Abstract

We present an iterative method for solving linear systems, which has the property of minimizing at every step the norm of the residual vector over a Krylov subspace. The algorithm is derived from the Arnoldi process for constructing an l_2-orthogonal basis of Krylov subspaces. It can be considered as a generalization of Paige and Saunders' MINRES algorithm and is theoretically equivalent to the Generalized Conjugate Residual (GCR) method and to ORTHODIR. The new algorithm presents several advantages over GCR and ORTHODIR.

GMRES: a Generalized Minimal Residual algorithm for solving nonsymmetric linear systems

Youcef Saad, Martin H. Schultz

Research Report YALEU/DCS/RR-254
May 3, 1985

This work was supported by ONR Grant number N000014-82-K-0184 and NSF Grant number MCS-8106181.

1. Introduction

One of the most effective iterative methods for solving large sparse symmetric positive definite linear systems of equations is a combination of the conjugate gradient method with some preconditioning technique [3, 8]. Moreover, several different generalizations of the conjugate gradient method have been presented in the recent years to deal with nonsymmetric problems [2, 9, 5, 4, 13, 14] and symmetric indefinite problems [10, 3, 11, 14].

For solving indefinite symmetric systems, Paige and Saunders [10] proposed an approach which exploits the relationship between the conjugate gradient method and the Lanczos method. In particular, it is known that the Lanczos method for solving the eigenvalue problem for an NxN matrix A is a Galerkin method onto the Krylov subspace $K_k \equiv \text{span}\{v_1, Av_1, ... A^{k-1}v_1\}$, while the conjugate gradient method is a Galerkin method for solving the linear system Ax=f, onto the Krylov subspace K_k with $v_1 = r_0/||r_0||$. Thus, the Lanczos method computes the matrix representation T_k of the linear operator $P_kA_{|K_k}$, the restriction of P_kA to K_k, where P_k is the l_2-orthogonal projector onto K_k. The Galerkin method for Ax=f in K_k leads to solving a linear system with the matrix T_k which is tridiagonal if A is symmetric. In general, T_k is indefinite when A is and some stable direct method must be used to solve the corresponding tridiagonal Galerkin system. The basis of Paige and Saunders' SYMMLQ algorithm is to use the stable LQ factorization of T_k. Paige and Saunders also showed that it is possible to formulate an algorithm called MINRES using the Lanczos basis to compute an approximate solution x_k which minimizes the residual norm over the Krylov subspace K_k.

In the present paper we introduce and analyse a generalization of the MINRES algorithm for solving nonsymmetric linear systems. This generalization is based on the Arnoldi process [1, 12] which is an analogue of the Lanczos algorithm for nonsymmetric matrices.

Instead of a tridiagonal matrix representing $P_kA_{|K_k}$, as is produced by the Lanczos method for symmetric matrices, Arnoldi's method produces an upper Hessenberg matrix. Using the l_2-orthonormal basis generated by the Arnoldi process, we will show that the approximate solution which minimizes the residual norm over K_k, is easily computed by a technique similar to that of Paige and Saunders. We call the resulting algortihm the Generalized Minimal Residual (GMRES) method. We will establish that GMRES is mathematically equivalent to the generalized conjugate residual method (GCR) [5, 16] and to ORTHODIR [9]. It is known that when A is positive real, i.e. when its symmetric part is positive definite then the generalized conjugate residual method and the ORTHODIR method will produce a sequence of approximations x_k which converge to the exact solution. However, when A is not positive real GCR may breakdown. ORTHODIR on the other hand does not breakdown, but is known to be numerically less stable than GCR [5], although this seems to be a scaling difficulty.

Thus, systems in which the coefficient matrix is not positive real provide the main motivation for developing GMRES. For the purpose of illustration, consider the following 2x2 linear system Ax=f, where

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad ; \quad f = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \text{and} \quad x_0 = 0.$$

The GCR algorithm can be briefly described as follows:

```
1. Start: Set $p_0 = r_0 = f - Ax_0$
2. Iterate: For i=0,1,...until convergence do:
    compute $\alpha_i = (r_i, Ap_i)/(Ap_i, Ap_i)$
    $x_{i+1} = x_i + \alpha_i p_i$
    $r_{i+1} = r_i - \alpha_i Ap_i$
    $p_{i+1} = r_{i+1} + \sum_{j=0}^{i} \beta_j^{(i)} p_j$
    where $\{\beta_j^{(i)}\}$ are chosen so that $(Ap_{i+1}, Ap_j) = 0$,for $0 \leq j \leq i$.
```

If one attempts to execute this algorithm for the above example one would obtain the following results:

1. At step i=0 we get $\alpha_0=0$ and therefore $x_1=x_0$, $r_1=r_0$. Moreover, the vector p_1 is zero.
2. At step i=1, a division by zero takes place when computing α_2 and the algorithm breaks down.

We will prove that GMRES *cannot* breakdown even for problems with indefinite symmetric parts unless it has already converged. Moreover, we will show that the GMRES method requires only half the storage required by the GCR method and 1/3 fewer arithmetic operations than GCR.

In section 2 we will briefly recall Arnoldi's method for generating l_2-orthogonal basis vectors as it is described in [13]. In section 3, we will present the GMRES algorithm and its analysis. Finally, in section 4 we present some numerical experiments.

2. Arnoldi's method

Arnoldi's method [1] which uses the Gram-Schmidt method for computing an l_2-orthonormal basis $\{v_1,v_2,..v_k\}$ of the Krylov subspace $K_k \equiv \mathrm{span}\{v_1, Av_1, ..A^{k-1}v_1\}$ can be described as follows:

Algorithm 1 : Arnoldi

```
1. Start: Choose an initial vector $v_1$ with $||v_1||=1$.

2. Iterate: For j=1,2,..,...do:

        $h_{i,j} = (Av_j, v_i)$ , $i=1,2,..j$,

        $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$ ,

        $h_{j+1,j} = || \hat{v}_{j+1} ||$, and

        $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$.
```

In a practical implementation it is usually more suitable to replace the Gram-Schmidt algorithm of step 2 by the modified Gram-Schmidt algorithm [15]. If V_k is the Nxk matrix whose columns are the l_2-orthonormal basis $\{v_1,v_2,..v_k\}$, then $H_k \equiv V_k^T A V_k$, is the upper kxk Hessenberg matrix whose entries are the scalars $h_{i,j}$ generated by Algorithm 1. If we call P_k the l_2-orthogonal projector onto K_k, and denote by A_k the section of A in K_k, i.e. the operator $A_k = P_k A_{|K_k}$, we notice that H_k is nothing but the matrix representation of A_k in the basis $\{v_1,v_2,...v_k\}$. Thus Arnoldi's original method [1] was a Galerkin method for approximating the eigenvalues of A by those of H_k [1, 12].

In order to solve the linear system

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

by the Galerkin method using the l_2-orthogonal basis V_k, we seek an approximate solution x_k of the form $x_k = x_0 + z_k$, where x_0 is some initial guess to the solution x, and z_k is a member of the

Krylov subspace $K_k \equiv \text{span}\{r_0, Ar_0, \ldots A^{k-1}r_0\}$, with $r_0 = f - Ax_0$. Suppose that k steps of Algorithm 1 are carried out starting with $v_1 = r_0/\|r_0\|$. Then it is easily seen that the Galerkin condition that the residual vector $r_k \equiv f - Ax_k$ be l_2-orthogonal to K_k yields

$$z_k = V_k\, y_k, \text{ where } y_k = H_k^{-1}\|r_0\|\, e_1$$

and e_1 is the unit vector $e_1 \equiv (1,0,0\ldots0)^T$ [13]. Hence we can define the algorithm [13]:

Algorithm 2 : Full Orthogonalization Method

```
1. Start: Choose $x_0$ and compute $r_0=f-Ax_0$ and $v_1=r_0/\|r_0\|$.
2. Iterate: For j=1,2,..,k do:
      $h_{i,j} = (Av_j, v_i)$ , i=1,2,..j,
      $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$,
      $h_{j+1,j} = \| \hat{v}_{j+1} \|$, and
      $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
3. Form the solution:
    $x_k = x_0 + V_k y_k$, where $y_k = H_k^{-1}\| r_0 \| e_1$
```

In practice, the number k of iterations in step 2 is chosen so that the approximate solution x_k will be sufficiently accurate. Fortunately, it is simple to determine a posteriori when k is sufficiently large without having to explictly compute the approximate solution because we can compute the residual norm of x_k thanks to the relation [13, 14]:

$$\| f - A\, x_k \| = h_{k+1,k} |e_k^T\, y_k|. \tag{2}$$

Note, that if the algorithm stops at step k, then clearly it is unnecessary to compute the vector v_{k+1}.

Algorithm 2 has a number of important properties [14]:

- Apart from a multiplicative constant, the residual vector r_k of x_k is nothing but the vector v_{k+1}; Hence, the residual vectors produced by Algorithm 2 are l_2-orthogonal to each other;

- Algorithm 2 does not breakdown if and only if the degree of the minimal polynomial of v_1 is at least k and the matrix H_k is nonsingular;
- The process terminates in at most N steps.

Algorithm 2 generalizes a method developed by Parlett [11] for the symmetric case. It is also known to be mathematically equivalent to the ORTHORES algorithm developed by Young and Jea [9].

A difficulty with the full orthogonalization method is that it becomes increasingly expensive as the step number k increases. There are two distinct ways of avoiding this difficulty. The first is simply to restart the algorithm every m steps. The second is to truncate the l_2-orthogonalization process, by insisting that the new vector v_{i+1} be l_2-orthogonal to only the previous l vectors where l is some integer parameter. The resulting Hessenberg matrix H_k is then banded and the algorithm can be implemented in such a way as to avoid storing all previous but only the l most recent v_i's. The details on this Incomplete l_2-orthogonalization Method (IOM(l)), can be found in [14]. A drawback of these truncation techniques is the lack of any theory concerning the global convergence of the resulting method. Such a theory is difficult because there is no optimality property similar to that of the conjugate gradient method. In the next section we derive a method which we call GMRES based on Algorithm 1 to provide an approximate solution which satisfies an optimality property.

3. The Generalized Minimal Residual (GMRES) algorithm

3.1. The algorithm

The approximate solution of the form x_0+z, which minimizes the residual norm over z in K_k, can in principal be obtained by several known algorithms:

- The ORTHODIR algorithm of Jea and Young [9];
- Axelsson's method [2];
- The Generalized Conjugate Residual method [4, 5].

However, if the matrix is indefinite these algorithms may breakdown or have stability problems. Here we introduce a new algorithm to compute the same approximate solution by using the basis generated by Arnoldi's method, Algorithm 1.

To describe the algorithm we start by noticing that after k steps of Arnoldi's method we have an l_2-orthonormal system V_{k+1} and a (k+1)xk matrix $\overline{H}_k$ whose only nonzero entries are the elements $h_{i,j}$ generated by the method. Thus $\overline{H}_k$ is the same as H_k except for an additional row whose only nonzero element is $h_{k+1,k}$ in the (k+1,k) position. The vectors v_i and the matrix $\overline{H}_k$ satisfy the important relation:

$$A\, V_k = V_{k+1}\overline{H}_k. \tag{3}$$

Now we would like to solve the least squares problem:

$$\min_{z \in K_k} \| f - A[x_0+z] \| = \min_{z \in K_k} \|r_0 - A\, z\|. \tag{4}$$

If we set $z = V_k y$, we can view the norm to be minimized as the the following function of y:

$$J(y) = \|\beta v_1 - A\, V_k\, y\| \tag{5}$$

where we have let $\beta = \| r_0 \|$ for convenience. Using (3) we obtain

$$J(y) = \| V_{k+1} [\, \beta e_1 - \overline{H}_k\, y \,] \|. \tag{6}$$

Here, the vector e_1 is the first column of the (k+1)x(k+1) identity matrix. Recalling that V_{k+1} is l_2-orthonormal, we see that

$$J(y) = \| \beta e_1 - \overline{H}_k\, y \|. \tag{7}$$

Hence the solution of the least squares problem (4) is given by :

$$x_k = x_0 + V_k\, y_k \tag{8}$$

where y_k minimizes the function J(y), defined by (7), over $y \in \mathbf{R}^k$.

The resulting algorithm is similar to the Full Orthogonalization Method, Algorithm 2, described earlier, the only difference being that the vector y_k used in step 3 for computing x_k is now replaced by the minimizer of J(y). Hence we define the following structure of the method:

Algorithm 3 : The Generalized Minimal Residual Method (GMRES)

```
1. Start: Choose $x_0$ and compute $r_0=f-Ax_0$ and $v_1=r_0/||r_0||$.
2. Iterate: For j=1,2,..,k,.. until satisfied do:
      $h_{i,j} = (Av_j, v_i)$ , i=1,2,..j,
      $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j}v_i$,
      $h_{j+1,j} = || \hat{v}_{j+1}||$, and
      $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
3. Form the approximate solution:
    $x_k = x_0 + V_k y_k$,  where  $y_k$ minimizes (7).
```

When using the GMRES algorithm we can easily use the Arnoldi matrix H_k for estimating the eigenvalues of A. This is particularly useful in the hybrid Chebyshev procedure proposed in [6].

It is clear that we face the same practical difficulties with the above GMRES method as with the Full Orthogonalization Method. When k increases the number of vectors requiring storage increases like k and the number of multiplications like $\frac{1}{2}k^2N$. To remedy this difficulty, we can use the algorithm iteratively, i.e. we can restart the algorithm every m steps, where m is some fixed integer parameter. This restarted version of GMRES denoted by GMRES(m) is described below.

Algorithm 4: GMRES(m)

```
1. Start: Choose $x_0$ and compute $r_0=f-Ax_0$ and $v_1=r_0/||r_0||$.
2. Iterate: For j=1,2,..,m do:
      $h_{i,j} = (Av_j, v_i)$ , $i=1,2,..j$,
      $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j}v_i$,
      $h_{j+1,j} = || \hat{v}_{j+1}||$, and
      $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
3. Form the approximate solution:
   $x_m = x_0 + V_m y_m$,  where  $y_m$ minimizes $|| \beta e_1 - \overline{H}_m y ||$, $y \in \mathbf{R}^m$.
4. Restart:
   Compute $r_m=f-Ax_m$; if satisfied then stop
   else compute $x_0:=x_m$, $v_1:=r_m/||r_m||$ and go to 2.
```

Note that in certain applications we will not restart GMRES. Such is the case for example in the solution of stiff ODE's [7] and in the hybrid adaptive Chebyshev method [6].

3.2. Practical implementation

We now describe a few important additional details concerning the practical implementation of GMRES. Consider the matrix $\overline{H}_k$, and let us suppose that we want to solve the least squares problem:

$$\min_{y} || \beta e_1 - \overline{H}_k y || \ .$$

A classical way of solving such problems is to factor $\overline{H}_k$ into Q_kR_k using plane rotations. This is quite simple to implement because of the special structure of $\overline{H}_k$. However, it is desirable to be able to update the factorizaton of $\overline{H}_k$ progressively as each column appears, i.e. at every step of the Arnoldi process. This is important because, as will be seen, it enables us to obtain the residual norm of the approximate solution *without computing* x_k thus allowing us to decide when to stop the process without wasting needless operations.

We now show in detail how such a factorization can be carried out. In what follows, we let F_j represent the rotation matrix which rotates the unit vectors e_j and e_{j+1}, by the angle θ_i:

$$F_j = \begin{array}{cl} \left[\begin{array}{cccccccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & \cdot & & & & & \\ & & & c_j & -s_j & & & \\ & & & s_j & c_j & & & \\ & & & & & 1 & & \\ & & & & & & \cdot & \\ & & & & & & & 1 \end{array}\right] & \begin{array}{l} \\ \\ \\ \\ \longleftarrow \text{row } j+1 \\ \\ \\ \\ \end{array} \end{array}$$

where $c_j \equiv \cos(\theta_j)$, $s_j \equiv \sin(\theta_j)$.

Assume that the rotations F_i, $i=1,..j$ have been previously applied to $\overline{H}_j$ to produce the following upper triangular matrix of dimension $(j+1) \times j$

$$R_j = \left[\begin{array}{cccccc} x & x & x & x & x & x \\ & x & x & x & x & x \\ & & x & x & x & x \\ & & & x & x & x \\ & & & & x & x \\ & & & & & x \\ 0 & . & . & . & . & 0 \end{array}\right] .$$

The letter x stands for a nonzero element. At the next step the last column and row of $\overline{H}_{j+1}$ appear and are appended to the above matrix. In order to obtain R_{j+1} we must start by premultiplying the new column by the previous rotations. Once this is done we obtain a $(j+2)x(j+1)$ matrix of the form

$$\left[\begin{array}{cccccc:c} x & x & x & x & x & x & x \\ & x & x & x & x & x & x \\ & & x & x & x & x & x \\ & & & x & x & x & x \\ & & & & x & x & x \\ & & & & & x & x \\ 0 & . & . & . & . & 0 & r \\ 0 & . & . & . & . & 0 & h \end{array}\right] .$$

The principal upper $(j+1)xj$ submatrix of the above matrix is nothing but R_j, and h stands for

$h_{j+2,j+1}$ which is not affected by the previous rotations. The next rotation will then consist in eliminating that element h in position j+2,j+1. This is achieved by the rotation F_{j+1} defined by

$$c_{j+1} \equiv r \,/\, (\, r^2 + h^2\,)^{1/2}$$

$$s_{j+1} \equiv -\, h \,/\, (\, r^2 + h^2\,)^{1/2}.$$

Note that the successive rotations F_j must also simultaneously be applied to the right side βe_1.

Thus, after k steps of the above process, we have achieved the following decomposition of $\overline{H}_k$:

$$Q_k\, \overline{H}_k = R_k$$

where Q_k is (k+1)x(k+1) and is the accumulated product of the rotation matrices F_j, while R_k is an upper triangular matrix of dimension (k+1)xk, whose last row is zero. Since Q_k is unitary, we have:

$$\begin{aligned} J(y) &= \| \beta e_1 - \overline{H}_k\, y \| \\ &= \| Q_k[\beta e_1 - \overline{H}_k\, y\,] \| \\ &= \| g_k - R_k\, y \;\; \|, \end{aligned} \tag{9}$$

where $g_k \equiv Q_k\, \beta e_1$ is the transformed right hand side. Since the last row of R_k is a zero row, the minimization of (9) is achieved by solving the upper triangular linear system which results from removing the last row of R_k and the last component of g_k. This provides y_k and the approximate solution x_k is then formed by the linear combination (8).

We claimed earlier that it is possible to obtain the residual norm of the approximate solution x_k while performing the above factorization, without explicitly computing x_k. Indeed, notice that from the definition of J(y), the residual norm is nothing but $J(y_k)$ which, from (9), is in turn equal to $\|g_k - R_k y_k\|$. But by construction of y_k, this norm is the absolute value of the last component of g_k. We have proved:

> **Proposition 1:** *The residual norm of the approximate solution x_k is equal to the $(k+1)^{st}$ component of the right hand side g_k obtained by premultiplying βe_1 by the k succesive rotations transforming $\overline{H}_k$ into an upper triangular matrix.*

Therefore, since g_k is updated at each step, the residual norm is available at every step of the QR factorization at no extra cost. This is very useful in the practical implementation of the algorithm because it will prevent us from taking unnecessary iterations while allowing us to avoid the extra computation needed to obtain x_k explicitly.

Next we describe an efficient implementation of the last step of GMRES. If we can show that we can obtain the residual vector as a combination of the Arnoldi vectors $v_1, .. v_m$ and Av_m, then after step m we do not need v_{m+1}. Note that computing $\hat{v}_{m+1}$ and its norm costs (2m+1)N multiplications, so elimination of its computation is a significant saving. Assume that the first m-1 Arnoldi steps have already been performed, i.e. that the first m-1 columns of $\bar{H}_m$ are available as well as the first m vectors v_i,i=1...m. Since we will not normalize v_i at every step, we do not have explicitly the vectors v_i but rather the vectors $w_i = \mu_i v_i$ where μ_i are some *known* scaling coefficients.

All we need in order to be able to compute x_m is the matrix $\bar{H}_m$ and the vectors $v_1, ... v_m$. Since the vectors v_i, i=1,..m, are already known, we need compute only the coefficients $h_{i,m}$, i=1,...m+1. Noting that $h_{i,m} = (Av_m, v_i)$, for $i \leq m$ we see that these first m coefficients can be obtained as follows:

1. Compute Av_m, and
2. Compute the m inner-products (Av_m, v_i), i=1,..m.

Clearly, the scaling coefficients μ_i must be used in the above computations as v_i, i=1,..m, are only available as $w_i = \mu_i v_i$, where $\mu_i = ||w_i||$. This determines the mth column of $\bar{H}_m$ except for the element $h_{m+1,m}$. We wish to compute this coefficient without having to compute w_{m+1}. By definition and the orthogonality of the v_i's

$$h_{m+1,m}^2 = || Av_m - \sum_{i=1}^{m} h_{i,m} v_i ||^2 = || Av_m ||^2 - \sum_{i=1}^{m} h_{i,m}^{\ 2} \tag{10}$$

Hence the last coefficient can be obtained from the $h_{i,m}$'s, i=1,..m, and the norm of Av_m.

Now we will show how to compute the residual vector $r_m = f - Ax_m$ from the v_i's, i=1,..m and Av_m. This computation is necessary only when restarting. From (6) the residual vector can be expressed as:

$$r_m = V_{m+1}[\beta e_1 - \bar{H}_m\, y_m] \tag{11}$$

If we define $t \equiv [t_1, t_2, ... t_{m+1}]^T \equiv \beta e_1 - \bar{H}_m y_m$, then

$$\begin{aligned} r_m &= (\sum_{i=1}^{m} t_i v_i\ ) + t_{m+1} v_{m+1} \\ &= (\sum_{i=1}^{m} t_i v_i\ ) + t_{m+1} \frac{1}{h_{m+1,m}} [Av_m - \sum_{i=1}^{m} h_{i,m} v_i\ ] \\ &= \frac{t_{m+1}}{h_{m+1,m}} Av_m + \sum_{i=1}^{m} (t_i - t_{m+1} h_{i,m} / h_{m+1,m})\, v_i . \end{aligned}$$

It is to be expected that for large m, the alternative expression (10) for $h_{m+1,m}$ would be inaccurate as the orthogonality of the vectors v_i, on which it is based, is likely to be lost [11]. Moreover, in the restarted GMRES, the computation of r_m by (11) may be more time consuming than the explicit use of $r_m=f\text{-}Ax_m$. Therefore, it is not recommended to use the above implementation when m is large.

3.3. Comparison with other methods

From the previous description of GMRES, it is not clear whether or not this algorithm is more effective than GCR or ORTHODIR. Let us examine the computational costs of these three methods. We will denote by NZ the number of nonzero elements in A. We will evaluate the cost of computing the approximation x_k by GMRES. There are several possible implementations but we will refer to the one described in the previous section. If we neglect the cost of computing y_k, which is the solution of a least squares problem of size k, where k is usually much less than N, the total cost of computing x_k by GMRES can be divided in two parts:

- The computation of the Arnoldi vectors v_{j+1}, for j=1,2,...k. The j^{th} step in this loop requires (2j+1)N + NZ multiplications, assuming that the vectors v_i are not normalized but that their norms are only computed and saved. The last step requires only (k+1)N multliplications instead of (2k+1)N, i.e. kN fewer multiplications than the regular cost, as was shown in the previous section. Hence, the total number of multiplications for this part is approximately k(k+2)N + k NZ − k N = k(k+1)N + k NZ
- The formation of the approximate solution $x_0+V_ky_k$, in step 3 requires kN multiplications.

The k steps of GMRES therefore require k(k+2)N+kNZ multiplications. Dividing by the total number of steps k, we see that each step requires (k+2)N+NZ multiplications on the average . In [5], it was shown that both GCR and ORTHODIR require on the average $\frac{1}{2}(3\ k + 5)N + NZ$ multplications per step to produce the same approximation x_k. Therefore with the above implementation GMRES is always less expensive than either GCR or ORTHODIR. For large k the savings will be nearly 1/3.

The above comparison concerns the *nonrestarted GMRES algorithm*. Note that the notation adopted in [5] for the restarted versions of GCR and ORTHODIR differs slightly from ours in that GCR(m) has m+1 steps in each innerloop, while GMRES(m) has only m steps. Hence GMRES(m) is mathematically equivalent to GCR(m-1). When we restart GMRES, we will need the residual vector after the m steps are completed. The residual vector can be obtained either explicitly as f-Ax_m or, as will be described later, as a linear combination of Av_m and the v_i's, i=1,..m. Assuming the latter, we will perform (m+1)N extra multiplications. This will increase the average cost per step by (1+1/m)N to (m+3+1/m)N+NZ. The corresponding cost per step of the restarted GCR and ORTHODIR is $\frac{1}{2}$(3m+5)N+NZ. Thus GMRES(m) is more economical than GCR(m-1) for m>1. Note that the above operation count for GCR(m-1) and ORTHODIR(m-1) does not include the computation of the norm of the residual vector which is required in the stopping criterion while for GMRES, we will show shortly that this norm is available at every step at no extra cost. This remark shows that in fact the algorithms require the same number of operations when m=1.

For GMRES(m), it is clear that all we need to store is the v_i's, the approximate solution, and a vector for Av_i, which means (m+2)N storage locations. For large m, this is nearly half the (2m+1)N storage required by both GCR and ORTHODIR. The comparison of costs is summarized in the following table in which GCR(m-1) and GMRES(m) are the restarted versions of GCR and GMRES, using m steps in each innerloop. Note that the operation count of ORTHODIR(m-1) is identical with that of GCR(m-1) [5].

Table 1

Method	Multiplicatons	Storage

GCR(m-1)
GMRES(m)
	[(3m+5)/2])N + NZ
	(m+3+1/m)N + NZ
		(2m+1)N
		(m+2)N

3.4. Theoretical Aspects of GMRES

A question often raised in assessing iterative algorithms is whether they may breakdown. As we showed in the introduction, GCR can breakdown when A is not positive real, i.e. when its symmetric part is not positive definite. In this section we will show that GMRES cannot breakdown, regardless of the positiveness of A.

Initially, we assume that the first m Arnoldi vectors can be constructed. This will be the case if $h_{j+1,j} \neq 0$, j=1,2,..,m. In fact if $h_{j+2,j+1} \neq 0$, the diagonal element $r_{j+1,j+1}$ of R_{j+1} obtained from the above algorithm satisfies:

$$r_{j+1,j+1} = (c_{j+1}\, r - s_{j+1} h_{j+2,j+1}) = (r^2 + h_{j+2,j+1}^{\;2}\,)^{1/2} > 0.$$

Hence, the diagonal elements of R_m do not vanish and therefore the least squares problem (9) can always be solved, establishing that the algorithm *cannot breakdown* if $h_{j+1,j} \neq 0$, j=1,....m.

Thus the only possible potential difficulty is that during the Arnoldi process we encounter an element $h_{j+1,j}$ equal to zero. Assume that this actually happens at the jth step. Then since $h_{j+1,j} = 0$ the vector v_{j+1} cannot be constructed. However, from Arnoldi's algorithm it is easily seen that we have the relation $A\ V_j = V_j\ H_j$ which means that the subspace K_j spanned by V_j is invariant. Notice that if A is nonsingular then H_j whose spectrum is a part of the spectrum of A is also nonsingular. The quadratic form (5) at the jth step becomes

$$J(y) = \|\ \beta v_1 - A\ V_j\ y\| = \|\ \beta v_1 - V_j\ H_j\ y\| = \|\ V_j[\beta e_1 - H_j\ y\ ]\ \| = \|\ \beta e_1 - H_j\ y\ \|.$$

Since H_j is nonsingular the above function is minimum for $y = H_j^{-1}\ \beta e_1$ and the corresponding minimum norm is zero, i.e., the solution x_j is exact.

To prove that the converse is also true assume that x_j is the exact solution and that x_i, i=1,2...j-1 are not, i.e. $r_j = 0$ but $r_i \neq 0$ for i=0,1,.., j-1. Then $r_j = 0$ and from Proposition 1 we know that the residual norm is nothing but $s_j e_{j-1}^T g_{j-1}$, i.e. the previous residual norm times s_j. Since the previous residual norm is nonzero by assumption, we must have $s_j = 0$ which implies $h_{j+1,j} = 0$, i.e. the algorithm breaks down and $\hat{v}_{j+1} = 0$ which proves the result.

Moreover, it is possible to show that $\hat{v}_{j+1} = 0$ and $\hat{v}_i \neq 0$ i=1,2,..j is equivalent to the property that the degree of the minimal polynomial of the initial residual vector $r_0 = v_1$ is equal to j. Indeed assume the degree of the minimal polynomial of v_1 is j. This means that there exists a polynomial p_j of degree j, such that $p_j(A)v_1 = 0$, and p_j is the polynomial of lowest degree for which this is true. Therefore $K_{j+1} = \text{span}\{v_1, Av_1, .. A^j v_1\}$ is equal to K_j. Hence the vector $\hat{v}_{j+1}$ which is a member of $K_{j+1} = K_j$ and is orthogonal to K_j is necessarily a zero vector. Moreover, if $\hat{v}_i = 0$ for $i \leq j$ then there exists a polynomial p_i of degree i such that $p_i(A)v_1 = 0$ which contradicts the minimality of p_i.

To prove the converse assume that $\hat{v}_{j+1} = 0$ and $\hat{v}_i \neq 0$ i=1,2,..j. Then there exists a polynomial p_j of degree j such that $p_j(A)v_1 = 0$. Moreover, p_j is the polynomial of lowest degree for which this is true, otherwise we would have $\hat{v}_{i+1}=0$, for some $i<j$ by the first part of this proof which is a contradiction.

Proposition 2: *The solution x_j produced by GMRES at step j is exact if and only if the following four equivalent conditions hold:*

- *(1) The algorithm breaks down at step j;*
- *(2) $\hat{v}_{j+1}=0$;*
- *(3) $h_{j+1,j}=0$;*
- *(4) The degree of the minimal polynomial of the initial residual vector r_0 is equal to j.*

This uncommon type of breakdown is sometimes refered to as a happy breakdown in the context of the Lanczos algorithm. Because the degree of the minimal polynomial of v_1 cannot exceed N for an N-dimensional problem, an immediate corollary is:

Corollary 3: For an NxN problem GMRES terminates in at most N steps.

A consequence of Proposition 2 is that the restarted algorithm GMRES(m) does not breakdown. GMRES(m) would therefore constitute a very reliable algorithm if it always converged. Unfortunately this is not always the case, i.e. there are instances where the residual norms produced by the algorithm, although nonincreasing, do not converge to zero. In [5] it was shown that the GCR(m-1) method converges under the condition that A is positive real and so the same result is true for GMRES(m). It is easy to construct a counter example showing that this result does not extend to indefinite problems, i.e. that the method may not converge if the symmetric part of A is not positive definite. In fact it is possible to show that the restarted GMRES method may be stationary. Consider GMRES(1) for the problem $Ax=f$, where

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad ; \quad f = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad x_0 = 0,$$

which we considered in the introduction. The approximate solution x_1 minimizes the residual norm $\| f - A z \|$ where z is a vector of the form $z = \alpha f$. It is easily seen that $x_1=0$. Therefore the algorithm will provide a stationary sequence. Note that this is independent from the problem of breakdown. In fact GMRES will produce the solution in two steps but GMRES(1) never will.

Since the residual norm is minimized at every step of the method it is clear that it is nonincreasing. Intuitively, for m large enough the residual norm will be reduced by a sufficiently small ratio as to ensure convergence. Thus we would expect GMRES(m) to be convergent for sufficiently large m. However, note that ultimately when m=N, the result is trivial, i.e. the method converges in one step. Thus, we will not attempt to show that the method GMRES(m) converges for sufficiently large m. On the other hand it is useful to show that if A is nearly positive real, i.e. when it has a small number of eigenvalues on the left half plane, then m need not be too large for convergence to take place.

In order to analyse this convergence, we let $\mathbf{P}_m$ be the space of all polynomials of degree $\leq$m and let σ represent the spectrum of A. The following result was established in [5] for the GCR algorithm and is a simple consequence of the optimality property.

Proposition 4: *Suppose that A is diagonalizable so that $A=XDX^{-1}$ and let*

$$\epsilon^{(m)} = \min_{p \in \mathbf{P}_m,\, p(0)=1} \; \max_{\lambda_i \in \sigma} |p(\lambda_i)| \, . \tag{12}$$

Then the residual norm provided at the m^{th} step of GMRES satisfies:

$$\| r_{m+1} \| \leq \kappa(X)\, \epsilon^{(m)} \, \|r_0\|,$$

where $\kappa(X)=\|X\| \, \|X^{-1}\|$.

When A is positive real with symmetric part M, the following error bound can be derived from the proposition, see [5]:

$$\| r_m \| \leq [\, 1-\alpha/\beta]^{m/2} \, \|r_0\|,$$

with $\alpha=(\lambda_{min}(M))^2$, $\beta=\lambda_{max}(A^TA)$.

This proves the convergence of the GMRES(m) for all m when A is positive real [5].

When A is not positive real the above result is no longer true but we can establish the following explicit upper bound for $\epsilon^{(m)}$.

Theorem 5: *Assume that there are ν eigenvalues $\lambda_1,\lambda_2,..., \lambda_\nu$ of A with nonpositive real parts and let the other eigenvalues be enclosed in a circle centered at C with C>0 and having radius R with C>R. Then*

$$\epsilon^{(m)} \leq [\frac{R}{C}]^{m-\nu} \max_{j=\nu+1,N} \prod_{i=1}^{\nu} \frac{|\lambda_i-\lambda_j|}{|\lambda_i|} \leq [\frac{D}{d}]^{\nu} [\frac{R}{C}]^{m-\nu} \tag{13}$$

where

$$D= \max_{i=1,\nu \,;\, j=\nu+1,N} | \lambda_i-\lambda_j | \quad \text{and} \quad d= \min_{i=1,\nu} | \lambda_i |$$

Proof: Consider the particular class of polynomials defined by p(z) = r(z) q(z) where r(z) =

$(1-z/\lambda_1)(1-z/\lambda_2) \ldots (1-z/\lambda_\nu)$ and $q(z)$ is an arbitrary polynomial of degree $\leq m-\nu$, such that $q(0)=1$. Clearly, since $p(0)=1$ and $p(\lambda_i)=0$, $i=1,..\nu$, we have

$$\epsilon^{(m)} \leq \max_{j=\nu+1,N} | p(\lambda_j) | \leq \max_{j=\nu+1,N} |r(\lambda_j)| \max_{j=\nu+1,N} |q(\lambda_j)| \; .$$

It is easily seen that

$$\max_{j=\nu+1,N} |r(\lambda_j)| = \max_{j=\nu+1,N} \prod_{i=1}^{\nu} \frac{| \lambda_i-\lambda_j |}{|\lambda_i|} \leq (D/d)^\nu .$$

Moreover, by the maximum principle, the maximum of $|q(z)|$ for z belonging to the set $\{\lambda_j\}_{j=\nu+1,N}$ is no larger than its maximum over the circle that encloses that set. Taking the polynomial $q(z)=[(C-z)/C]^{m-\nu}$ whose maximum modulus on the circle is $(R/C)^{m-\nu}$ yields the desired result□

A similar result was shown by Chandra [3] for the symmetric indefinite case. Note that when the eigenvalues of A are all real then the maximum of the product term in the second part of inequality (13) satisfies

$$\max_{j=\nu+1,N} \prod_{i=1}^{\nu} \frac{|\lambda_i-\lambda_j|}{|\lambda_i|} = \prod_{i=1}^{\nu} \frac{|\lambda_i-\lambda_N|}{|\lambda_i|}$$

where λ_N is the largest eigenvalue of A. A simple consequence of the above theorem is the following corollary.

Corollary 6: Under the assumptions of Proposition 4 and Theorem 5, GMRES(m) converges for any initial vector x_0 if

$$m > \nu \operatorname{Log} \left[\frac{D\,C}{d\,R} \kappa(X)^{1/\nu} \right] / \operatorname{Log} \left[\frac{C}{R} \right] .$$

A few comments are in order. First note that, in general, the upper bound (13) is not not likely to be sharp, and so convergence may take place for m much smaller than would be predicted by the result. Second, observe that the minimal m that ensures converges is related only to the eigenvalue distribution and the condition number of X. In particular, it is independent of the problem-size N. Third, it may very well happen that the minimal m would be larger than N, in which case the information provided by the corollary would be trivial since the method is exact for m=N.

4. Numerical experiments

In this section we report a few numerical experiments comparing the performances of GMRES with other conjugate gradient-like methods. The tests were performed on a VAX-11/780 using double precision corresponding to a unit round off of nearly 6.93×10^{-18}. The GMRES(k) algorithm used in the following tests computes explicitly the last vector v_{k+1} of each outer iteration, i.e. it does not implement the modification described at the end of Section 3.2.

The test problem was derived from the five point discretization of the following partial differential equation which was described in H. Elman's thesis [5]:

$$-(bu_x)_x - (cu_x)_x + d\ u_x + (du)_x + e\ u_y + (eu)_y + fu = g$$

on the unit square, where

$$b(x,y) = e^{-xy}, \quad c(x,y) = e^{xy} \quad d(x,y) = \beta(x + y),$$

$$e(x,y) = \gamma(x+y) \text{ and } f(x,y) = 1./(1+x+y)$$

subject to the Dirichlet boundary conditions u=0 on the boundary. The right hand side g was chosen so that the solution was known to be $xe^{xy}\sin(\pi x)\sin(\pi y)$. The parameters β and γ are useful for changing the degree of symmetry of the resulting linear systems. Note that the matrix A resulting from the discretization remains positive real independent of these parameters.

We will denote by n the number of interior nodes on each side of the square and by $h=1/(n+1)$ the mesh size. In the first example we took n=48, $\gamma=50$ and $\beta=1$. This yielded a matrix of dimension N=2304. The system was preconditioned by the MILU preconditioning applied on the right, i.e. we solved $A\ M^{-1}\ (Mx) = f$ where M was some approximation to A^{-1} provided by an approximate LU factorization of A see [5]. The process was stopped as soon as the residual norm was reduced by a factor of $\epsilon=10^{-6}$. The following plot compares the results obtained for GCR(k), GMRES(k), and ORTHOMIN(k) for some representative values of k.

The plot shows that ORTHOMIN(k) did not converge for k=1 and k=5 on this example. In fact, we observed that it exhibited the same nonconverging behavior for all values of k between 1 and 5. Another interesting observation is that GMRES(5) performed almost as well as GCR(1). Note that the value k=5 yielded the best possible result that was obtained for all reasonable choices of k and similarly GCR(1) corresponded to the best possible performance for GCR(k).

It is worth pointing out that for moderate accuracy ($\epsilon \geq 10^{-2}$), GMRES(5) was slightly better than GCR(1). Finally, we should indicate that the reason why ORTHOMIN performed so badly in this example is that the *preconditioned system* is not positive real. The MILU preconditioning seems to be more prone to such peculiarities than the simpler ILU preconditioning. In fact for this example ORTHOMIN(1) performed very well when the ILU preconditioning was used.

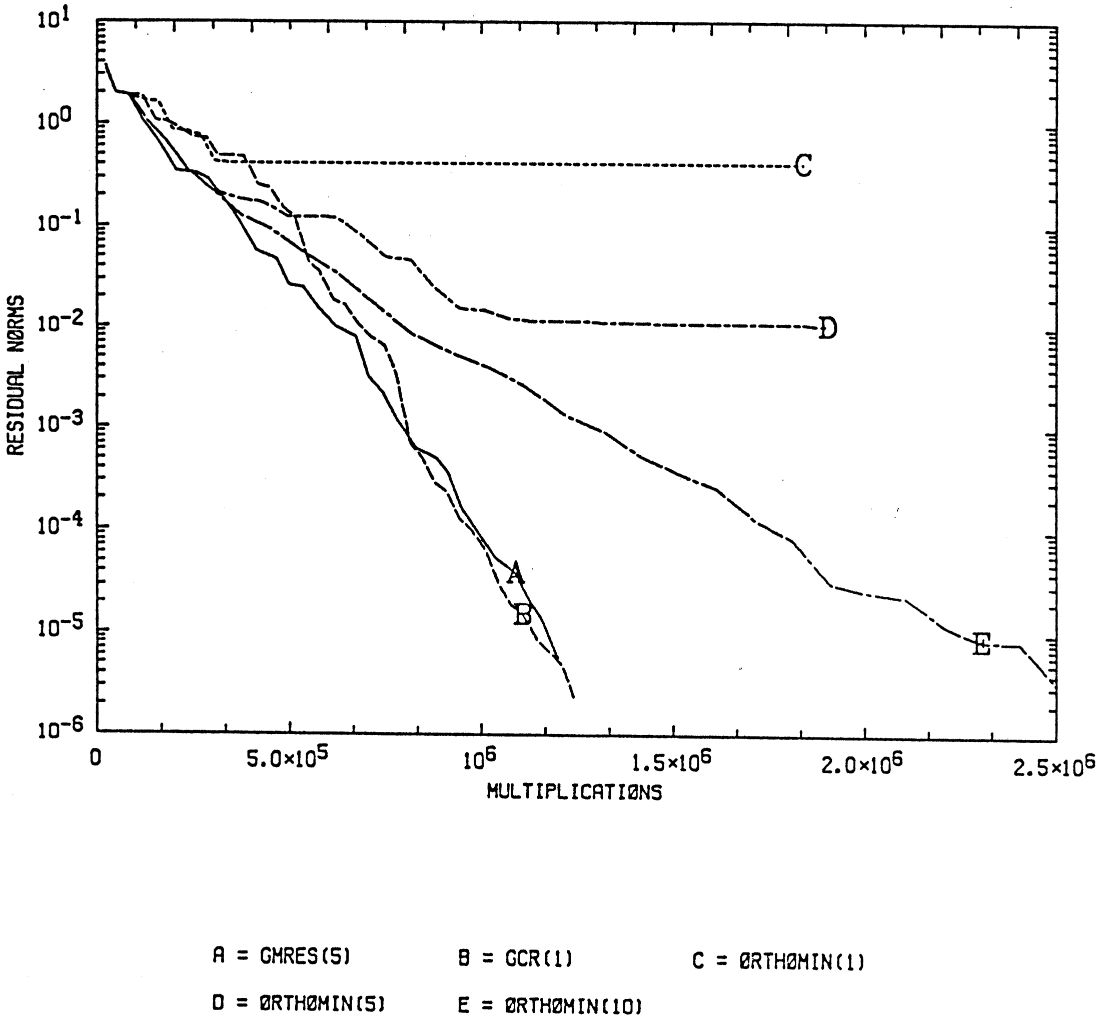

Figure 4-1: n = 48, MILU Preconditioning, γ=50., β=1.

In the next test we took n=18 which yielded a matrix of smaller dimension N=324, and γ=50., β=–20. The main purpose of this experiment was to show that there are instances where using a large parameter m is important. Here again we used the MILU preconditioning and the stopping tolerance was $\epsilon=10^{-6}$. This example was more difficult to treat. ORTHOMIN(k) diverged for all values of k between 1 and 10. Also GCR(1), GCR(2) and GCR(3) diverged as well as their equivalent versions GMRES(k), k=2,3,4. The process GMRES(k) started to converge with k=5 and improved substantially as k increased. The best performance was realized for larger values of k. The following plot shows the results obtained for GMRES(5), GMRES(20) and ORTHOMIN(10). In order to be able to appreciate the gains made by GMRES(20) versus its equivalent version GCR(19), we also plotted the results for GCR(19). Note that we saved nearly 25% in the number of multiplications but also almost half the storage which was quite important here since we needed to keep 22 vectors in memory versus 39 for GCR(19).

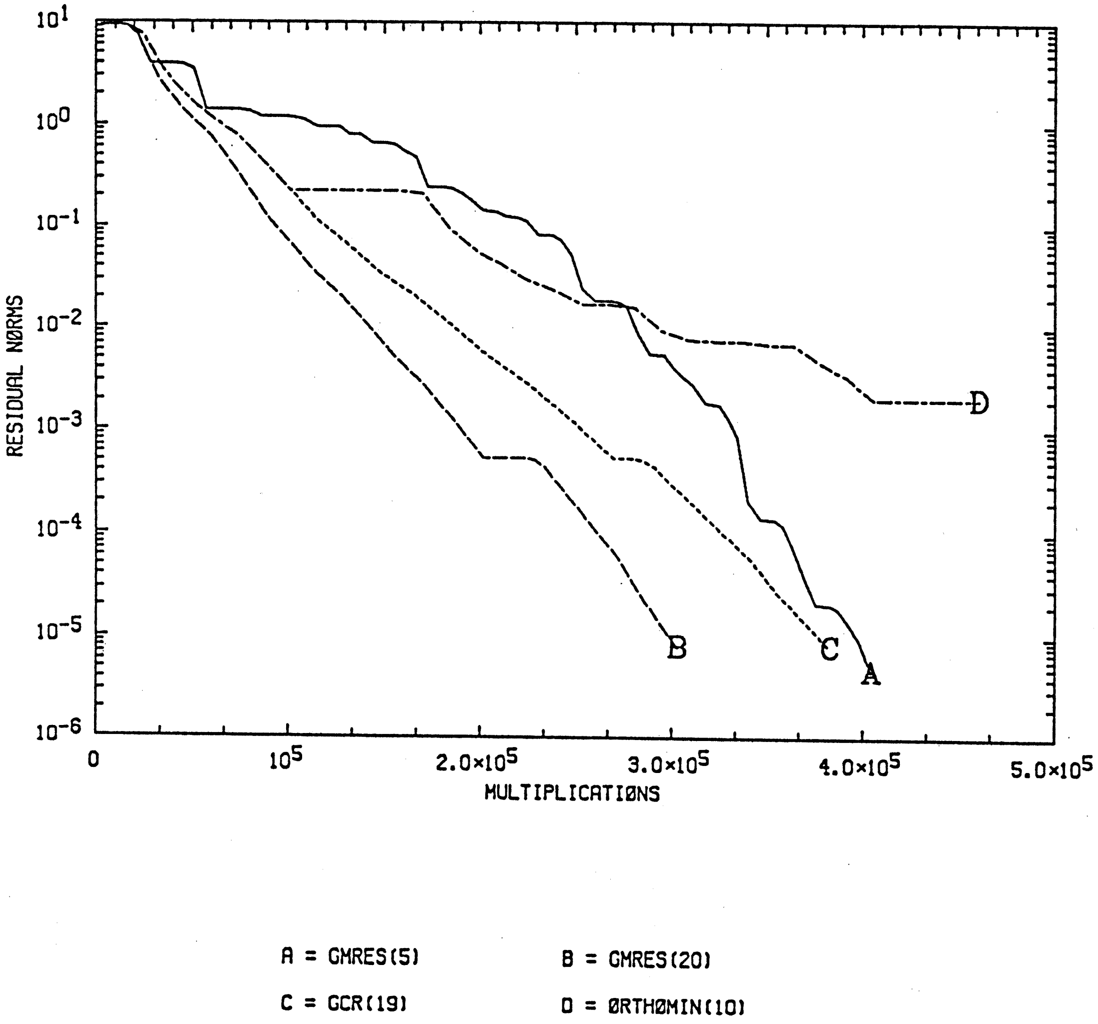

Figure 4-2: n = 18, MILU Preconditioning, γ=50., β=-20.

References

[1] W.E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.* 9:17-29, 1951.

[2] O. Axelsson. Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations. *Lin. Alg. Appl.* 29:1-16, 1980.

[3] R. Chandra. *Conjugate Gradient Methods for Partial Differential Equations.* Ph.D. Thesis, Yale University, Computer Science Dept., 1978.

[4] S.C. Eisenstat, H.C. Elman and M.H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J Num. Anal.* 20:345-357, 1983.

[5] H.C. Elman. *Iterative Methods for Large Sparse Nonsymmetric Systems of Linear Equations.* Ph.D. Thesis, Yale University, Computer Science Dpt., 1982.

[6] H.C. Elman, Y. Saad, P. Saylor. *A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations.* Technical Report YALU/DCS/TR-301, Yale University, 1984.

[7] W.C. Gear and Y. Saad. Iterative solution of linear equations in ODE codes. *SIAM J. Sci. Stat. Comp.* 4:583-601, 1983.

[8] A.L. Hageman and D.M. Young. *Applied Iterative Methods.* Academic Press, New York, 1981.

[9] K.C. Jea and D.M. Young. Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods. *Lin. Alg. Appl.* 34:159-194, 1980.

[10] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations . *SIAM J Numer. Anal.* 12:617-624, 1975.

[11] B.N.Parlett. A new look a the Lanczos algorithm for solving symmetric systems of linear equations. *Lin. Alg. Appl.* 29:323-346, 1980 .

[12] Y. Saad. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Lin. Alg. Appl.* 34:269-295, 1980.

[13] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation* 37:105-126, 1981.

[14] Y. Saad. Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems. *SIAM J. Sci. Stat. Comp.* 5:203-228, 1984.

[15] G.W. Stewart. *Introduction to Matrix Computations.* Academic Press, New-York, 1973.

[16] P.K.W. Vinsome. ORTHOMIN, an iterative method for solving sparse sets of simultaneous linear equations. In *Proceedings of the Fourth Symposium on Resevoir Simulation,* Society of Petroleum Engineers of AIME, , 1976, pp. 149-159.