets of Linear Equations*, edited by J. K. Reid (Academic Press, New York, 1971), p. 57.
4. O. Axelsson, *BIT* **25**, 166 (1985).
5. R. F. Boisvert and R. A. Sweet, in *Sources and Development of Mathematical Software*, edited by W. R. Cowell (Prentice–Hall, Englewood Cliffs, NJ, 1984), p. 200.
6. I. S. Duff, in *Sources and Development of Mathematical Software*, edited by W. R. Cowell (Prentice–Hall, Englewood Cliffs, NJ, 1984), p. 165.
7. S. R. Fulton, P. E. Ciesielski, and W. H. Schubert, *Mont. Weather Rev.* **114**, 943 (1986).
8. G. H. Golub and C. F. VanLoan, *Matrix Computations* (Johns Hopkins Univ. Press, Baltimore, MD, 1989).
9. G. H. Golub and D. P. O'Leary, *SIAM Rev.* **31**, 50 (1989).

[illegible] *Computing*, edited by J. C. Diaz (Dekker, New York, 1989), p. 265.

11. L. A. Hageman and D. M. Young, *Applied Iterative Methods* (Academic Press, New York, 1981).
12. W. W. Hager, *Applied Numerical Linear Algebra* (Prentice–Hall, Englewood Cliffs, NJ, 1988).
13. J. G. Haltiner and R. T. Williams, *Numerical Weather Prediction and Dynamic Meteorology* (Wiley, New York, 1980).
14. M. R. Hestenes and E. L. Stiefel, *J. Res. Nat. Bur. Stand.* **49**, 409 (1952).
15. W. D. Joubert, in *Proceedings, Copper Mountain Conf. on Iterative Methods, University of Colorado, April 1–5, 1990* (unpublished).
16. C.-Y. J. Kao and L. H. Auer, *Mont. Weather Rev.* **118**, 1551 (1990).
17. D. R. Kincaid and L. J. Hayes, *Iterative Methods for Large Linear Systems*, edited by D. M. Young and T.-Z. Mai (Academic Press, New York, 1990), p. 293.
18. R. S. Lindzen and H.-L. Kuo, *Mont. Weather Rev.* **97**, 732 (1959).
19. J. A. Meijerink and H. A. van der Vorst, *Math. Comput.* **31**, 148 (1977).
20. S. E. Mudrick, *J. Atmos. Sci.* **39**, 2414 (1982).
21. N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, in *Proceedings, Copper Mountain Conf. on Iterative Methods, University of Colorado, April 1–5, 1990* (unpublished).
22. A. J. Navarra, *Comput. Phys. Commun.* **53**, 321 (1989).
23. A. J. Navarra, *J. Comput. Phys.* **69**, 143 (1987).
24. T. C. Oppe, W. D. Joubert, and D. R. Kincaid, NSPCG User's Guide Version 1.0, CNA-126, April 1988, Univ. of Texas at Austin, Center for Numerical Analysis (unpublished).
25. G. Radicati, Y. Robert, and S. Succi, *J. Comput. Phys.* **80**, 489 (1989).
26. J. K. Reid, in *Large Sparse Sets of Linear Equations*, edited by J. K. Reid (Academic Press, New York, 1971), p. 231.
27. P. J. Roache, *Computational Fluid Dynamics* (Hermosa, Albuquerque, New Mexico, 1976).
28. Y. Saad, *SIAM J. Sci. Stat. Comput.* **10**, 1200 (1989).
29. Y. Saad and M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **7**, 856 (1986).
30. Y. Saad, *SIAM J. Sci. Stat. Comput.* **5**, 203 (1984).
31. P. Sonneveld, *SIAM J. Sci. Stat. Comput.* **10**, 36 (1989).
32. P. N. Swarztrauber and R. A. Sweet, *Trans. Math. Software* **5**, 352 (1979).
33. K. E. Torrance, *J. Fluid Mech.* **95**, 477 (1979).
34. H. A. van der Vorst, *J. Comput. Phys.* **44**, 1 (1981).
35. D. M. Young, *Comput. Phys. Commun.* **53**, 1 (1989).
36. D. M. Young, *Iterative Solution of Large Linear Systems* (Academic Press, New York, 1971).
37. E. L. Wachspress, *Iterative Solution of Elliptic Systems* (Prentice–Hall, Englewood Cliffs, NJ, 1966).