Computer Science Department
University of Minnesota
Twin Cities
4-192 EE/CSci Building
200 Union Street S.E.
Minneapolis, MN 55455

On Squaring Krylov Subspace Iterative Methods for Nonsymmetric Linear Systems

By

A. T. Chronopoulos and S. Ma

TR 89-67
September 1989
Technical Report

ON SQUARING KRYLOV SUBSPACE ITERATIVE METHODS FOR NONSYMMETRIC LINEAR SYSTEMS

A. T. CHRONOPOULOS and S. MA †

Abstract . The Biorthogonal Lanczos and the Biconjugate Gradients methods have been proposed as iterative methods to approximate the solution of nonsymmetric and indefinite linear systems. Sonneveld [19] obtained the Conjugate Gradient Squared by squaring the matrix polynomials of the Biconjugate Gradients method. Here we square the Biorthogonal Lanczos, the Biconjugate Residual and the Biconjugate Orthodir(2) methods. We make theoretical and experimental comparisons.

Key words . iterative methods, Biorthogonal Lanczos, Lanczos Square, nonsymmetric indefinite linear systems.

1. Introduction Consider the linear system of equations

$$Ax = f$$

where A is a nonsymmetric matrix of order n with symmetric part $M = \frac{(A + A^T)}{2}$ being positive definite or indefinite. D. Luenberger and C. C. Paige and M. A. Saunders [15] have obtained Conjugate Gradient and Lanczos based methods for indefinite symmetric systems. Generalizations to the conjugate gradient method were derived by Concus and Golub [4] and Widlund [21] for nonsymmetric systems with positive real matrix. However, on each iteration an auxiliary symmetric system of equations had to be solved. S. Eisenstat, H. Elman and M. Schultz [7], D.Young and K. Jea [22] devised generalizations to the conjugate residual method, which apply when the matrix of the system is positive real. Y. Saad and M. Schultz obtained GMRES(m) [18], which is based on the Arnoldi iteration but with residual error minimization property. They proved that it applies to general nonsymmetric systems provided that m direction vectors are kept in storage.

†Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455.

The research was partially supported by supported by Univ. of Minnesota Graduate School grant 0350-2104-07, and NSF grants CCR-8722260 and CER DCR-8420935.

Faber and Manteuffel [11], [12] proved that any Krylov subspace based variational method would require to store a number of direction vectors which may be equal to the dimension n of the linear system to ensure termination of the process in at most n steps. Thus all the methods described above seem to need storage of an a priori unspecified number of vectors (in addition to the matrix). This number depends on the nonsymmetry and indefiniteness and condition number of the matrix. The Biorthogonal Lanczos for solving linear systems [17] and the Biconjugate Gradients methods do not have this limitation. In the absence of break down these methods converge in at most n steps with a modest main memory storage requirement.

The Conjugate Gradient Squared method (CGS) [19] was derived from the Biconjugate Gradients (BI-CG) method by simply squaring the residual and direction matrix polynomials. CGS does not need multiplication by the transpose of a matrix. The residual and the directions in CGS are not bi-orthogonal or bi-conjugate respectively. However it can be viewed as the result of polynomial preconditioning with the polynomial varying from iteration to iteration. Thus it turns out that CGS is in practice faster than BI-CG. CGS computes exactly the same parameters as BI-CG and so it has exactly the same breakdown conditions as BI-CG. In fact along the iteration of CGS one can superimpose a BI-CG iteration with additional cost of one matrix vector multiplication but without the need for multiplication by the transpose.

One important advantage of the GCS method over BI-GC is the absence of multiplication by the transpose. This is necessary when applying the linear iterative solver as an inner iteration of a Newton step to solve a nonlinear system of equations: $F(X) = 0$. If the iterative method only requires multiplication by the Jacobian matrix $A = \frac{\partial F}{\partial x}$ then we can approximate it by the Taylor's expansion:

$$Av = \frac{F(x + \varepsilon v) - F(x)}{\varepsilon}$$

This kind of approximation can not be applied to approximate $A^T v$ and explicit evaluation of the jacobian is then required.

The main results in this paper are the derivation of the Lanczos Squared, the Conjugate Residual and Orthodir(2) Squared methods. The Lanczos squared forms the same tridiagonal matrix as the Biorthogonal Lanczos method. The need for multiplication by the matrix transpose has been eliminated. However we have not been able to use the "squared" directions to approximate the solution. This is beacuse unlike the Biorthogonal Lanczos the directions in the Lanczos Square are not part of a Biorthogonal sequence. Thus we have to compute the half of the Biorthogonal Lanczos directions at a cost of an additional matrix multiplication per iteration. The Biorthogonal Lanczos for solving linear systems is useful because its break down conditions are fewer than the Biconjugate Gradients method [17]. We have succeeded in squaring the Biconjugate Orthodir(2) method. This method has the same break down conditions as the Biorthogonal Lanczos method and has modest storage requirements. The Biorthogonal Lanczos needs to store m vectors in secondary memory in order to compute the solution iterate x_m.

In section 2 we describe Orthomin(k) and Orthodir(k) which are two generalizations of the Conjugate Residual method and can be used to obtain approximation to the solution of nonsymmetric positive real linear systems. In section 3 we review the Biorthogonal methods, introduce the Biconjugate Orthodir(2) method and discuss conditions under which these methods converge. In section 4 we obtain the Lanczos and Orthodir(2) Squareds methods. In section 5 we present the ILU preconditioned versions of the Conjugate Gradient Squared and Conjugate Residual Squared and in section 6 we present numerical tests.

2. Orthomin and Orthodir Method

The CR (Conjugate Residual) applied to the SPD (Symmetric Positive Definite) problem minimizes $||r_{i+1}||_2^2$ along the direction p_i in order to determine the steplength a_i in

$$x_{i+1} = x_i + a_i p_i.$$

Also, p_i is made A^TA-orthogonal to p_{i-1}. Symmetry is used to obtain

$$(Ap_i, Ap_j) = 0, \text{ for } i \neq j.$$

Positive definiteness is necessary to guarantee that $a_i = \dfrac{(r_i,Ap_i)}{(Ap_i,Ap_i)} = \dfrac{(r_i,Ar_i)}{(Ap_i,Ap_i)}$ is positive and so there is progress towards the solution in every step. The orthogonality and the norm reducing property of CR guarantee its convergence in at most n iterations.

If A is nonsymmetric but definite then the norm reducing property of CR is still valid but the orthogonality only holds locally. That is p_i is guaranteed to be A^TA-orthogonal only to p_{i-1}. This shortcoming is ameliorated in some of the generalizations of CR.

Algorithm 2.1: (Nonsymmetric Generalization of CR)

$x_0, p_0 = r_0 = f - Ax_0$

For $i = 0$ **Until** Convergence **Do**

$$a_i = \frac{(r_i,Ap_i)}{(Ap_i,Ap_i)}$$

$$x_{i+1} = x_i + a_i p_i$$

$$r_{i+1} = r_i - a_i Ap_i$$

Compute p_{i+1}, Ap_{i+1}

EndFor .

Since $(r_i,Ar_i) \geq 0$ the norms of the residuals is a decreasing sequence. The direction vectors must be constructed to significantly reduce the norm at each step.

(i) Orthomin :

$$p_{i+1} = r_{i+1} + \sum_{j=}^{i} j_i b_j^i p_j$$

$$b_j^i = -\frac{(Ar_{i+1}, Ap_j)}{(Ap_j, Ap_j)}, \quad j \leq i.$$

where $j_i = 0$ for Orthomin(n) or Generalized Conjugate Resisual (CGR) and $j_i = \max(0, i-k+1)$ for Orthomin(k), with $k < n$. We also need to compute the Ap_{i+1}. This can be done either directly

or via the recursion

$$Ap_{i+1} = Ar_{i+1} + \sum_{j=j_i}^{i} b_j^i Ap_j \tag{1}$$

Note that the work for Orthomin(k) is (a) that of GCR, for $j < k-1$ and (b) that of the $(k-1)$-th iteration of GCR, for $k-1 \le j$.

Young and Jea [22] proposed another generalization by defining differently the direction vectors. Unlike GCR, Orthodir is guaranteed to converge even for nonsymmetric indefinite problems.

(ii) Orthodir :

$$p_{i+1} = Ap_i + \sum_{j=j_i}^{i} b_j^i p_j$$

$$b_j^i = -\frac{(A^2 p_i, Ap_j)}{(Ap_j, Ap_j)}, \quad j \le i.$$

where $j_i = 0$ for Orthodir and $j_i = \max(0, i-k+1)$ for Orthodir(k). we need to compute the Ap_{i+1}. This can be done either directly or via the recursion

$$Ap_{i+1} = A^2 p_i + \sum_{j=j_i}^{i} b_j^i Ap_j \tag{2}$$

The work for Orthodir(k) is (a) that of Orthodir, for $j < k-1$ and (b) that of the $(k-1)$-th iteration of Orthodir, for $k-1 \le j$.

3. Biorthogonal Directions Methods

Lanczos [13] introduced a Bi-orthogonal vector generation method and used it to approximate the eigenvalues of unsymmetric matrices. This method can also be used to solve unsymmetric and indefinite linear systems of equations. In this section we simply review the various formulations of the Biorthogonal Lanczos and Biconjugate Gradient Methods.

Algorithm 3.1 The Bi-orthogonal Lanczos Method

$b_1 = d_1 = 0, v_0 = w_0 = 0$

v_1, and w_1 with $(v_1, w_1) = 1$

For $i = 1, \ldots, k$ **Do**

1. $\hat{v} = Av_i - a_i v_i - b_i v_{i-1}$
2. $\hat{w} = A^T w_i - a_i w_i - b_i w_{i-1}$
3. $a_i = (Av_i, w_i)$
4. Select b_{i+1}, d_{i+1}: $b_{i+1}\, d_{i+1} = (\hat{v}, \hat{w})$
5. $v_{i+1} = \dfrac{\hat{v}}{b_{i+1}}$
6. $w_{i+1} = \dfrac{\hat{w}}{d_{i+1}}$

EndFor

The method breaks down if for some index the inner product $(\hat{v}, \hat{w})$ is zero. If the method does not break down then the vectors v_i, w_i are biorthonormal. A simple selection for b_i, d_i is:

$$d_{i+1} = |\, (\hat{v}, \hat{w})\, |^{1/2}, \qquad b_{i+1} = d_{i+1} sign\, (\hat{v}, \hat{w})$$

The bi-orthogonal Lanczos method can be used to solve a linear system of equations. We select $v_1 = \dfrac{r_0}{||r_0||}$ and $w_1 = v_1$. Because of the biorthogonality we obtain: $W_m{}^T A V_m = T_m$, where the tri-diagonal matrix $T_m = triag[d_{i+1}, a_i, b_{i+1}]$, with $i = 1, \ldots, m$ and $V_m = [v_1, \ldots, v_m]$, $W_m = [w_1, \ldots, w_m]$. Now, the approximate solution to $Ax = f$ is given by :

$$x_m = x_0 + V_m z_m$$

where $T_m\, z_m = ||r_0||\, e_1$. The residual norm can be obtained from the formula:

$$||f - Ax_m|| = ||\, Av_m - a_m v_m - b_m v_{m-1}\, ||\; |e_m^T z_m|$$

Next we describe the Biconjugate Gradient Method [9] which uses a recurrence for forming the residuals and thus one does not need to store (in secondary memory) m direction vectors to compute x_m as in the Bi-orthogonal Lanczos case.

Algorithm 3.2 The Biconjugate Gradients Method (BI-CG)

Select r_0 and $\bar{r}_0$

Take $p_0 = r_0, \bar{p}_0 = \bar{r}_0$

For $i = 1, \ldots, k$ **Do**

1. $a_i = \dfrac{(\bar{r}_i, r_i)}{(\bar{p}_i, Ap_i)}$
2. $r_{i+1} = r_i - a_i A p_i$
3. $\bar{r}_{i+1} = \bar{r}_i - a_i A^T \bar{p}_i$
4. $p_{i+1} = r_{i+1} + b_i p_i$
5. $\bar{p}_{i+1} = \bar{r}_{i+1} + b_i \bar{p}_i$
6. $b_i = -\dfrac{(\bar{r}_{i+1}, Ap_i)}{(\bar{p}_i, Ap_i)}$

EndFor

The scalars a_i and b_i are selected to force respectively biorthogonality of the residuals $\{\bar{r}_i, r_i\}$ and biconjugacy (with respect to the matrix A) of the directions $\{\bar{r}_i, r_i\}$. The following theorem [9] states the relationships of the vectors generated by the Biconjugate Gradients Method.

Theorem 3.1: Provided that the Biconjugate Gradients Method does not break down, then for $0 \le j < i$

$$(\bar{r}_i, r_j) = (r_i, \bar{r}_j) = 0$$

$$(\bar{p}_i, Ap_j) = (p_i, A^T \bar{p}_j) = 0$$

$$(\bar{r}_i, p_j) = (r_i, \bar{p}_j) = 0$$

Proof: [9].

Remark 3.1: It follows from this theorem that the residual r_m is zero for $m \leq N$, where N is the dimension of A, provided that the method does not break down.

Corollary 3.1: The scalars generated by the Biconjugate Gradients Method can be computed in one of the following ways provided that the method has not failed:

$$a_i = \frac{(\bar{r}_i, r_i)}{(\bar{p}_i, Ap_i)} = \frac{(\bar{p}_i, r_i)}{(\bar{p}_i, Ap_i)} = \frac{(\bar{p}_i, r_0)}{(\bar{p}_i, Ap_i)}$$

$$b_i = -\frac{(\bar{r}_{i+1}, Ap_i)}{(\bar{p}_i, Ap_i)} = \frac{(\bar{r}_{i+1}, r_{i+1})}{(\bar{r}_i, r_i)} = \frac{(\bar{p}_{i+1}, r_0)}{(\bar{p}_i, r_0)}$$

Proof: The first equalities are from theorem 3.1. The second and third equality for a_i and the third equality for b_i can be obtained from theorem 3.1 and the defining equations for $\bar{p}_i$ and r_i. The second equality for b_i needs $a_i \neq 0$ and $Ap_i = \frac{(r_{i+1} - r_i)}{a_i}$. ■

The initial residual vector is $r_0 = f - Ax_0$. The choice of the conjugate residual vector $\bar{r}_0$ varies. We can derive an algorithm similar to 3.2 which is the biconjugate version of the Conjugate Residual algorithm. This algorithm will be called the Biconjugate Residual method.

Remark 3.2: The Biconjugate Residual (Bi-CR) method is a special case of the Biconjugate Gradients with initial biconjugate residual $A^T\bar{r}_0$

Proof : It can be easily checked that the biconjugate direction and residual will be

$$A^T\bar{r}_i, \quad A^T\bar{p}_i.$$

where $\bar{r}_0$ and $\bar{p}_0$ are as in the BI-CG method. ■

In general one can select $\bar{r}_0 = Mr_0$ where M is any square matrix. To obtain $\bar{r}_i = Mr_i$, and $\bar{p}_i = Mp_i$ we need the following condition

$$A^TM = MA^T$$

An interesting choice for M is A^{kT} or $\lambda(A^T)$ a nonzero polynomial in A^T.

We note that although the method does not break down for $(\bar{r}_i, r_i) = 0$ it becomes stationary because $r_{i+1} = r_i$ and thus p_{i+1} is a linear combination of $\{p_i, \ldots, p_0\}$. We can modify algorithm 3.2 so that it never becomes stationary and the break down conditions are the same as in algorithm 3.1. This new algorithm is the biorthogonal version of the Orthogonal Directions (OD) algorithm derived by Fletcher [9]. Since the OD algorithm is essentially Orthodir(2) we name this algorithm the Biorthogonal Orthodir(2).

Algorithm 3.3 The Biorthogonal Orthodir(2). (Bi-OD)

Select, x_0, r_0 and $\bar{r}_0$

Take, $p_0 = r_0$, $\bar{p}_0 = \bar{r}_0$, $b_0 = 0$.

For $i = 0$ **Until** Convergence **Do**

1. $\hat{a}_i = \dfrac{(r_i, A^T\bar{p}_i)}{(A^T\bar{p}_i, Ap_i)}$

2. $r_{i+1} = r_i - \hat{a}_i Ap_i$

3. $x_{i+1} = x_i + \hat{a}_i p_i$

4. $a_i = \dfrac{(A^2p_i, A^T\bar{p}_i)}{(A^T\bar{p}_i, A\bar{p}_i)}$

5. $b_i = \dfrac{(A^2p_i, A^T\bar{p}_i)}{(A^T\bar{p}_{i-1}, A\bar{p}_{i-1})}$

6. $p_{i+1} = Ap_i - a_i p_i - b_i p_{i-1}$

7. $\bar{p}_{i+1} = A^T\bar{p}_i - a_i\bar{p}_i - b_i\bar{p}_{i-1}$

8. $Ap_{i+1} = A^2p_i - a_i Ap_i - b_i Ap_{i-1}$

9. $A^T\bar{p}_{i+1} = A^{2T}\bar{p}_i - a_i A^T\bar{p}_i - b_i A^T\bar{p}_{i-1}$

EndFor

We note that this algorithm breaks down for any $(A^T\bar{p}_i, A\bar{p}_i) = 0$. This is also true of the Biconju-

gate Residual method. However, this algorithm will never become stationary when some $a_i = 0$. The biconjugate residual vectors $\bar{r}_i$ are not computed in this algorithm. The break down conditions of this algorithm are similar to the Biorthogonal Lanczos method.

The following theorem is the analogue of Theorem 3.1 and also states that Biconjugate Residual is a special case of Biconjugate Orthodir(2).

Theorem 3.2: The direction and the residual vectors in the Biconjugate Orthodir(2) satisfy the following relations, for $j < i$:

$$(\bar{r}_i, r_j) = (r_i, A^T \bar{r}_j) = 0$$

$$(A^T \bar{p}_i, Ap_j) = (Ap_i, A^T \bar{p}_j) = 0$$

$$(\bar{r}_i, Ap_j) = (r_i, A^T \bar{p}_j) = 0$$

Also, if the Biconjugate Residual does not break down it produces the same x_i iterates as the Biconjugate Orthodir(2).

Proof: The biconjugacy relations can be easily proven by induction. For the last statement we must prove that in Bi-CR the directions and steplengths are the same as in Bi-Orthodir(2). If we use the equations $p_{i+1} = r_{i+1} + b_i p_i$, $r_{i+1} = r_i - a_i A p_i$ and $p_i = r_i + b_{i-1} p_{i-1}$ by eliminating the terms r_{i+1} and r_i we obtain the expression :

$$p_{i+1} = Ap_i - \lambda_i p_i - \mu_i p_{i-1}$$

for some parameters λ_i, μ_i. Similar equations can be obtained for $\bar{p}_{i+1}$. The parameters λ_i, μ_i are determined by the biconjugacy relations of the direction vectors and so they are the same as in algorithm 3.2. The parameters $\hat{a}_i$ are the same as the steplengths in Bi-CR from corrolary 3.1. ■.

The conditions for feasibility of the Biorthogonal Lanczos and Biconjugate Orthodir(2) methods can be expressed in terms of the matrices of moments of the initial residual r_0. Let us consider the any vector $u = p(A)v_1$, where $p(A)$ is a polynomial of degree at most $m-1$. Let us denote the space of polynomials with degree at $m-1$ by P_{m-1}. The bilinear form

$(p, q) = (p(A)v_1, q(A^T)w_1)$ on the space P_{m-1} has all the properties of an inner product except the positive definiteness. Let us consider the question whether a sequence of orthogonal polynomials can be constructed with respect to this bilinear form. We need the following lemma, which can be found in [3].

Lemma 3.1: The first m orthogonal polynomials with respect to the above bilinear form can be constructed if and only if

$$\det(M_k) \neq 0, \quad k = 1, \ldots, m.$$

where $M_k = \{(A^{i+j-2}v_1, w_1), i,j = 1, \ldots, k\}$ is the a moments matrix. ■

This lemma has been used by Saad in [17] to obtain the conditions under which the Biorthogonal Lanczos and the Biconjugate Gradients methods give an approximate solution to the linear system of equations.

Proposition 3.1: Let M_k and M'_k be the moment matrices of dimension k with entries $m_{ij} = (A^{i+j-2}v_1, v_1)$ and $m_{ij} = (A^{i+j-1}v_1, v_1)$ respectively. Then the approximate solution x_m can be computed by the Biorthogonal Lanczos if and only if (i) holds and (ii) holds for only for $k = m$. Also, the first m steps of the Biconjugate Gradients method can be carried out if and only if (i) and (ii) hold for all $k = 1, \ldots, m$.

(i) $\det(M_k) \neq 0, \quad k = 1, \ldots, m.$

(ii) $\det(M'_k) \neq 0, \quad 1 \leq k \leq m.$

Proof: [17] ■

It follows from this proposition that the Biorthogonal Lanczos method is less likely to break down. However, it needs to store (in secondary memory) all the vectors $v_1, \ldots, v_m$ in order to compute the approximate solution x_m at termination of the iteration. This shortcoming is removed in the Biorthogonal Orthodir(2) method. The next proposition contains conditions under which this method will give the approximate solution in less than n iterations.

Proposition 3.2: Let M_k be the moment matrices of dimension k with entries $m_{ij} = (A^{i+j}r_0, r_0)$.

Then the Biorthogonal Orthodir(2) will produce the approximate solution x_m if and only if

(i) $\det(M_k) \neq 0, \quad k = 1, \ldots, m.$

Proof : Since the method only breaks down if any of the denominators $(A^T\bar{p}_i, Ap_i) = 0$ using lemma 3.1 we obtain the result. ■

Since the "square " versions and the biorthogonal methods compute the same parameters they have the same break down conditions.

4. The Lanczos and Orthodir(2) Squared

In the first part of this section we will derive the Lanczos Square Method by squaring the Biorthogonal Lanczos method matrix polynomials and obtaining a simple recurrence equation for generating them. Since we have not succeeded in finding a way to approximate the solution to the linear system using these "squared" directions we will have to essentially compute the vectors $V_i = \{v_1, \ldots, v_m\}$ of the Biorthogonal Lanczos in order to achieve this. This of course results in an additional matrix vector multiplication per iteration. However the there is no need to multiply by the transpose of the matrix.

Let us use some notation for the matrix polynomials of the vectors in the Biorthognal Lanczos (BI-L) method. We denote by $\phi_i(A)v_1$ and $\psi_i(A^T)w_1$ the vectors v_i and w_i respectively. We note that the parameters in BI-L can be expressed in terms of these polynomials: $a_i = (v_1, A(\phi_i\psi_i)(A)w_1)$ and $b_i = (v_1, \phi_i\psi_i(A)w_1)$, where the polynomial is scaled before computing a_i. Therefore we must find a recursion to compute $\Phi_i = \phi_i\psi_i$ and $A\Phi_i$. Multiplication of the polynomial ϕ_i and ψ_i from equations 1. and 2. of algorithm 3.1 yield

$$\hat{\Phi}_{i+1} = A[(A\phi_i\psi_i - a_i\phi_i\psi_i) - (b_i\phi_{i-1}\psi_i + d_i\phi_i\psi_{i-1})] + a_i^2\phi_i\psi_i +$$

$$a_i(b_i\phi_{i-1}\psi_i + d_i\phi_i\psi_{i-1}) + b_id_i\phi_{i-1}\psi_{i-1}$$

In order to be able to compute $\hat{\Phi}_i$ recursively we need to compute recursively $\phi_i\psi_{i+1}$, $\phi_i\psi_{i+1}$.

>From 1. and 2. of algorithm 3.1 we obtain:

$$\Theta_{i+1} = \phi_i\psi_{i+1} = (A\phi_i\psi_i - a_i\phi_i\psi_i) - d_i\phi_i\psi_{i-1}$$

$$\overline{\Theta}_{i+1} = \phi_{i+1}\psi_i = (A\phi_i\psi_i - a_i\phi_i\psi_i) - b_i\phi_{i-1}\psi_i$$

We note that we should scale $\hat{\Phi}_i$ by $\frac{1}{b_{i+1}d_{i+1}}$, Θ_{i+1} by $\frac{1}{b_{i+1}}$ and $\overline{\Theta}_{i+1}$ by $\frac{1}{d_{i+1}}$. However since all the instances of Θ_{i+1} and $\overline{\Theta}_{i+1}$ are multiplied by the reciprocals of the scale factors we can simply drop these scale factors. Also, since b_i or d_i only appear in the form b_id_i we can assume that $b_i = d_i$. This could introduce complex arithmetic in the BI-CG case as b_i might be imaginary but for LS only real arithmetic is needed. Then we also have $\overline{\Theta}_i = \Theta_i$. Thus we obtain the following simplified expression

$$\hat{\Phi}_{i+1} = (A - a_i)^2\Phi_i - 2(A - a_i)\Theta_i + b_i^2\Phi_{i-1} \qquad (4.1)$$

and

$$\Theta_{i+1} = (A\Phi_i - a_i\Phi_i) - \Theta_i \qquad (4.2)$$

We next present the Lanczos Squared method at first in polynomial form then in vector form. The inner product $[\phi, \psi]$ with the matrix polynomials (in A) ϕ and ψ stands for the inner product $(\phi(A)u_1, \phi(A)w_1)$.

Algorithm 4.1 The Lanczos Squared (LS) Method

$\beta_1 = 0, \quad \Theta_1 = \Phi_1 = 1$

For $i = 1, \ldots, k$ **Do**

$a_i = [1, A\Phi_i]$

$Y_i = (A - a_i)\Phi_i$

$\hat{Y}_i = AY_i$

$\hat{\Phi}_{i+1} = \hat{Y}_i - a_iY_i - 2(\hat{\Theta}_i - a_i\Theta_i) + \beta_i\Phi_i$

$\beta_{i+1} = [1, \hat{\Phi}_{i+1}]$

$$\Phi_{i+1} = \frac{\hat{\Phi}_{i+1}}{\beta_{i+1}}$$

$$\Theta_{i+1} = Y_i - \Theta_i$$

$$\hat{\Theta}_{i+1} = \hat{Y}_i - \hat{\Theta}_i$$

EndFor

Let us use the notation $\hat{u}_i$ and u_i for the (un)scaled vector forms of $\Phi_i(A)$. Also, p_i, q_i denote $A\Phi_i(A)$, $\Theta_i(A)$. We then have the following vector form of the LS method.

Algorithm 4.2 The Lanczos Squared (LS) Method

$b_1 = d_1 = 0, v_0 = w_0 = 0$

v_1, and w_1 with $(v_1, w_1) = 1$

$z_1 = p_1$

For $i = 1, \ldots, k$ **Do**

$p_i = Au_i$

$a_i = (u_1, p_i)$

$y_i = p_i - a_i u_i$

$p_i = Ay_i$

$\hat{u}_{i+1} = p_i - a_i y_i - 2(z_i - a_i q_i) + \beta_i u_i$

$\beta_{i+1} = (u_1, \hat{u}_{i+1})$

$$u_{i+1} = \frac{\hat{u}_{i+1}}{\beta_{i+1}}$$

$q_{i+1} = y_i - q_i$

$z_{i+1} = p_i - z_i$

EndFor

We note that this algorithm only generated the same tridiagonal matrix T_m of the Biorthogonal Lanczos method without use of the transpose of the matrix A. The vectors $u_i = \Phi(A)_i v_1$ are the in polynomial form the "squares" of the Lanczos vectors $v_i = \phi(A)_i v_1$, as $\Phi_i = \phi^2$. Since the vectors $u_i, \bar{u}_i$ are not biorthogonal it does not seem easy to define the vector z_m so that $x_m = x_0 + U_m z_m$ is an approximation to the solution of the linear system. Therefore the only way to obtain the solution is by actually forming the vectors $\{v_1, \ldots, v_m\}$ of the Biorthogonal Lanczos, for an additional matrix vector product per iteration. The vectors $\{w_1, \ldots, w_m\}$ are only required for forming the tridiagonal matrix T_m. However, this method is exactly the Biorthogonal Lanczos with the need to multiply by the matrix transpose removed.

We next square the Biconjugate Orthodir(2) method. Let us use the notation $p_i = \phi_i(A) r_0$, $\bar{p}_i = \phi_i(A^T)\bar{r}_0$ and $r_i = \psi_i(A) r_0$. Also, the polynomials Φ_i and Θ_i denote ϕ^{2_i} and $\phi_i \phi_{i-1}$ respectively. Now squaring the direction matrix polynomial is the same as in Lanczos Square and gives the equations (4.1) and (4.2) above. Squaring the residual equation gives

$$\Psi_{i+1} \equiv \psi_{i+1}^2 = \psi_i^2 + \hat{a}_i^2 A^2 \Phi_i - 2\hat{a}_i A \overline{\Theta}_i.$$

where $\overline{\Theta}_i = \psi_i \phi_i$. To compute $\overline{\Theta}_i$ we need the following three recurrence equations:

$$\overline{\Theta}_{i+1} = \psi_i \phi_{i+1} - \hat{a}_i A \phi_{i+1} \phi_i$$

$$\Delta_{i+1} \equiv \psi_{i+1} \phi_i = \overline{\Theta}_i - \hat{a}_i A \Phi_i$$

$$\overline{\Delta}_{i+1} \equiv \psi_i \phi_{i+1} = A \overline{\Theta}_i - b_i \overline{\Theta}_i - c_i \Delta_i$$

The result from multiplying the polynomial form of p_{i+1}, r_{i+1} by equation 2. and p_i by equation 6. of algorithm 3.3. Now we formulate the polynomial form of the Orthodir(2) square algorithm.

Algorithm 3.3 Orthodir(2) Square.

Select, x_0, r_0 and $\bar{r}_0$

Take, $b_0 = 0$, $\Theta_0 = \Phi_0 = 1$, $\overline{\Theta}_0 = 1$.

For $i = 0$ **Until** Convergence **Do**

$$\hat{a}_i = \frac{[A^T, \overline{\Theta}_i]}{[1, A^2\Phi_i]}$$

$$\Psi_{i+1} = \Psi_i + \hat{a}_i^2 A^2\Phi_i - 2\hat{a}_i A\overline{\Theta}_i$$

$$b_i = \frac{[A^T, A^2\Phi_i]}{[1, A^2\Phi_i]}$$

$$c_i = \frac{[A^T, A^2\Phi_i]}{[1, A^2\Phi_{i-1}]}$$

$$\Phi_{i+1} = (A - b_i)^2\Phi_i - 2c_i(A - b_i)\Theta_i + c_i^2\Phi_{i-1}$$

$$\Theta_{i+1} = (A\Phi_i - b_i\Phi_i) - c_i\Theta_i$$

$$\Delta_{i+1} = \overline{\Theta}_i - \hat{a}_i A\Phi_i$$

$$\overline{\Delta}_{i+1} = A\overline{\Theta}_i - b_i\overline{\Theta}_i - c_i\Delta_i$$

EndFor

The algorithm needs four matrix vector products per iteration these are: $A\Phi_i, A^2\Phi_i, A\Theta_i, A\overline{\Theta}_i$. Also, it needs one multiplication by the matrix transpose in the first step. This can be removed by adding an additional matrix vector product per iteration. One could reduce the matrix vector products to just two per iteration by introducing additional vector updates. The approximation to the solution is given by:

$$x_{i+1} = x_i - \hat{a}_i[\hat{a}_i A\Phi_i - 2\overline{\Theta}_i]r_0$$

In summary Orthodir(2) squared has three attractive properties: (i) It has the fast convergence. That is the residual polynomial is the square of the residual polynomial of Bi-Orthodir(2). (ii) It has modest storage requirements. (iii) It has the same break down conditions as the Bi-Lanczos method. It is easy to construct the vector form of this algorithm. We will not do it here since we have included any implementation results.

5. Preconditioned Methods

For the preconditioning matrix Pr, we look for matrices such that

$$P_r A \simeq \mathrm{I}$$

and the linear system Pr x = y must be easy to solve. One natural choice is the Incomplete LU factorization, which is the same as LU factorization, except that the resulting L and U matrices keep the original sparsity pattern to facilitate the solution of LUx = y. Let Nz(A) denote the set of pairs of [i,j] for which the entries a_{ij} of the matrix A are nonzero, the *nonzero pattern* of A.

Algorithm 5.1: The Incomplete LU Factorization.

For i=1 step 1 **Until** N **Do**

For j = i+1 step 1 **Until** N **Do**

If ((i,j) belongs to NZ(A)) **then**

$$s_{ij} = A_{ij} - \sum_{t=1}^{min(i,j)-1} L_{it} U_{tj}$$

If ($i \geq j$) **then** $L_{ij} = s_{ij}$

If (i < j) **then** $U_{ij} = \dfrac{s_{ij}}{L_{ii}}$

EndIf

Endfor

Endfor

We next present the Orthomin(k), CGS and CRS algorithms with right preconditioning. That is we are solving the transformed linear system $[AQ^{-1}]Qx = f$. We are using the notation P_r=Q^{-1}. The Orthomin(k) method with right preconditioning minimizes the norm of the residual error over a Krylov space at each iteration. Thus it is easier to monitor the convergence.

Algorithm 5.2: Orthomin(k)

$$p_0 = r_0 = f - Ax_0$$

For i = 0 step 1 **Until** Convergence **Do**

$$a_i = \frac{(r_i\ ,A\ p_i)}{(A\ p_i\ ,A\ p_i)}$$

$$x_{i+1} = x_i + a_i\ p_i$$

$$r_{i+1} = r_i - a_i\ A\ p_i$$

$$p_{i+1} = P_r\ r_{i+1} + \sum_{j=j_i}^{i} b_j^{\,i}\ p_j\ ,$$

$$b_j^{\,i} = -\frac{(A\ P_r r_{i+1}\ ,A\ p_j)}{(A\ p_j\ ,A\ p_j)}\ ,\ j \le i.$$

$$Ap_{i+1} = A\ P_r\ r_{i+1} + \sum_{j=j_i}^{i} b_j^{\,i}\ Ap_j\ ,\ \text{with } j_i = \min(0,\ \text{i-k+1}).$$

Endfor

Vector Operations (addition and multiplication) per iteration are (6k+10)N + 1 Mv + 1 Wprec, where Mv stands for Matrix-vector product and Wprec is the preconditioning work. We present next the CGS method with right preconditioning as it appears in [19].

Algorithm 5.3: Conjugate Gradient Squared (CGS)

$$r_0 = f - Ax_0,\quad q_0 = p_{-1} = 0,\quad \rho_{-1} = 1$$

For i = 0 step 1 **Until** Convergence **Do**

$$\rho_i = (r_0,\ r_i)\ ,\ \beta_i = \frac{\rho_i}{\rho_{i-1}}$$

$$u_i = r_i + \beta_i q_i$$

$$p_i = u_i + \beta_i(q_i + \beta_i p_{i-1})$$

$$v_i = A\ P_r p_i.$$

$$\sigma_i = (r_0, v_i)$$

$$\alpha_i = \frac{\rho_i}{\sigma_i}$$

$$q_{i+1} = u_i - \alpha_i v_i$$

$$x_{i+1} = x_i - \alpha_i P_r(u_i + q_{i+1})$$

$$r_{i+1} = r_i - \alpha_i A P_r(u_i + q_{i+1})$$

Endfor

The vector operations per iteration are 19N + 2 Mv + 2 Wprec, where Mv stands for matrix-vector product, Wprec preconditioning work.

>From remark 3.2 it follows that we can obtain the CRS method from the CGS method by simply computing differently the parameters: σ_i, ρ_i. These parameters for CRS are: $\sigma_i = (A^T r_0, v_i)$ and $\rho_i = (A^T r_0, r_i)$. Thus if the matrix transpose is available CRS simply has one additional matrix vector operation in the first iteration. Otherwise we have to compute these parameters as: $\sigma_i = (r_0, Av_i)$ and $\rho_i = (r_0, Ar_i)$. Then this introduces two additional matrix vector products per iteration. This form of CRS appear in [16]. We can reduce the matrix vector products to two per iteration by using additional vector updates. This is done in the preconditioned form of CRS that we present next.

Algorithm 5.3: Conjugate Residual Squared (CRS)

$$r_0 = Ax_0 - f\ ,\ q_0 = p_{-1} = 0,\ \rho_{-1} = 1$$

$$AP_r q_0 = P_r q_0 = AP_r p_{-1} = 0$$

For i = 0 step 1 **Until** Convergence **Do**

$$\rho_i = (r_0, AP_r r_i)\ ,\ \beta_i = \frac{\rho_i}{\rho_{i-1}}$$

$$u_i = r_i + \beta_i q_i \ , \ P_r u_i = P_r r_i + \beta_i \ P_r q_i \ ,$$

$$p_i = u_i + \beta_i (q_i + \beta_i p_{i-1})$$

$$v_i = AP_r \ p_i = AP_r u_i + \beta AP_r q_i + \beta^2 AP_r p_{i-1}$$

$$\sigma_i = (r_0, AP_r v_i)$$

$$\alpha_i = \frac{\rho_i}{\sigma_i}$$

$$q_{i+1} = u_i - \alpha_i v_i$$

$$P_r q_{i+1} = P_r u_i - \alpha_i P_r v_i$$

$$AP_r q_{i+1} = AP_r u_i - \alpha \ AP_r v_i$$

$$r_{i+1} = r_i - 2\alpha AP_r u_i + \alpha^2 AP_r v_i$$

$$x_{i+1} = x_i - \alpha_i \ Pr \ (u_i + q_{i+1})$$

Endfor

In the next section we present numerical test comparing these three methods.

6. Numerical Tests

We have discretized four boundary value problems in partial differential equations on a square or rectangular region by the method of finite differences. The first problem is a standard elliptic test problem which can be found in [8] and the right hand side function is constructed so that the analytic solution is known. The other three problems have been used by Sonneveld in [19] for testing CGS.

Problem (I)

$$-(b(x,y)u_x)_x - (c(x,y)u_y)_y + (d(x,y)\ u)_x + (e(x,y)\ u) + f(x,y)\ u = g(x,y) \ ,$$

$$\Omega = (0,1)\ x\ (0,1)$$

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

$e(x,y) = \gamma\ (x+y)$, $f(x,y) = \dfrac{1}{(1+x\ y)}$,

$u(x,y) = xe^{xy} sin(\pi y) sin(\pi y)$,

with Dirichlet boundary condition and g(x,y) the corresponding right hand side function. By controlling γ and β, we could change the degree of nonsymmetry. In this report, we set $\gamma = 50.0$, $\beta = 1.0$. We have used the five point difference operator for the Laplacian, central difference for the first derivative. For initial value, we have chosen x(i) = 0.5*mod(i,50)/10.0

Problem (II) (Convection-Diffusion)

$$-\varepsilon(u_{xx} + u_{yy}) + cos(\alpha)u_x + sin(\alpha)u_y = 0$$

$$u(x,y) = x^2 + y^2 \quad on \quad \partial\Omega$$

$$\Omega = (0,1)\ x\ (0,1)$$

We have used the five point difference operator for the Laplacian, central difference scheme for the first derivative. For small ε values, we might need to use upwind difference scheme for the first derivative to maintain diagonal dominance In our experiment $\alpha = 0.5$ and $\varepsilon = 0.1$. For initial guess, we have chosen $x(i)$=0.5*$mod(i,50)$/10.

Problem (III) (Berkeley)

$$-\varepsilon(u_{xx} + u_{yy}) + v_x u_x + v_y u_y = 0$$

$$v_x = 2y(1 - x^2)\ ,\ v_y = -2x(1 - y^2)\ ,$$

$$\Omega = (-1,1) * (0,1)$$

$$u(x,y) = 0\ ,\ x = -1$$

$$u(x,y) = 0\ ,\ x = 1$$

$$u(x,y) = 0\ ,\ y = 1$$

$$u(x,0) = 1 + tanh(10(2x+1))\ ,\ y = 0\ ,\quad -1 \le x \le 0$$

$$\frac{\partial u}{\partial n} = 0\ ,\ \ y=0\ ,\ 0 \le x \le 1$$

We have used five point difference operator for the Laplacian and central difference for the first derivative. For small ε values, we might need to use upwind difference scheme for the first derivative to maintain diagonal dominance, but in our experiment of $\varepsilon = 0.1$. For initial value, we

have chosen $x(i)$=0.5*$mod(i,50)$/10.

Problem (IV)

$$-u_{xx} + u_x + (1 + y^2)(-u_{yy} + u_y) = f(x,y)$$

$$\Omega = (0,1)\ x\ (0,1)$$

where

$$u(x,y) = e^{(x+y)} + x^2(1-x)^2\ ln(1+y^2)$$

with Dirichlet boundary condition, f(x,y) the corresponding right hand side. We have used five point difference operator for the Laplacian, central difference for the first derivative. For initial value, we have chosen $x(i)$=0.5*$mod(i,50)$/10.

The preconditioning we have chosen is the ILU(5) preconditioning with in vector form similar to [20]. The number of vector floating point operations per iteration for the three method (without/with) preconditioning. (i) Orthomin(4): 43/56 , (ii) CGS 37/62, (iii) CRS 37/62. We present a table for the three methods containing the number of iterations (without/with) preconditioning for each of the four problems. The number of interior nodes for the discretization was chosen $nx = 128$ in the x-dimension and $ny = 128$ in the y-dimension except for problem (III) where $ny = 64$. The stopping criterion was the norm of the residual error less than 10^{-6}. We also, present four plots with the residual error norm versus the vector floating point operations for the three methods with preconditioning.

7. Conclusions

We have introduced the Lanczos Squared, Conjugate Residual Squared and Orthodir(2) Squared methods. We studied theoretically the conditions under which these methods do not fail. We showed that CRS is a special case of the CGS method. It turns out the Orthodir(2) Squared has the most attractive properties of all these methods. It has the fast convergence of CGS, it has limited storage requirements, and it has the fewest break down conditions. Since we have not yet implemented the Orthodir(2) Squared we can not comment on its stability properties. The numer-

ical experiment of Orthomin(k), CGS, CRS show that the last two methods are more efficient than Orthomin(k). Also, CRS seems to perform a little better than CGS. As future work we intend to implement the Orthodir(2) Squared and compare to the other methods in particular for problems when these methods seem to break down.

REFERENCES

[1] S. F. Ashby, T. A. Manteuffel, P. E. Saylor, "A Taxonomy for conjugate gradient methods" *Tech. Rep. UIUCDCS-R-88-1414, Dept. of Comp. Sci., Univ. of Illinois*, Urbana, ILL, 1988.

[2] O. Axelsson, "Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations" *Lin. Algebra and its Applications,29, 1-16, 1980.*

[3] C. Brezinski, "Pade-type Approxiamtion and General Orthogonal Polynomials", *Birkhauser-Verlag, Boston, 1980.*

[4] P. Concus and G. H. Golub "A generalized conjugate gradient method for nonsymmetric systems of linear equations", *In: Lect. Notes in Econ. and Math. systems, 134, Springer-Verlag, Berlin,* 1976, pp. 56-65.

[5] A. T. Chronopoulos "s-Step Iterative Methods for (Non)Symmetric (In)definite Linear Systems," *Submitted to SIAM Journal on Numerical Analysis. Also, Univ. of Minnesota,* Dept. of Csci TR 89-35, May 1989.

[6] A. T. Chronopoulos and C. W. Gear "s-Step Iterative Methods for Symmetric Linear Systems," *Journal of Comp. and Applied Math., 25, (1989) 153-168.*

[7] S.C. Eisenstat, H. C. Elman and M. H. Schultz "Variational iterative methods for nonsymmetric systems of linear equations", *SIAM J. Numer. Anal. 20, (1983),*

pp. 345-357.

[8] H.Elman, " Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations", *Ph.D Thesis, Department of Computer Science,Yale,Univ.,1982.*

[9] R. Fletcher, "Conjugate Gradient Methods For Indefinite Systems", *Lecture Notes in Mathematics 506, Springer-Verlag, 1976.*

[10] M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. NBS 49 (1952) pp. 409-436.*

[11] V. Faber and T. A. Manteuffel, "Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method" SIAM J. Numer. Anal., Vol. 21, No. 2, April 1984.

[12] V. Faber and T. A. Manteuffel, "Orthogonal Error Methods" SIAM J. Numer. Anal., Vol. 24, No. 1, February 1987.

[13] C. Lanczos "An Iteration for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators", *J. Res. Nat. Bur. Standards, Vol. 45, 1950, pp. 255-282.*

[14] D. G. Luenberger, "Optimization by vector space methods" *John Wiley, New York, 1969.*

[15] C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations" *SIAM J. Numer. Anal. 12, (1975), pp. 617-624.*

[16] S. J. Polak, et.al. "Semiconductor Device Modelling from the Numerical Point of View" *International J. for Numer. Methods in Engineering, Vol. 24, pp. 763-838* (1987).

[17] Y. Saad, "The Lanczos Biorthogonalization Algorithm and Other Oblique Projection Methods for Solving Large Unsymmetric Systems", *SIAM J. Num. Anal., Vol. 19, No. 3, June 1982.*

[18] Y. Saad and M. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comp., Vol. 7, No. 3, July 1986.*

[19] P.Sonneveld, "CGS, a Fast Lanczos-Type Solver For Nonsymmetric Systems", *SIAM Sci.Stat Comp, Vol.10(1989), pp. 36-52.*

[20] H. Van Der Vorst, "A Vectorizable Variant of some ICGG Methods", *Journ. of Scient. and Stat. Computing, No. 3, pp. 350-356, Sept. 1982.*

[21] O. Widlund, "A Lanczos method for a class of nonsymmetric systems of linear equations," *SIAM J. Num. Anal., 15 (1978), pp. 801-812.*

[22] D. M. Young and K. C. Jea, " Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods" *Linear Algebra Appl., 34 (1980), pp. 159-194.*

Problem	(I)	(II)	(III)	(IV)
Orthomin(4)	392/111	707/167	323/99	378/112
CGS	275/56	212/73	205/72	225/78
CRS	275/56	212/72	207/64	216/65

Table 1. Number of Iterations for the (without/with) ILU(5), for tol.=10^{-6}.

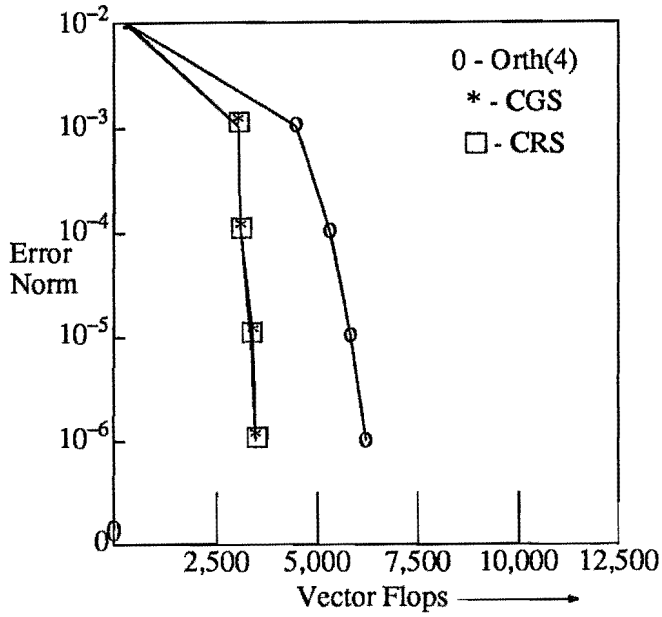

Fig. 1 Problem (I)

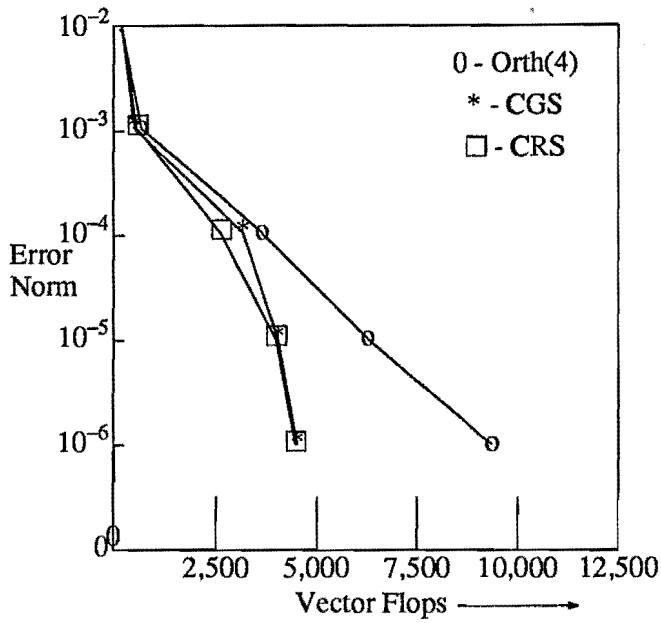

Fig. 2 Problem (II)

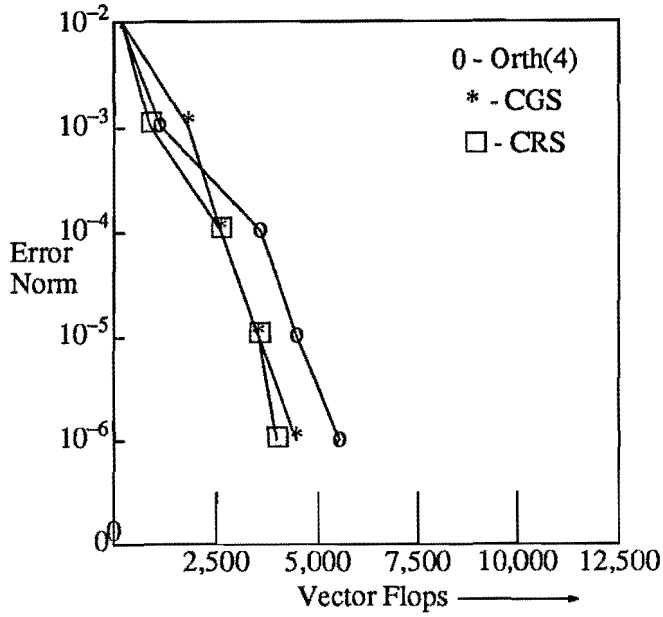

Fig. 3 Problem (III)

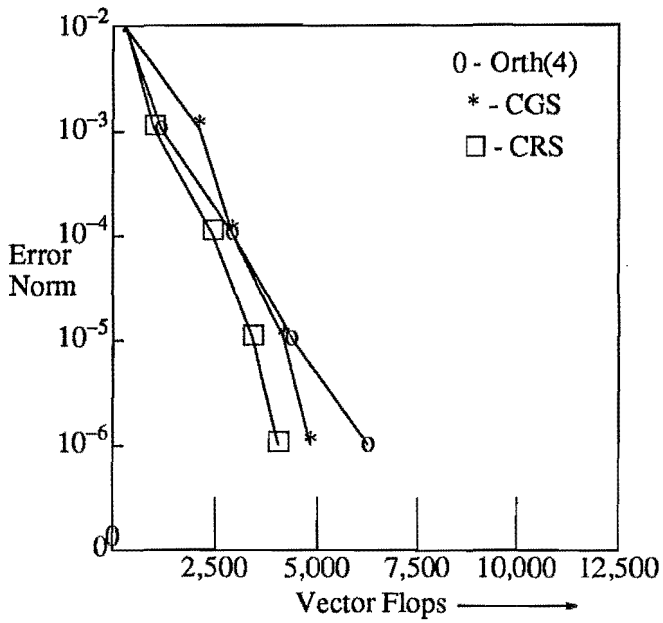

Fig. 4 Problem (IV)