LAPACK WORKING NOTE 56
REDUCING COMMUNICATION COSTS IN THE CONJUGATE GRADIENT ALGORITHM ON DISTRIBUTED MEMORY MULTIPROCESSORS*

EDUARDO D'AZEVEDO†, VICTOR EIJKHOUT‡ AND CHARLES ROMINE †

Abstract. The standard formulation of the conjugate gradient algorithm involves two inner product computations. The results of these two inner products are needed to update the search direction and the computed solution. Since these inner products are mutually interdependent, in a distributed memory parallel environment their computation and subsequent distribution requires two *separate* communication and synchronization phases. In this paper, we present three related mathematically equivalent rearrangements of the standard algorithm that reduce the number of communication phases. We present empirical evidence that two of these rearrangements are numerically stable. This claim is further substantiated by a proof that one of the empirically stable rearrangements arises naturally in the symmetric Lanczos method for linear systems, which is equivalent to the conjugate gradient method.

Key words. preconditioned conjugate gradient, Krylov subspace methods, iterative methods, parallel computers.

AMS(MOS) subject classifications. 65F10

1. Introduction. The conjugate gradient (CG) method is an effective iterative method for solving large sparse symmetric positive definite systems of linear equations. It is robust and, when coupled with an effective preconditioner [18], is generally able to achieve rapid convergence to an accurate solution.

One drawback of the standard formulation of the conjugate gradient algorithm on distributed memory parallel machines is that it involves the computation of two *separate* inner products of distributed vectors. Moreover, the first inner product must be completed before the data are available for computing the second inner product, leading to two distinct synchronization points. Hence, a distributed memory implementation of the standard conjugate gradient method has two separate communication phases for these two inner products. Since communication is quite expensive on the current generation of distributed memory multiprocessors, it is desirable to reduce the communication overhead by combining these two communication phases into one.

Saad [15, 16] has shown one rearrangement of the computation that eliminates a communication phase by computing $\|r_k\|$ based on the relationship

$$\|r_{k+1}\|^2 = \alpha_k^2 \|Ap_k\|^2 - \|r_k\|^2 \tag{1.1}$$

to be numerically unstable.[1] Meurant [10] proposed using (1.1) as a predictor for $\|r_{k+1}\|$, reevaluating the actual norm on the next iteration with an extra inner product. Van Rosendale [19] has proposed (without numerical results) an m-step conjugate gradient algorithm to increase parallelism.

* This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc. and in part by DARPA under contract number DAAL03-91-C-0047

† Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831–6367.

‡ Computer Science Department, University of Tennessee, Knoxville, TN 37996–1301.

[1] Here and throughout the paper, we use $\|\cdot\|$ to denote the l_2 norm.

The conjugate gradient algorithm is known to be closely related to the Lanczos algorithm for tridiagonalizing a matrix [4]. Paige [11, 12, 13] has done detailed analysis to show some variants of the Lanczos algorithm are unstable. Strakos [17] and Greenbaum [5, 6] have considered the close connection between the Lanczos and CG algorithm in the analysis of stability of CG computations under perturbations in finite arithmetic.

In §2, we present three rearrangements of the conjugate gradient computation that eliminate one of the communication phases by computing both inner products at once. (In two of these, an extra inner product is required). We show a natural association between one of these rearrangements and the Lanczos algorithm in §3. A discussion of how the rearrangements of the computation affect the stability properties of the conjugate gradient algorithm and some `MATLAB` numerical experiments on the effectiveness of the rearrangements are included in §4.

2. The conjugate gradient algorithm. The conjugate gradient algorithm involves the computation of a parameter σ_k, which is the A-inner product $\langle p_k, Ap_k \rangle$ of the search direction p_k. In this section we will present several variants of the conjugate gradient algorithm, based on elimination of this A-inner product.

2.1. The standard formulation. We begin by reviewing the standard conjugate gradient procedure [2, 8] for solving the linear system

$$Ax = b \ . \tag{2.1}$$

For simplicity, we assume a zero initial guess, and residual vector $r_1 = b$, with $\langle x, y \rangle = x^t y$ as the usual inner product.

For $k = 1, 2, \ldots$

$$\begin{aligned}
\gamma_k &= \boxed{\langle r_k, r_k \rangle} \\
\beta_k &= \gamma_k / \gamma_{k-1} \quad (\beta_1 = 0) \\
p_k &= r_k + \beta_k p_{k-1} \quad (p_1 = r_1) \qquad (2.2) \\
v_k &= Ap_k \\
\sigma_k &= \boxed{\langle p_k, v_k \rangle} \\
\alpha_k &= \gamma_k / \sigma_k \\
x_{k+1} &= x_k + \alpha_k p_k \\
r_{k+1} &= r_k - \alpha_k v_k \ . \qquad (2.3)
\end{aligned}$$

Each of the inner products highlighted requires a communication step on a distributed memory machine.

Saad [15, 16] and Meurant [10] have considered eliminating the first inner product for $\gamma_k = \langle r_k, r_k \rangle$. We propose eliminating the second communication phase by finding alternative expressions for σ_k.

We will rely on one intrinsic property of the CG procedure: the orthogonality of residual vectors (and equivalently the conjugacy of search directions [8, page 420])

$$\frac{\langle r_k, r_{k+1} \rangle}{\langle r_k, r_k \rangle} \equiv \frac{\langle p_k, Ap_{k+1} \rangle}{\langle p_k, Ap_k \rangle} = 0. \tag{2.4}$$

2.2. Variant 1. We derive the first rearrangement by substituting $p_k = r_k + \beta_k p_{k-1}$ for both occurrences of p_k in $\langle p_k, Ap_k \rangle$:

$$\begin{aligned}
\sigma_k &= \langle p_k, v_k \rangle = \langle p_k, Ap_k \rangle \\
&= \langle r_k + \beta_k p_{k-1}, Ar_k + \beta_k v_{k-1} \rangle \\
&= \langle r_k, Ar_k \rangle + \beta_k \langle r_k, v_{k-1} \rangle + \\
&\quad \beta_k \langle p_{k-1}, Ar_k \rangle + \beta_k^2 \langle p_{k-1}, v_{k-1} \rangle \\
\sigma_k &= \langle r_k, Ar_k \rangle + 2\beta_k \boxed{\langle r_k, v_{k-1} \rangle} + \beta_k^2 \sigma_{k-1} \ . \qquad (2.5)
\end{aligned}$$

From (2.3) and (2.4):

$$\begin{aligned}
r_k &= r_{k-1} - \alpha_{k-1} v_{k-1} \\
\langle r_k, r_k \rangle &= \langle r_k, r_{k-1} \rangle - \alpha_{k-1} \langle r_k, v_{k-1} \rangle \\
\gamma_k &= 0 - \alpha_{k-1} \boxed{\langle r_k, v_{k-1} \rangle} \ . \qquad (2.6)
\end{aligned}$$

Therefore by (2.5), (2.6) and $\beta_k = \gamma_k / \gamma_{k-1}$,

$$\begin{aligned}
\sigma_k &= \langle r_k, Ar_k \rangle + 2\beta_k(-\gamma_k/\alpha_{k-1}) + \beta_k^2 \sigma_{k-1} \\
&= \eta_k - \beta_k^2 \sigma_{k-1}, \quad \text{where} \quad \eta_k = \langle r_k, Ar_k \rangle \\
&= \eta_k + \beta_k \epsilon_k^{(1)}, \quad \text{where} \quad \epsilon_k^{(1)} = -\beta_k \sigma_{k-1} \ . \qquad (2.7)
\end{aligned}$$

2.3. Variants 2 and 3. If we expand the occurrences of p_k in $\langle p_k, Ap_k \rangle$ one at a time, we find different rearrangements of the algorithm.

$$\begin{aligned}
\sigma_k &= \langle p_k, Ap_k \rangle \\
&= \langle r_k + \beta_k p_{k-1}, Ap_k \rangle \\
&= \langle r_k, Ap_k \rangle + \beta_k \boxed{\langle p_{k-1}, Ap_k \rangle} \\
&= \langle r_k, A(r_k + \beta_k p_{k-1}) \rangle \\
&= \langle r_k, Ar_k \rangle + \beta_k \langle r_k, Ap_{k-1} \rangle \qquad (2.8) \\
\sigma_k &= \eta_k + \beta_k \epsilon_k^{(2)} \qquad \text{where } \epsilon_k^{(2)} = \langle r_k, Ap_{k-1} \rangle
\end{aligned}$$

Thus we find a rearrangement of the conjugate gradient method where we compute inner products $\langle r_k, Ar_k \rangle$, $\langle r_k, r_k \rangle$, and $\langle r_k, Ap_{k-1} \rangle$ simultaneously and compute $\sigma_k = \langle p_k, Ap_k \rangle$ by recurrence formula (2.8).

Further expansion of (2.8) with $p_{k-1} = r_{k-1} + \beta_{k-1} p_{k-2}$ gives

$$\begin{aligned}
\sigma_k &= \langle r_k, Ar_k \rangle + \beta_k \langle r_k, Ap_{k-1} \rangle \\
&= \langle r_k, Ar_k \rangle + \beta_k \langle r_k, A(r_{k-1} + \beta_{k-1} p_{k-2}) \rangle \\
&= \langle r_k, Ar_k \rangle + \beta_k \langle r_k, Ar_{k-1} \rangle + \beta_k \beta_{k-1} \boxed{\langle r_k, Ap_{k-2} \rangle} \qquad (2.9)
\end{aligned}$$

Now mathematically $\langle r_k, Ap_{k-2} \rangle = 0$ since

$$\begin{aligned}
\langle r_k, Ap_{k-2} \rangle &= \langle p_k - \beta_k p_{k-1}, Ap_{k-2} \rangle \\
&= \langle p_k, Ap_{k-2} \rangle + \beta_k \langle p_{k-1}, Ap_{k-2} \rangle = 0
\end{aligned}$$

Hence, assuming A-conjugacy of $\langle p_k, Ap_{k-2} \rangle$ and $\langle p_{k-1}, Ap_{k-2} \rangle$, we have

$$\begin{aligned}
\sigma_k &= \langle r_k, Ar_k \rangle + \beta_k \langle r_k, Ar_{k-1} \rangle \\
&= \eta_k + \beta_k \epsilon_k^{(3)} \qquad \text{where } \epsilon_k^{(3)} = \langle r_k, Ar_{k-1} \rangle \qquad (2.10)
\end{aligned}$$

This gives us a rearrangement of the conjugate gradient method where we compute inner products $\langle r_k, Ar_k\rangle$, $\langle r_k, r_k\rangle$, and $\langle r_k, Ar_{k-1}\rangle$ simultaneously, and compute $\sigma_k = \langle p_k, Ap_k\rangle$ by recurrence formula (2.10).

2.4. The modified conjugate gradient method. Unifying the three variants above, we propose the following rearrangement of the conjugate gradient procedure. First initialize σ_1 and v_1, by performing one step of the standard algorithm

$$
\begin{aligned}
r_1 &= b, \quad \gamma_1 = \langle r_1, r_1\rangle, \quad p_1 = r_1, \quad v_1 = Ap_1 \\
\sigma_1 &= \langle p_1, v_1\rangle, \quad x_2 = (\gamma_1/\sigma_1)p_1 \\
\text{For } k &= 2, 3, \ldots \\
s_k &= Ar_k \\
\text{Compute } \gamma_k &= \boxed{\langle r_k, r_k\rangle} \text{ and } \eta_k = \boxed{\langle r_k, s_k\rangle} \\
\text{and } \epsilon_k &= \begin{cases} -\beta_k\sigma_{k-1} & \text{for variant 1} \\ \boxed{\langle r_k, Ap_{k-1}\rangle} & \text{for variant 2} \\ \boxed{\langle r_k, Ar_{k-1}\rangle} & \text{for variant 3} \end{cases} \\
\beta_k &= \gamma_k/\gamma_{k-1} \\
p_k &= r_k + \beta_k p_{k-1} \\
v_k &= s_k + \beta_k v_{k-1} \quad (v_k \equiv Ap_k) \\
\sigma_k &= \eta_k + \beta_k\epsilon_k \\
\alpha_k &= \gamma_k/\sigma_k \\
x_{k+1} &= x_k + \alpha_k p_k \\
r_{k+1} &= r_k - \alpha_k v_k \ .
\end{aligned} \tag{2.11}
$$

Note that the above procedure requires extra storage for the vector s_k and extra work in updating the vector v_k for all three variants. Variants 2 and 3 require an additional inner product and storage for v_{k-1} and s_{k-1} respectively.

2.5. Stability considerations of variants 1, 2 and 3. In §3, we relate variant 1 of the conjugate gradient algorithm to a version of the Lanczos process that has been shown to be stable by Paige [11]. Here we summarize some important relationships among the three variants described above that provide some insight into their behavior. Returning to the expression for ϵ_k given in (2.11), we note the following:

$$\epsilon_k^{(2)} = \langle r_k, Ap_{k-1}\rangle = \epsilon_k^{(1)} + \frac{\langle r_k, r_{k-1}\rangle}{\alpha_{k-1}}. \tag{2.12}$$

Since both versions 1 and 2 rely on orthogonality between the residual vectors r_k and r_{k-1} ("one-step" orthogonality), these are numerically equivalent formulations. However, we further note that

$$\epsilon_k^{(3)} = \langle r_k, Ar_{k-1}\rangle = \epsilon_k^{(1)} + \frac{\langle r_{k+1}, r_{k-1}\rangle}{\alpha_{k+1}}. \tag{2.13}$$

This expression for the term ϵ_k in variant 3 of the conjugate gradient algorithm relies on orthogonality between the residual vectors r_{k+1} and r_{k-1}. Theoretically, these vectors should be orthogonal; however, this "two-step" orthogonality is not assured numerically. Hence, one might expect that variant 3 would exhibit greater instability than either of the other two variants. The results in §4 show that this is indeed the case, and variant 3 should be avoided computationally.

3. The Lanczos algorithm. In this section we present a natural connection between variant 1 of the modified conjugate gradient algorithm and the Lanczos process. The Lanczos process is a procedure to compute an orthogonal matrix $\tilde{Q}$ and tridiagonal matrix $\tilde{T}$ such that

$$\tilde{Q}^t A \tilde{Q} = \tilde{T} = \begin{bmatrix} \tilde{\alpha}_1 & \tilde{\beta}_1 & & \\ \tilde{\beta}_1 & \tilde{\alpha}_2 & \ddots & \\ & \ddots & \ddots & \tilde{\beta}_{n-1} \\ & & \tilde{\beta}_{n-1} & \tilde{\alpha}_n \end{bmatrix}, \quad \tilde{Q}^t \tilde{Q} = I \ . \tag{3.1}$$

We arrive at the Lanczos algorithm by equating columns in $A\tilde{Q} = \tilde{Q}\tilde{T}$, where $\tilde{q}_j$'s are columns of $\tilde{Q}$, $\tilde{Q} = [\tilde{q}_1 | \ldots | \tilde{q}_n]$,

$$\begin{aligned} A\tilde{q}_j &= \tilde{\beta}_{j-1}\tilde{q}_{j-1} + \tilde{\alpha}_j \tilde{q}_j + \tilde{\beta}_j \tilde{q}_{j+1} \\ \tilde{\beta}_j \tilde{q}_{j+1} &= A\tilde{q}_j - \tilde{\alpha}_j \tilde{q}_j - \tilde{\beta}_{j-1}\tilde{q}_{j-1}, \\ \langle \tilde{q}_j, A\tilde{q}_j \rangle &= \tilde{\alpha}_j, \qquad \langle \tilde{q}_{j+1}, A\tilde{q}_j \rangle = \tilde{\beta}_j \ . \end{aligned} \tag{3.2}$$

Paige [11] presents the following *stable* version of the Lanczos algorithm. Choose $\tilde{r}_1$ to be nonzero,

$$\begin{aligned} &\tilde{\beta}_1 = \pm \|\tilde{r}_1\|_2, \qquad \tilde{q}_1 = \tilde{r}_1 / \tilde{\beta}_1, \\ &\text{For } j = 1, 2, \ldots \\ &\qquad \tilde{\alpha}_j = \langle \tilde{q}_j, A\tilde{q}_j \rangle = \langle \tilde{r}_j, A\tilde{r}_j \rangle \,/\, \langle \tilde{r}_j, \tilde{r}_j \rangle \\ &\qquad \tilde{r}_{j+1} = A\tilde{q}_j - \tilde{\alpha}_j \tilde{q}_j - \tilde{\beta}_j \tilde{q}_{j-1}, \\ &\qquad \tilde{\beta}_{j+1} = \pm \|\tilde{r}_{j+1}\|_2 \\ &\qquad \tilde{q}_{j+1} = \tilde{r}_{j+1} / \tilde{\beta}_{j+1} \ . \end{aligned} \tag{3.3}$$

In the same paper, Paige [11] shows that using the equivalent formula $\tilde{\beta}_j = \langle q_{j+1}, Aq_j \rangle$ in place of $\|\tilde{r}_j\|$ to compute $\tilde{\beta}_j$ leads to poor stability.

Golub and Van Loan [4, page 342] present the application of the Lanczos process in solving linear equations, which is mathematically equivalent to the CG algorithm:

$$\begin{aligned} \tilde{Q}_k^t A \tilde{Q}_k y_k &= \tilde{Q}_k^t b, \qquad \tilde{Q}_k y_k = x_k \\ \tilde{T}_k y_k &= \tilde{b}_k, \qquad \tilde{b}_k = \tilde{Q}_k^t b \end{aligned} \tag{3.4}$$

Since A (and hence T) is symmetric positive definite, the solution $x_k = \tilde{Q}_k y_k$ can then be computed from (3.4) by the stable LDL^t factorization of $\tilde{T}_k$

$$\begin{aligned} x_k &= \tilde{Q}_k \tilde{T}_k^{-1} \tilde{Q}_k^t b = \tilde{Q}_k \tilde{T}_k^{-1} \tilde{b}_k \\ &= \tilde{Q}_k \left[\tilde{L}_k \tilde{D}_k \tilde{L}_k^t \right]^{-1} \tilde{b}_k \\ &= \left(\tilde{P}_k \tilde{L}_k^t \right) \left[\tilde{L}_k^{-t} \tilde{D}_k^{-1} \tilde{L}_k^{-1} \right] \tilde{b}_k, \qquad \text{where} \quad \tilde{P}_k \tilde{L}_k^t = \tilde{Q}_k \\ &= \tilde{P}_k a_k, \qquad \text{where} \quad \tilde{L}_k \tilde{D}_k a_k = \tilde{b}_k \ . \end{aligned} \tag{3.5}$$

With further simplifications, one can show $a_k = [\alpha_1, \ldots, \alpha_k]^t$ and this process can be rewritten in the more familiar form $x_k = x_{k-1} + \alpha_k p_k$ of the CG algorithm.

We note that the above process requires computation of $\tilde{\beta}_j = \pm\langle \tilde{r}_j, \tilde{r}_j\rangle$ and $\tilde{\alpha}_j = \langle \tilde{r}_j, A\tilde{r}_j\rangle / \langle \tilde{r}_j, \tilde{r}_j\rangle$; as in the modified CG rearrangement (2.11), *both* inner products can be computed together. Moreover, the use of the alternative formula $\tilde{\beta}_j = \langle q_{j+1}, Aq_j\rangle$ for $\|\tilde{r}_j\|$ leads to instability, as in the case of Saad's rearrangement (1.1).

The update formulas from the CG algorithm (2.11)

$$p_j = r_j + \beta_j p_{j-1}, \qquad r_{j+1} = r_j - \alpha_j A p_j \tag{3.6}$$

can be rewritten as a three-term recurrence relation (3.2) similar to the Lanczos algorithm. We have

$$\begin{aligned} r_{j+1} &= r_j - \alpha_j A\,(r_j + \beta_j p_{j-1}) \\ &= r_j - \alpha_j A r_j - (\alpha_j\beta_j/\alpha_{j-1})\,(r_{j-1} - r_j) \\ &= (1 + \alpha_j\beta_j/\alpha_{j-1})\,r_j - \alpha_j A r_j - (\beta_j\alpha_j/\alpha_{j-1})\,r_{j-1} \\ Ar_j &= (-\beta_j/\alpha_{j-1})\,r_{j-1} + (1 + \alpha_j\beta_j/\alpha_{j-1})\,r_j - (1/\alpha_j)\,r_{j+1} \end{aligned} \tag{3.7}$$

By normalizing the vectors $r_j = q_j\sqrt{\gamma_j}$, the above can be written as

$$AQ = QT, \qquad Q = [q_1|\ldots|q_n]\,, \tag{3.8}$$

with $T = \{t_{ij}\}$ symmetric tridiagonal,

$$t_{jj} = \frac{1}{\alpha_j} + \frac{\beta_j}{\alpha_{j-1}}, \quad t_{jj-1} = \frac{-\sqrt{\beta_j}}{\alpha_{j-1}}, \quad t_{jj+1} = \frac{-\sqrt{\beta_{j+1}}}{\alpha_j}\ . \tag{3.9}$$

Note relationship (3.8) follows directly from the update formulas (3.6). However, while this relationship holds for all variants of CG, numerically the entries t_{ij} will differ.

Although matrix Q is mathematically orthogonal, and (3.8) is a tridiagonalization of A, commonly there is some loss of numerical orthogonality. We can show that variant 1 of the modified CG algorithm produces q_j's (and r_j's) that are numerically equivalent to those obtained by a stable variant of the Lanczos process, i.e. the q_j's are as numerically orthogonal as the Lanczos process produces.

First we require consistent computation of $\langle q_j, Aq_j\rangle$; that is, we show that the choice of $\sigma_j = \eta_j - \beta_j^2\sigma_{j-1}$ (as in variant 1) causes the diagonal entries of T and $\tilde{T}$ to be the same:

$$\begin{aligned} t_{jj} &= \frac{1}{\alpha_j} + \frac{\beta_j}{\alpha_{j-1}} \\ &= \frac{1}{\gamma_j}(\boxed{\sigma_j} + \beta_j^2\sigma_{j-1}) \\ &= \frac{1}{\gamma_j}(\boxed{\eta_j - \beta_j^2\sigma_{j-1}} + \beta_j^2\sigma_{j-1}) \\ &= \frac{\eta_j}{\gamma_j} = \frac{\langle r_j, Ar_j\rangle}{\langle r_j, r_j\rangle} = \tilde{\alpha}_j\ . \end{aligned} \tag{3.10}$$

By expanding the right hand side of

$$|\tilde{\beta}_j| = \|Aq_j - t_{jj}q_j - t_{j-1,j}q_{j-1}\|, \tag{3.11}$$

it can be shown that the off-diagonal entries of T and $\tilde{T}$ are equivalent ($|\tilde{\beta}_j| = |t_{jj+1}|$), using only the assumption of one-step orthogonality, $\langle r_{k+1}, r_k\rangle = \langle r_k, r_{k-1}\rangle = 0$.

4. Numerical experiments on stability. The aim of the following experiments is to determine the stability and convergence properties of the modified conjugate gradient procedures.

We performed a number of `MATLAB` experiments in solving $Ax = b$ by the conjugate gradient procedure to study the convergence behavior on different distributions of eigenvalues of the preconditioned matrix. In variant 2 (resp. 3), $\langle r_k, v_{k-1}\rangle$ (resp. $\langle r_k, s_{k-1}\rangle$) is computed by an extra inner product. Meurant's rearrangement is taken from [10] and the Lanczos rearrangement is adapted from [4, page 342] by evaluating the two inner products for $\tilde{\alpha}_j$ together as $\tilde{\alpha}_j = \langle \tilde{r}_j, A\tilde{r}_j\rangle \,/\, \langle \tilde{r}_j, \tilde{r}_j\rangle$.

Test 1. The matrices considered have the eigenspectrum used by Strakos [17] and Greenbaum and Strakos [6]

$$\lambda_i = \lambda_1 + \frac{i-1}{n-1}(\lambda_n - \lambda_1)\rho^{n-i}, \qquad i = 2, \ldots, n, \quad \rho \in (0,1). \tag{4.1}$$

We have used $n = 100$, $\lambda_1 =$ `1E-3`, $\kappa = \lambda_n/\lambda_1 =$ `1E5` and $\rho = 0.6, 0.8, 0.9, 1.0$ in the experiments. For $\rho = 1$, we have a uniformly distributed spectrum, and $\rho < 1$ describes quantitatively the clustering at λ_1.

Test 2. The eigenspectrum has a gap, $\{1, \ldots, 50, 10051, \ldots, 10100\}$.

Test 3. The eigenspectrum has double eigenvalues, $\{1, 1, 2, 2, \ldots, 50, 50\}$.

Test 4. The eigenspectrum consists of the roots of the Chebyshev polynomial $T_n(x)$ shifted from $[-1, 1]$ to the interval $[a, b]$

$$\lambda_i = \frac{(b-a)}{2}\cos\left(\frac{\pi/2 + (i-1)\pi}{n}\right) + \frac{(b+a)}{2}, \quad i = 1, \ldots, n. \tag{4.2}$$

We have used $n = 100$, $a = 1$, $b =$ `1E5`.

As done in Hageman and Young [7], Greenbaum [5] and Strakos [17], we operate on *diagonal* matrices. This procedure is equivalent to representing all vectors over the basis of eigenvectors of matrix A. In all cases, a random[2] right hand side and zero initial guess are used.

We display the decrease of A-norm of the error at each iteration divided by the A-norm of the initial error

$$\frac{\langle \tilde{x} - x_k, A(\tilde{x} - x_k)\rangle^{1/2}}{\langle \tilde{x} - x_0, A(\tilde{x} - x_0)\rangle^{1/2}}, \qquad \tilde{x} = A^{-1}b. \tag{4.3}$$

Figures 4.1–4.6 display the convergence results from Test 1. Note that for $\rho = 0.8, 0.9$ both the standard algorithm and the stable variants of CG exhibit similar slow convergence behavior. For Test 1 with $\rho = 0.6$, the standard CG algorithm shows the best convergence properties. Variant 3 exhibits the poor stability that was predicted in § 2.5. The results from the other variants show similar stable behavior.

Figures 4.7–4.12 display the convergence results on Tests 2–4. Again with the exception of variant 3, all the results on Tests 2–4 again show similar convergence behavior among the standard CG and the different rearrangements of CG.

[2] uniform over $[-1, 1]$

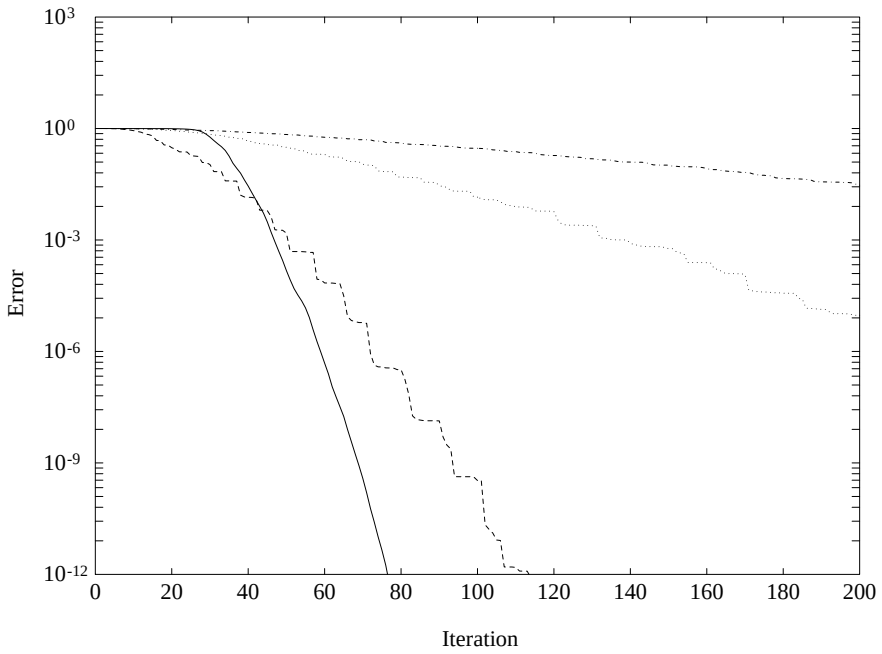

FIG. 4.1. Standard CG on Test 1. *Dashed curve:* $\rho = 0.6$*; dotted curve:* $\rho = 0.8$*; dash-dot curve:* $\rho = 0.9$*; solid curve:* $\rho = 1$.

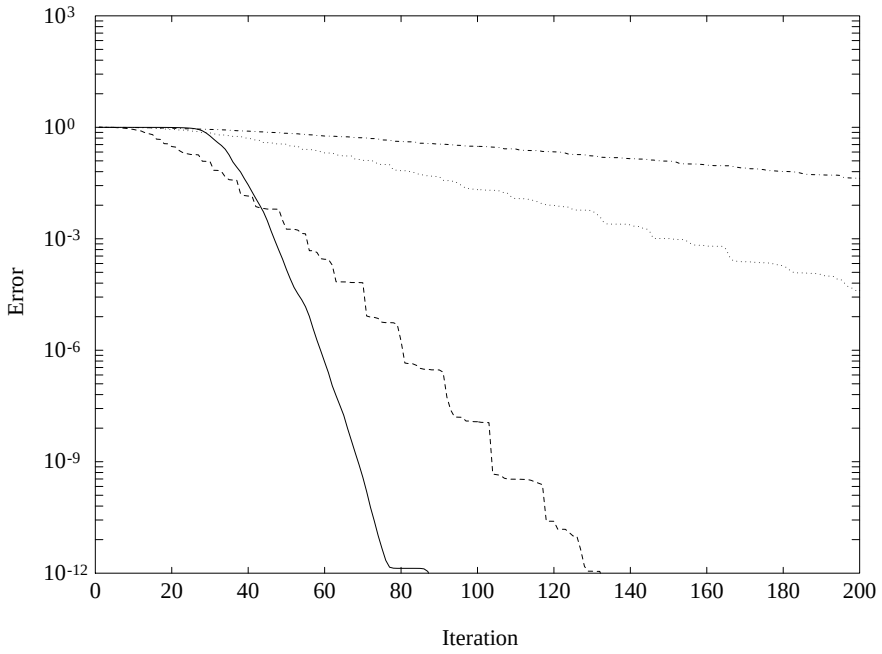

FIG. 4.2. Variant 1 on Test 1. *Dashed curve:* $\rho = 0.6$*; dotted curve:* $\rho = 0.8$*; dash-dot curve:* $\rho = 0.9$*; solid curve:* $\rho = 1$.

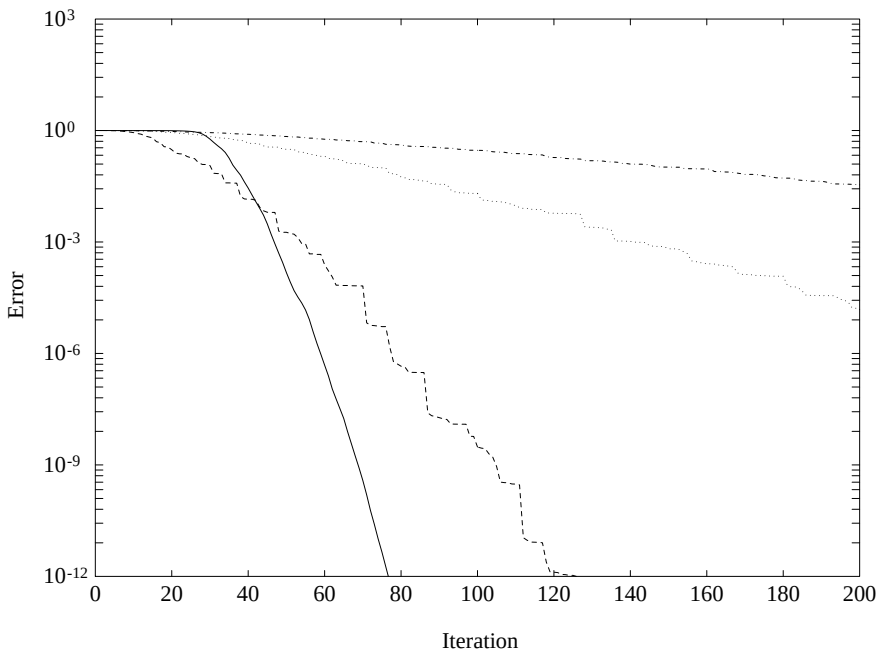

FIG. 4.3. Variant 2 on Test 1. *Dashed curve:* $\rho = 0.6$*; dotted curve:* $\rho = 0.8$*; dash-dot curve:* $\rho = 0.9$*; solid curve:* $\rho = 1$.

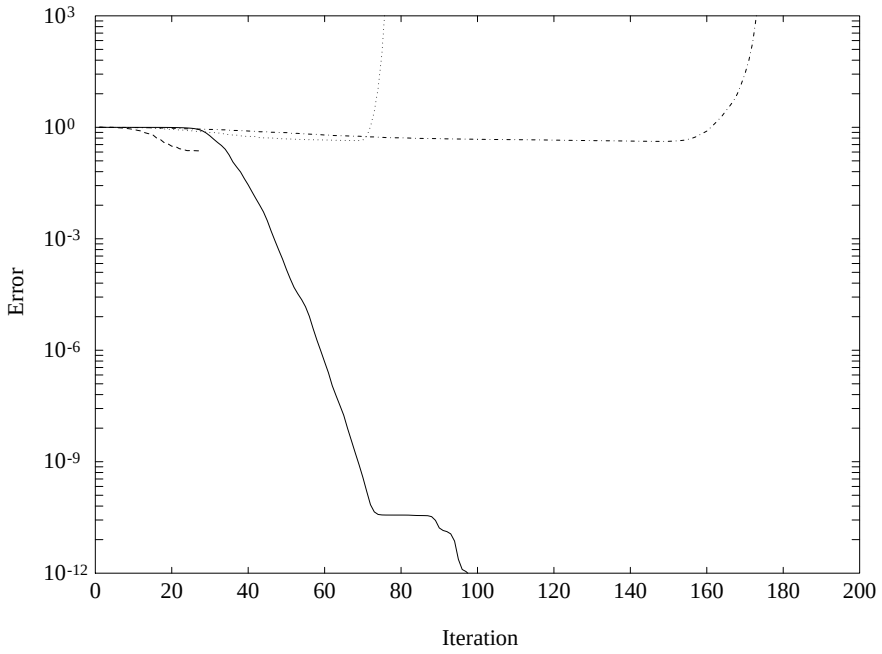

FIG. 4.4. Variant 3 on Test 1. *Dashed curve:* $\rho = 0.6$*; dotted curve:* $\rho = 0.8$*; dash-dot curve:* $\rho = 0.9$*; solid curve:* $\rho = 1$.

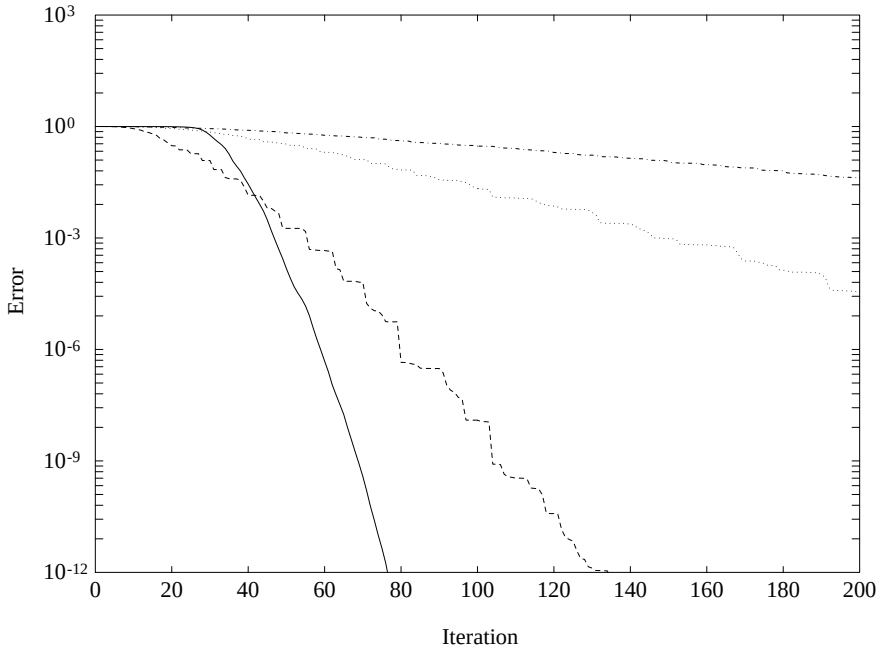

FIG. 4.5. Meurant Rearrangement on Test 1. *Dashed curve:* $\rho = 0.6$*; dotted curve:* $\rho = 0.8$*; dash-dot curve:* $\rho = 0.9$*; solid curve:* $\rho = 1$.

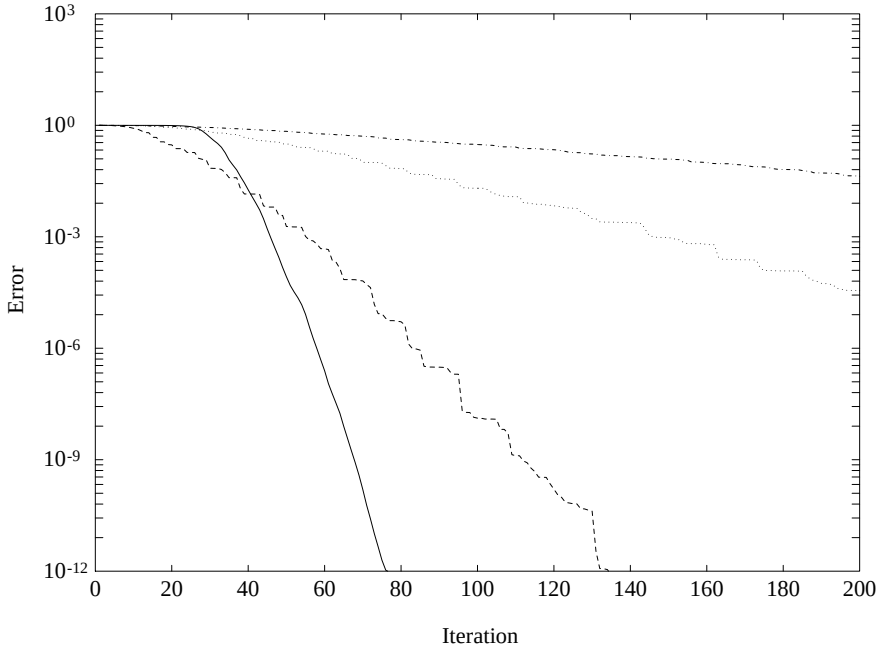

FIG. 4.6. Lanczos Rearrangement on Test 1. *Dashed curve:* $\rho = 0.6$*; dotted curve:* $\rho = 0.8$*; dash-dot curve:* $\rho = 0.9$*; solid curve:* $\rho = 1$.

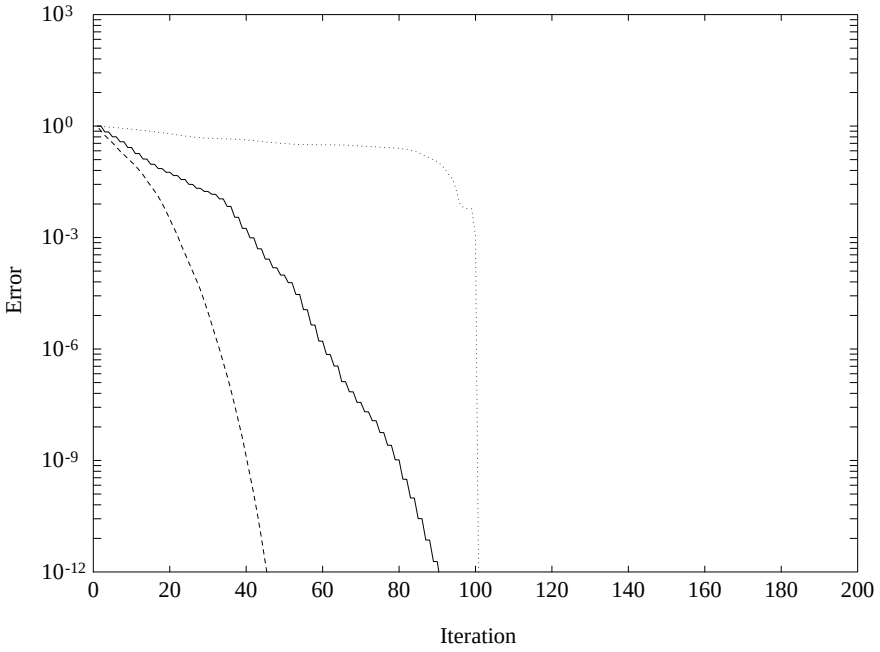

FIG. 4.7. Standard CG on Tests 2–4. *Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.*

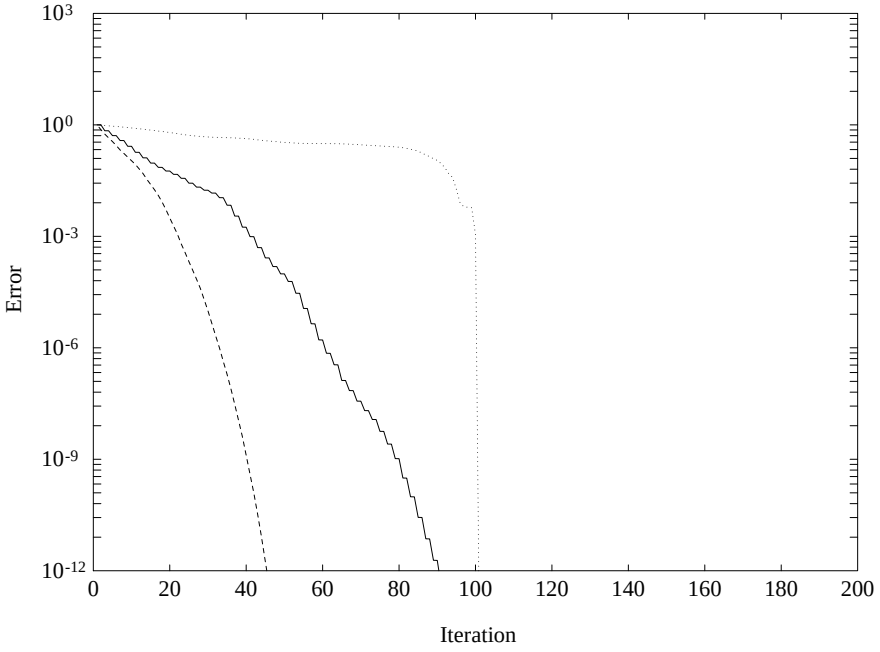

FIG. 4.8. Variant 1 on Tests 2–4. *Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.*

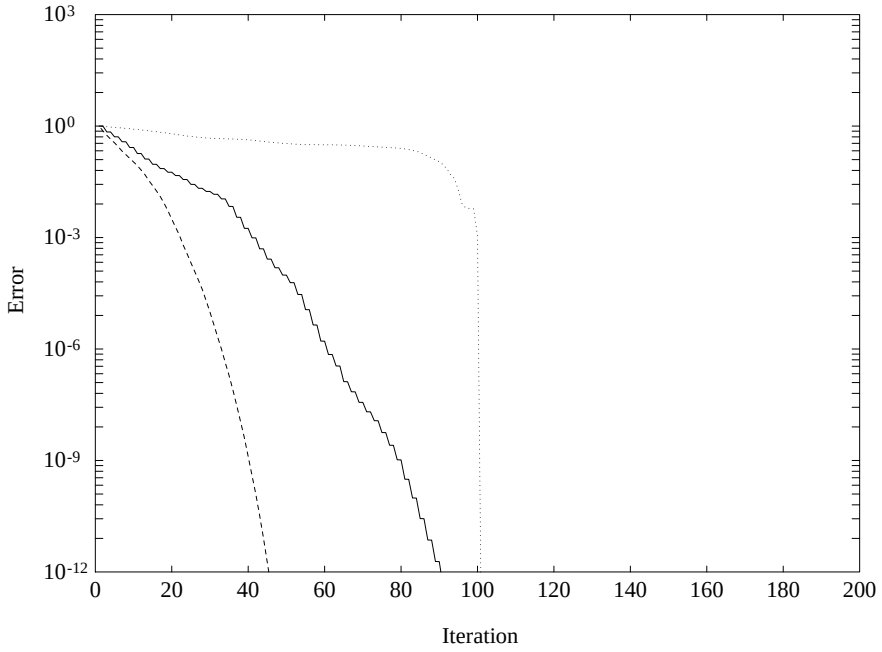

FIG. 4.9. Variant 2 on Tests 2–4. *Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.*

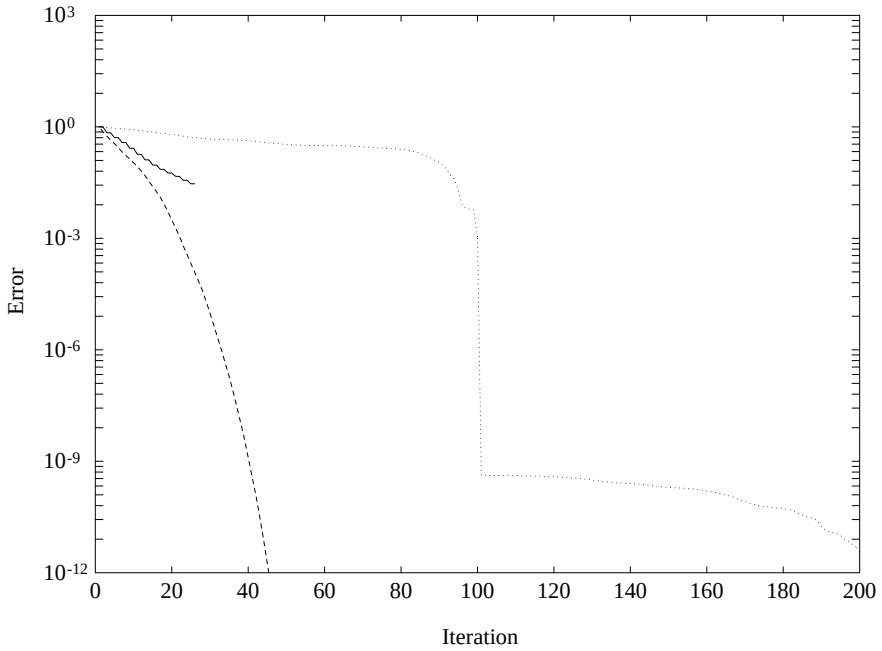

FIG. 4.10. Variant 3 on Tests 2–4. *Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.*

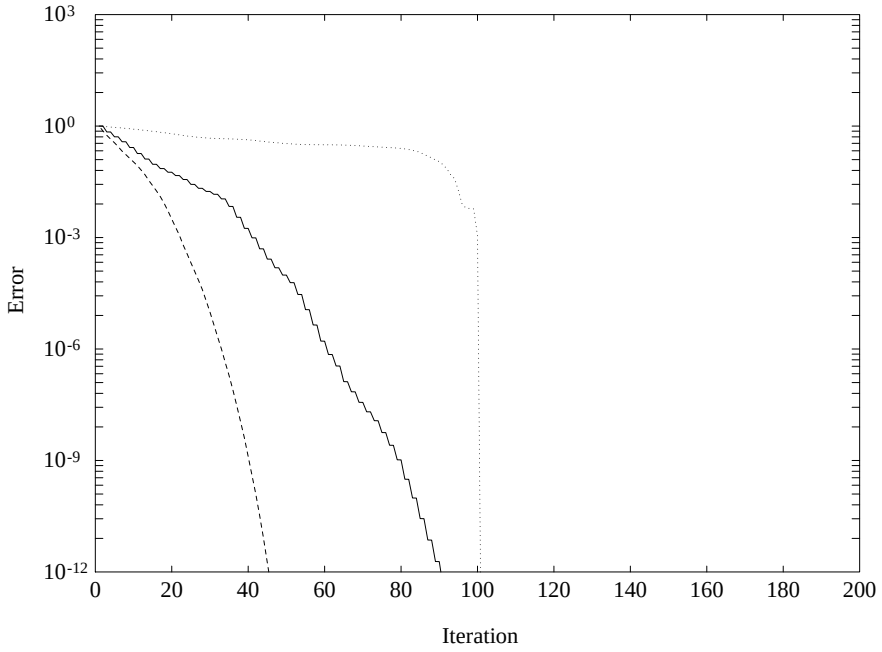

FIG. 4.11. Meurant Rearrangement on Tests 2–4. *Solid curve: Test* 2*; dashed curve: Test* 3*; dotted curve Test* 4*.*

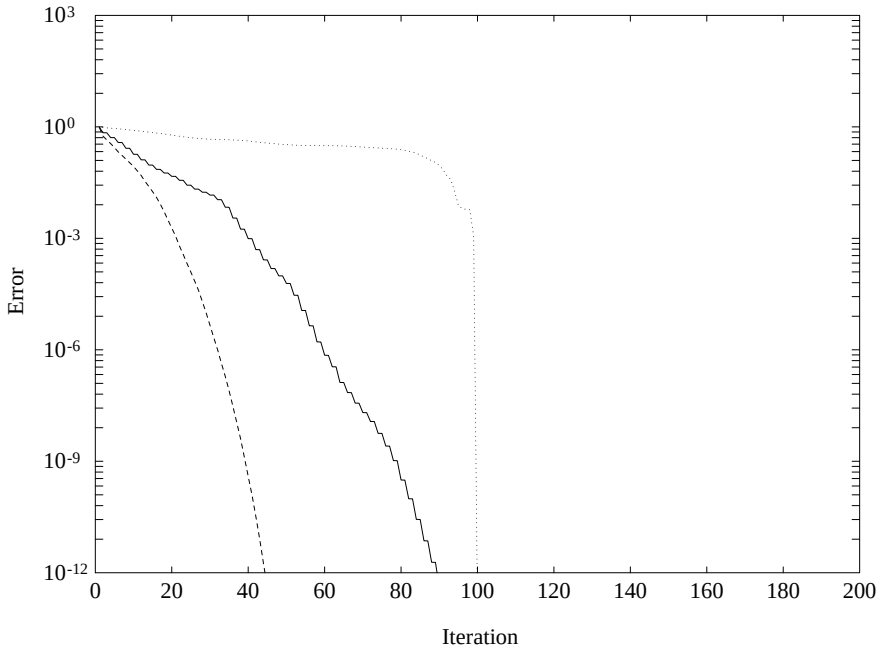

FIG. 4.12. Lanczos Rearrangement on Tests 2–4. *Solid curve: Test* 2*; dashed curve: Test* 3*; dotted curve Test* 4*.*

Table 5.1
Description of test problems.

Problem	Order	Nonzeros	Description
BCSSTK13	2003	11973	Fluid Flow Generalized Eigenvalues
BCSSTK14	1806	32630	Roof of Omni Coliseum, Atlanta
BCSSTK15	3948	60882	Module of an Offshore Platform
BCSSTK18	11948	80519	R.E.Ginna Nuclear Power Station

Table 5.2
Timing results.

	standard CG		modified CG	
Problem	Iterations	Time	Iterations	Time
BCSSTK13	1007	19.56	1007	16.99
BCSSTK14	232	2.72	232	2.35
BCSSTK15	376	7.60	376	6.72
BCSSTK18	697	34.55	697	32.80

5. Parallel performance. To gauge the effectiveness of the modified CG procedure, we performed a number of experiments in comparing the run-time in standard CG and variant 1. We chose variant 1 since although variant 2 has similar convergence properties, it requires an extra inner product. The test matrices were chosen from the Harwell-Boeing Test Collection [3]. The experiments were performed on 16 nodes of the iPSC/860 hypercube. Each matrix was first reordered by the bandwidth reducing Reverse Cuthill-McKee ordering [9]. The matrix was then equally block partitioned by rows and distributed across the processors in ELLPACK format [14]. In all cases, a random right hand side and zero initial guess were used, and convergence was assumed when

$$\|r_k\| \leq 10^{-8}\|r_0\|. \tag{5.1}$$

The conjugate gradient procedure is rarely used without some form of preconditioning to accelerate convergence. In the tests described below, we used a block preconditioner derived as follows: Let A_i be the diagonal block of the matrix A contained in processor i, and write $A_i = L_i + D_i + L_i^t$ where L_i is strictly lower triangular and D_i is diagonal. Then the preconditioning matrix M is $M = \mathrm{diag}(M_1, M_2, \ldots, M_p)$, where $M_i = (L_i + D_i)D_i^{-1}(L_i + D_i)^t$. As shown in Axelsson and Barker [1], this corresponds to each processor doing a single SSOR step (with $\omega = 1$) on its diagonal block A_i. We chose this preconditioner since it requires no additional communication among the processors when implemented in parallel.

Table 5.1 is a brief description of the problems selected from the Harwell-Boeing Test Collection. Table 5.2 shows the number of iterations and time (in seconds) required to solve the corresponding problems on an Intel iPSC/860 with 16 processors. In all cases, variant 1 shows an improvement over the standard algorithm in the time required for solution, ranging from 5% to 13%. Moreover, variant 1 shows no unstable behavior since it takes the same number of iterations as standard CG.

6. Conclusion. We have presented three rearrangements of the standard conjugate gradient procedure that eliminate one synchronization point by performing both required inner products at once. Two of these rearrangements (variants 2 and 3) re-

quire an extra inner product. We showed that variant 1 has a natural connection with the Lanczos process for solving linear equations, which implies that variant 1 is stable. `MATLAB` simulations on matrices with "pathological" spectra confirm that variants 1 and 2 are stable. Variant 3, which relies on two steps of orthogonality between the residual vectors, exhibited unstable behavior. Computational experiments using parallel versions of both variant 1 and the standard conjugate gradient algorithm show that the modified version reduces the execution time by as much as 13% on an Intel iPSC/860 with 16 processors.

REFERENCES

[1] O. AXELSSON AND V. A. BARKER, *Finite Element Solution of Boundary Value Problems*, Academic Press, 1984.

[2] P. CONCUS, G. GOLUB, AND D. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, New York, 1976, pp. 309–322.

[3] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Softw., 15 (89), pp. 1–14.

[4] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 1983.

[5] A. GREENBAUM, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, J. Linear Algebra Appl., 113 (1989), pp. 7–63.

[6] A. GREENBAUM AND Z. STRAKOS, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.

[7] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.

[8] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952).

[9] J. W.-H. LIU AND A. H. SHERMAN, *Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices*, SIAM J. Num. Anal., 13 (1975), pp. 198–213.

[10] G. MEURANT, *Multitasking the conjugate gradient method on the CRAY X-MP/48*, Parallel Comput., 5 (1987), pp. 267–280.

[11] C. C. PAIGE, *Computational variants of the Lanczos method for the eigenproblem*, J. Inst. Maths. Applics., 10 (1972), pp. 373–381.

[12] ———, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, J. Inst. Maths. Applics., 18 (1976), pp. 341–349.

[13] ———, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, J. Linear Algebra Appl., 34 (1980), pp. 235–258.

[14] J. R. RICE AND R. F. BOISVERT, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.

[15] Y. SAAD, *Practical use of polynomial preconditionings for the conjugate gradient method*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 865–881.

[16] Y. SAAD, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1200–1232.

[17] Z. STRAKOS, *On the real convergence rate of the conjugate gradient method*, J. Linear Algebra Appl., 154-156 (1991), pp. 535–549.

[18] H. A. VAN DER VORST, *High performance preconditioning*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1174–1185.

[19] J. VAN ROSENDALE, *Minimizing inner product data dependencies in conjugate gradient iteration*, NASA Contractor Report 172178, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665, 1983.