Lapack Working Note 56
Conjugate Gradient Algorithms with Reduced Synchronization Overhead on Distributed Memory Multiprocessors*

E. F. D'Azevedo[†], V.L. Eijkhout[‡], C. H. Romine[†]

December 3, 1999

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.

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, coupled with an effective preconditioner [18], is generally able to achieve rapid convergence to an accurate solution.

*This work was supported in part by DARPA under contract number DAAL03-91-C-0047 and in part by DOE under contact number DE-AC05-84OR21400

†Mathematical Sciences Section, Oak Ridge National Laboratory, P.O. Box 2008, Bldg. 6012, Oak Ridge, TN 37831-6367

‡Department of Computer Science, University of Tennessee, 104 Ayres Hall, Knoxville, TN 37996

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. 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 $||r_k||_2$ based on the relationship

$$||r_{k+1}||_2^2 = \alpha_k^2 ||Ap_k||_2^2 - ||r_k||_2^2 \tag{1}$$

to be numerically unstable. Meurant [10] proposed using (1) as a predictor for $||r_{k+1}||$ and reevaluate 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.

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 a rearrangement of the conjugate gradient computation that eliminates one communication phase by computing both inner products at once. We show a natural association between this rearrangement and the Lanczos algorithm in §3. A discussion of how this rearrangement of the computation affects the stability properties of the conjugate gradient algorithm and some `MATLAB` numerical experiments on the effectiveness of the rearrangement are included in §4.

2 The conjugate gradient algorithm

In this section we will present several variants of the conjugate gradient algorithm, based on elimination of the A-inner product of the search directions.

2.1 The standard formulation

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

$$Ax = b \ . \tag{2}$$

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.

$$
\begin{array}{l}
\text{For } k = 1, 2, \ldots \\
\qquad \begin{array}{rcl}
\gamma_k &=& \boxed{\langle r_k, r_k \rangle} \\
\beta_k &=& \gamma_k / \gamma_{k-1} \quad (\beta_1 = 0) \\
p_k &=& r_k + \beta_k p_{k-1} \quad (p_1 = r_1) \\
v_k &=& A p_k \\
\sigma_k &=& \boxed{\langle p_k, v_k \rangle} \\
\alpha_k &=& \gamma_k / \sigma_k \\
x_{k+1} &=& x_k + \alpha_k p_k \\
r_{k+1} &=& r_k - \alpha_k v_k \ .
\end{array}
\end{array}
\tag{3, 4}
$$

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 the following intrinsic properties of the CG procedure: the orthogonality of residual vectors (and equivalently the conjugacy of search directions [8, page 420])

$$\frac{\langle r_k, r_{k+1} \rangle}{\langle r_k, r_k \rangle} \equiv \frac{\langle p_k, A p_{k+1} \rangle}{\langle p_k, A p_k \rangle} = 0 \tag{5}$$

and the fact that

$$r_i^t A r_j = 0 \qquad \text{for } |i - j| > 1. \tag{6}$$

2.2 Rearranged method 1

We derive the first rearrangement by expanding $p_k^t A p_k$ by substituting $p_k = r_k + \beta_k p_{k-1}$ for both occurrences of p_k:

$$
\begin{aligned}
\sigma_k &= \langle p_k, v_k \rangle = \langle p_k, A p_k \rangle \\
&= \langle r_k + \beta_k p_{k-1}, A r_k + \beta_k v_{k-1} \rangle \\
&= \langle r_k, A r_k \rangle + \beta_k \langle r_k, v_{k-1} \rangle + \\
&\qquad \beta_k \langle p_{k-1}, A r_k \rangle + \beta_k^2 \langle p_{k-1}, v_{k-1} \rangle \\
\sigma_k &= \langle r_k, A r_k \rangle + 2\beta_k \boxed{\langle r_k, v_{k-1} \rangle} + \beta_k^2 \sigma_{k-1} \ .
\end{aligned}
\tag{7}
$$

From (4) and (5):

$$
\begin{aligned}
r_k &= r_{k-1} - \alpha_{k-1} v_{k-1} \\
\langle r_k, r_k \rangle &= \langle r_k, r_{k-1} \rangle - \alpha_{k-1} \langle r_k, v_{k-1} \rangle \\
\gamma_k &= 0 - \alpha_{k-1} \boxed{\langle r_k, v_{k-1} \rangle} \ .
\end{aligned}
\tag{8}
$$

Therefore by (7), (8) and $\beta_k = \gamma_k/\gamma_{k-1}$,

$$\begin{aligned}
\sigma_k &= \langle r_k, Ar_k\rangle + 2\beta_k(-\gamma_k/\alpha_{k-1}) + \beta_k^2\sigma_{k-1} \\
\sigma_k &= \delta_k - \beta_k^2\sigma_{k-1}, \quad \text{where} \quad \delta_k = \langle r_k, Ar_k\rangle \ . \qquad (9)
\end{aligned}$$

2.3 Rearranged methods 2 and 3

If we expand the occurrences of p_k in $p_k^t A p_k$ one at a time we find different arrangements of the algorithm.

$$\begin{aligned}
\sigma_k &= \langle p_k, Ap_k\rangle \\
&= \langle r_k + \beta_k p_{k-1}, Ap_k\rangle = \langle r_k, Ap_k\rangle \\
&= \langle r_k, A(r_k + \beta_k p_{k-1})\rangle \\
&= \langle r_k, Ar_k\rangle + \beta_k \langle r_k, Ap_{k-1}\rangle \\
\sigma_k &= \gamma_k + \beta_k \epsilon_k^{(1)} \qquad \text{where } \epsilon_k^{(1)} = \langle r_k, Ap_{k-1}\rangle \qquad (10)
\end{aligned}$$

Thus we find a rearrangement of the conjugate gradient method where we compute inner products $r_k^t A r_k$, $r_k^t r_k$, and $r_k^t A p_{k-1}$ simultaneously and compute $\sigma_k = p_k^t A p_k$ by recurrence formula (10).

Successive expansion of $p_k = r_k + \beta_k p_{k-1}$ gives

$$p_k = r_k + \beta_k r_{k-1} + \beta_k\beta_{k-1} r_{k-2} + \cdots.$$

Using equation (6) we then find

$$\begin{aligned}
\sigma_k &= \ldots \\
&= \langle r_k, A(r_k + \beta_k r_{k-1} + \beta_k\beta_{k-1} r_{k-2} + \cdots)\rangle \\
&= \langle r_k, Ar_k\rangle + \beta_k \langle r_k, Ar_{k-1}\rangle \qquad (11)
\end{aligned}$$

This gives us a rearrangement of the conjugate gradient method where we compute inner products $r_k^t A r_k$, $r_k^t r_k$, and $r_k^t A r_{k-1}$ simultaneously and compute $\sigma_k = p_k^t A p_k$ by recurrence formula (11).

2.4 The modified conjugate gradient method

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

$$\begin{aligned}
r_1 &= b, \quad \gamma_1 = \langle r_1, r_1\rangle, \quad p_1 = r_1, \quad v_1 = Ap_1 \\
\sigma_1 &= \langle p_1, v_1\rangle, \quad x_2 = (\gamma_1/\sigma_1)p_1
\end{aligned}$$

For $k = 2, 3, \ldots$

$$s_k = Ar_k$$

$$
\begin{aligned}
\text{Compute } \gamma_k &= \boxed{\langle r_k, r_k\rangle} \text{ and } \delta_k = \boxed{\langle r_k, s_k\rangle} \\
\text{and } \epsilon_k^{(1)} &= \boxed{\langle r_k, v_{k-1}\rangle} \qquad \text{for method 2 only} \\
\text{and } \epsilon_k^{(2)} &= \boxed{\langle r_k, s_{k-1}\rangle} \qquad \text{for method 3 only} \\
\beta_k &= \gamma_k/\gamma_{k-1} \\
p_k &= r_k + \beta_k p_{k-1} \\
v_k &= s_k + \beta_k v_{k-1} \quad (v_k \equiv Ap_k) \\
\sigma_k &= \begin{cases} \delta_k - \beta_k^2\sigma_{k-1} & \text{for method 1} \\ \delta_k + \beta_k\epsilon_k^{(1)} & \text{for method 2} \\ \delta_k + \beta_k\epsilon_k^{(2)} & \text{for method 3} \end{cases} \\
\alpha_k &= \gamma_k/\sigma_k \\
x_{k+1} &= x_k + \alpha_k p_k \\
r_{k+1} &= r_k - \alpha_k v_k \ .
\end{aligned} \tag{12}
$$

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 methods. Methods 2 and 3 require an additional inner product and storage for v_{k-1} and s_{k-1} respectively.

3 Stability proof of method 1

In this section we present a proof of the stability of the first method, based on an equivalence of (7) to a stable recurrence arising from the symmetric Lanczos process. The outline of the proof is as follows.

First we recall the fact that the conjugate gradients method is a tridiagonalization procudure. Recurrence relation (7) in method 1 is then shown to be equivalent to the pivot recurrence for the factorization of this tridiagonal matrix. The cornerstone of the proof is that the symmetric tridiagonal matrix produced by the symmetric Lanczos process gives the same pivot recurrence. If the original coefficient matrix is symmetric positive definite this recurrence is stable, hence the scalar recurrence in rearranged method 1 is stable.

The two basic vector updates of the conjugate gradients method

$$
r_{k+1} = r_k - Ap_k\alpha_k, \qquad p_{k+1} = r_{k+1} + \beta_{k+1}p_k
$$

can be summarized as

$$
R(I-J) = APD, \qquad PU = R
$$

where R and P are matrices containing the vector sequences $\{r_k\}$ and $\{p_k\}$ as columns, and

$$
J = (\delta_{i,j+1}), \qquad D = \mathrm{diag}(\alpha_k), \qquad u_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\beta_j & \text{if } i+1 = j \\ 0 & \text{otherwise} \end{cases} .
$$

With $H = (I - J)D^{-1}U$ (a tridiagonal matrix) we can write this as $AR = RH$. For the elements of H we find:

$$h_{nn} = \frac{r_n^t A r_n}{r_n^t r_n}, \qquad h_{n+1n} = \frac{r_n^t r_n}{r_{n+1}^t r_{n+1}} h_{nn+1}, \tag{13}$$

where we have used only orthogonality properties and the symmetry of A.

The pivots d_{kk} in the factorization of H satisfy a recurrence

$$\alpha_{n+1}^{-1} = h_{n+1n+1} - h_{n+1n}\alpha_{nn}h_{nn+1},$$

or, using $\alpha_k = r_k^t r_k / p_k^t A p_k$ and equation (13)

$$\frac{p_{n+1}^t A p_{n+1}}{r_{n+1}^t r_{n+1}} = \frac{r_{n+1}^t A r_{n+1}}{r_{n+1}^t r_{n+1}} - h_{nn+1}^2 \frac{r_n^t r_n}{r_{n+1}^t r_{n+1}} \alpha_{nn}.$$

Divide by $r_{n+1}^t r_{n+1}$ and use the fact that $h_{nn+1} = \alpha_n^{-1}\beta_n$:

$$\begin{aligned} p_{n+1}^t A p_{n+1} &= r_{n+1}^t A r_{n+1} - h_{nn+1}^2 r_n^t r_n \alpha_n \\ &= r_{n+1}^t A r_{n+1} - \beta_n^2 r_n^t r_n \alpha_n^{-1} \\ &= r_{n+1}^t A r_{n+1} - \beta_n^2 p_n^t A p_n. \end{aligned}$$

This shows that the recursive computation of $p_n^t A p_n$ in rearranged method 1 is equivalent by mere formula substitution to the pivot recursion for H. (We gloss over the fact that formulas (13) are based on properties of the conjugate gradients method formulated as a three-term recurrence, instead of as the usual coupled pair of two-term recurrences. We assume comparable numerical stability properties for these two variants of the basic method.)

The Lanczos method for constructing an orthonormal matrix Q that reduces A to symmetric tridiagonal form T by $Q^t A Q = T$ is easily constructed from the conjugate gradients method by letting

$$q_n = c_{nn}^{-1} r_n \qquad \text{where} \qquad c_{nn} = \|r_n\|.$$

Hence the orthonormal tridiagonalization can be written as

$$AQ = QT \qquad \text{with} \qquad Q = RC^{-1}, \qquad T = C^{-1}HC.$$

Using the fact that H is tridiagonal and that $H = C^{-1}TC$ with C diagonal, we find that the factorization of T generates the same pivot sequence. If A, and therefore T, is symmetric positive definite, this pivot recursion, and therefore recurrence (7) is stable.

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 Eijkhout's rearrangement, $\langle r_k, v_{k-1} \rangle$ 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]

$$\lambda_i = \lambda_1 + \frac{i-1}{n-1}(\lambda_n - \lambda_1)\rho^{n-i}, \qquad i = 2, \ldots, n, \quad \rho \in (0, 1). \tag{14}$$

We have used $n = 100$, $\lambda_1 = \texttt{1E-3}$, $\kappa = \lambda_n/\lambda_1 = \texttt{1E5}$ 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 $[a, b]$

$$\lambda_i = \frac{(b-a)}{2} \cos\left(\frac{\pi/2 + (i-1)\pi}{n}\right) + \frac{(b+a)}{2}, \quad i = 1, \ldots, n. \tag{15}$$

We have used $n = 100$, $a = 1$, $b = \texttt{1E5}$.

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[1] 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

$$\frac{\langle \tilde{x} - x_k, A(\tilde{x} - x_k) \rangle^{1/2}}{\langle \tilde{x} - x_0, A(\tilde{x} - x_0) \rangle^{1/2}}, \qquad \tilde{x} = A^{-1}b. \tag{16}$$

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

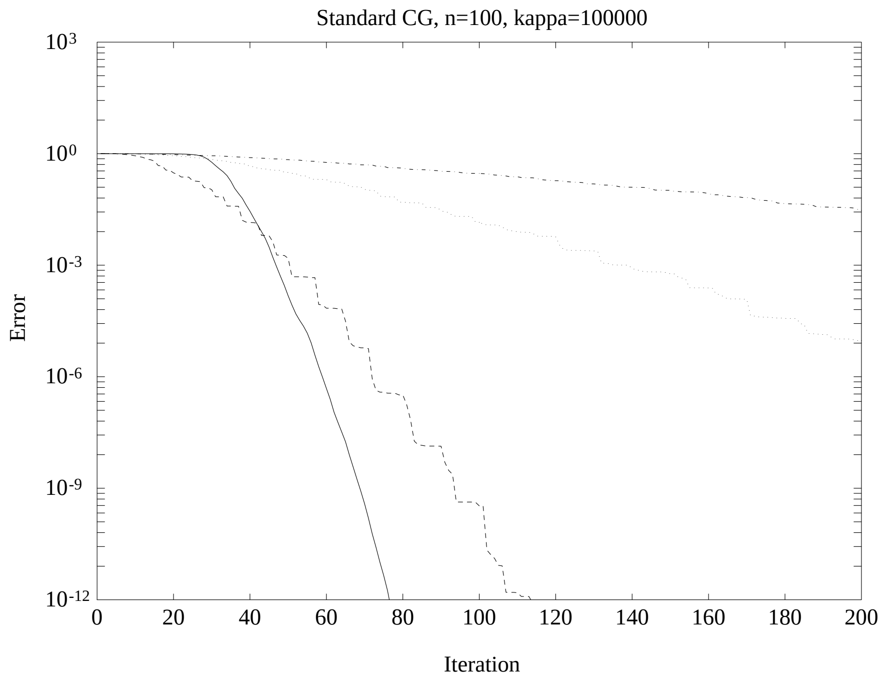

Figure 1: Classical CG on Test 1. Dashed curve: $\rho = 0.6$; dotted curve: $\rho = 0.8$; dash-dot curve: $\rho = 0.9$; solid curve: $\rho = 1$.

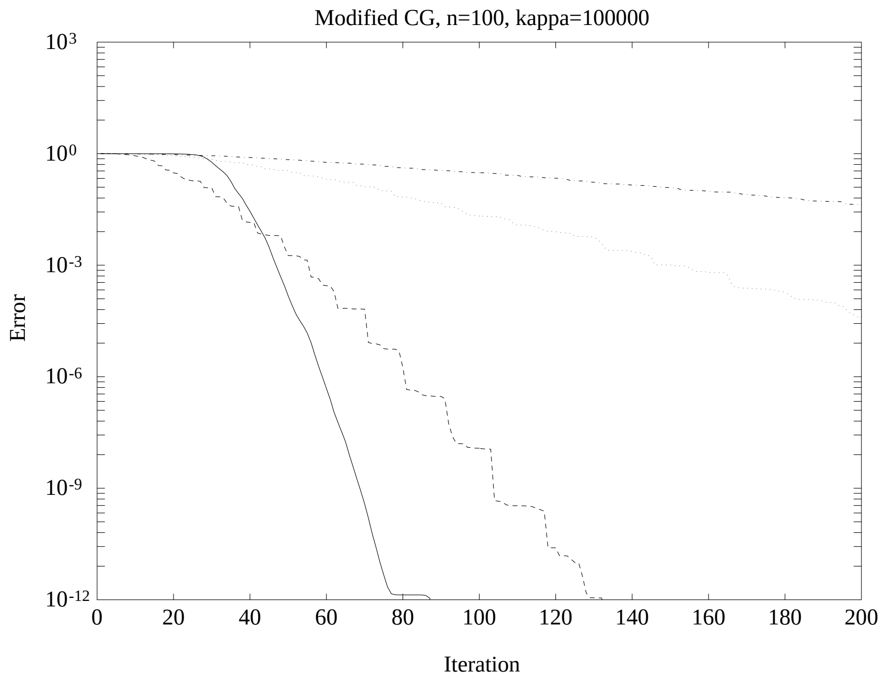

Figure 2: Modified CG on Test 1. Dashed curve: $\rho = 0.6$; dotted curve: $\rho = 0.8$; dash-dot curve: $\rho = 0.9$; solid curve: $\rho = 1$.

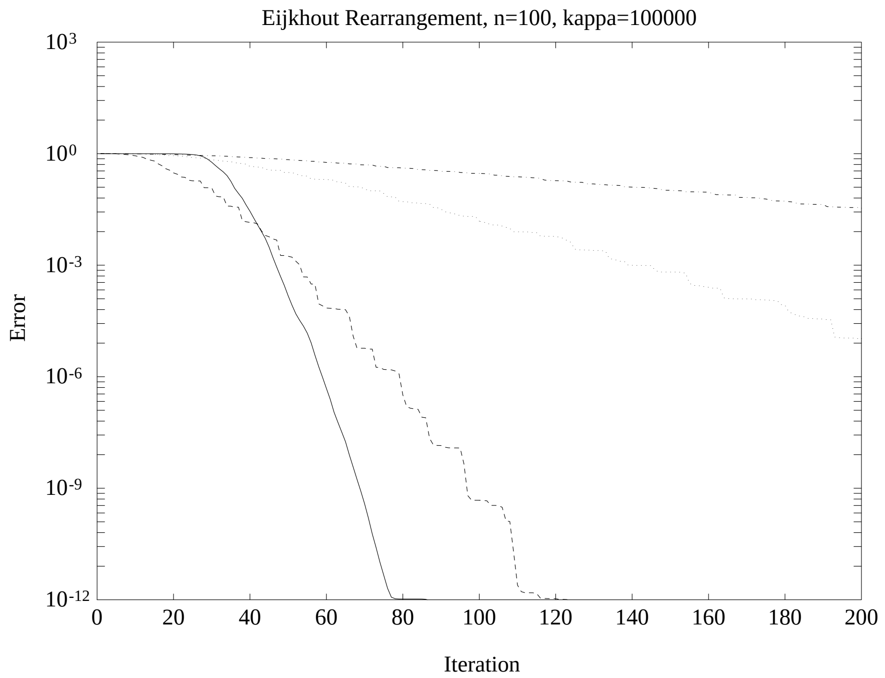

Figure 3: Eijkhout Rearrangement on Test 1. Dashed curve: $\rho = 0.6$; dotted curve: $\rho = 0.8$; dash-dot curve: $\rho = 0.9$; solid curve: $\rho = 1$.

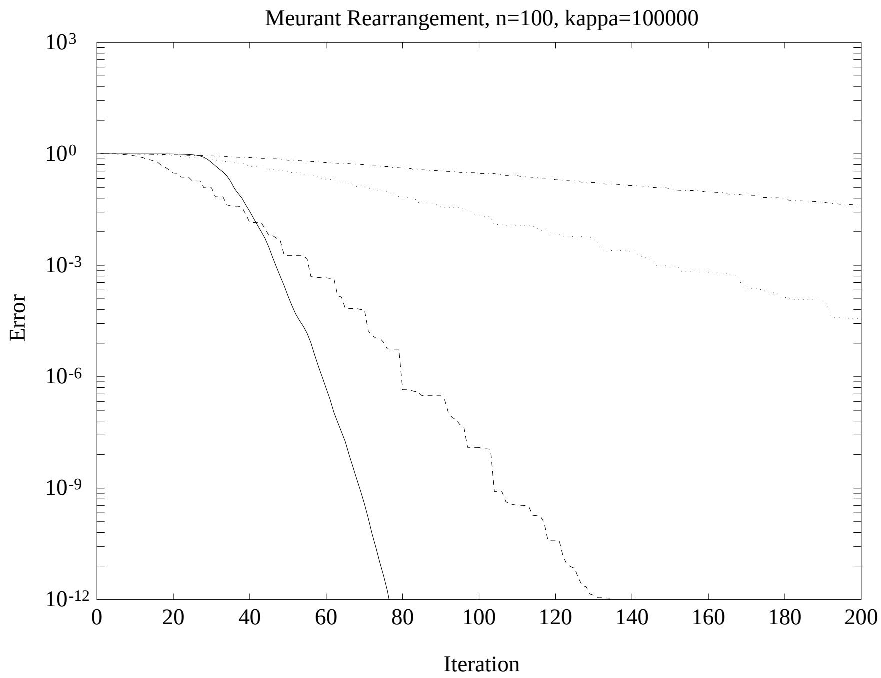

Figure 4: Meurant Rearrangement on Test 1. Dashed curve: $\rho = 0.6$; dotted curve: $\rho = 0.8$; dash-dot curve: $\rho = 0.9$; solid curve: $\rho = 1$.

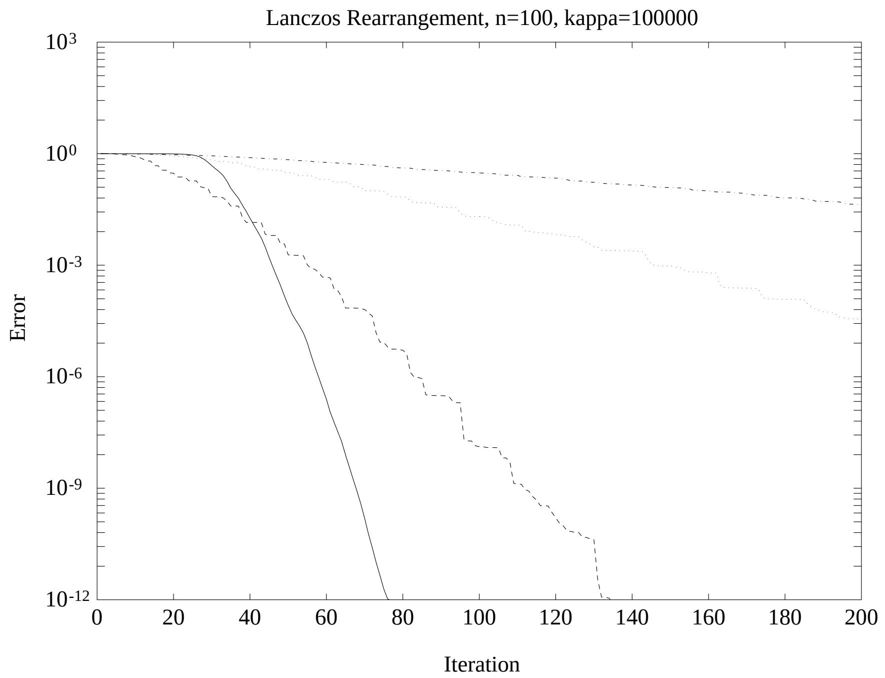

Figure 5: Lanczos Rearrangement on Test 1. Dashed curve: $\rho = 0.6$; dotted curve: $\rho = 0.8$; dash-dot curve: $\rho = 0.9$; solid curve: $\rho = 1$.

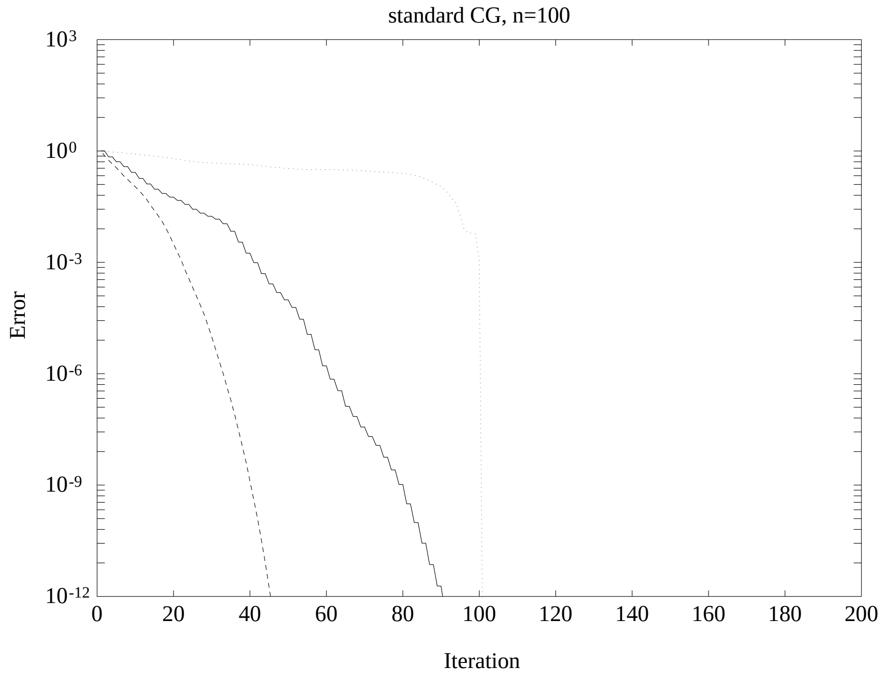

Figure 6: Classical CG on Tests 2–4. Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.

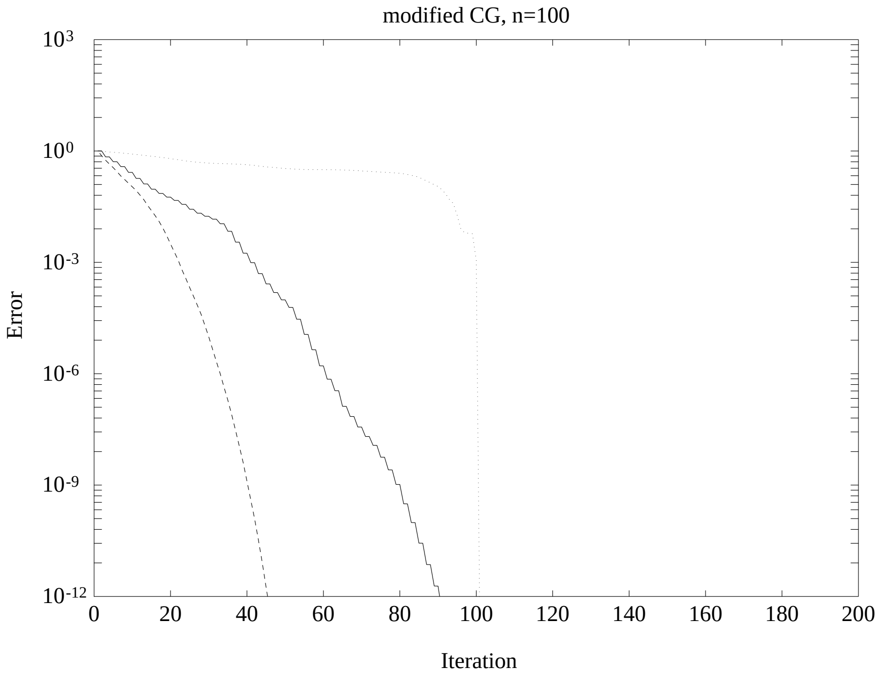

Figure 7: Modified CG on Tests 2–4. Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.

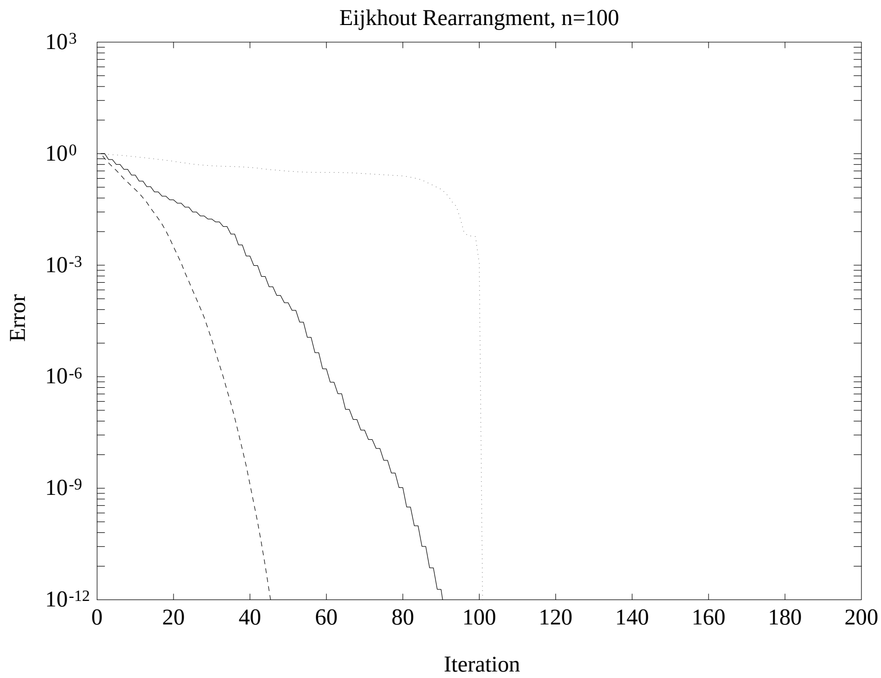

Figure 8: Eijkhout Rearrangement on Tests 2–4. Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.

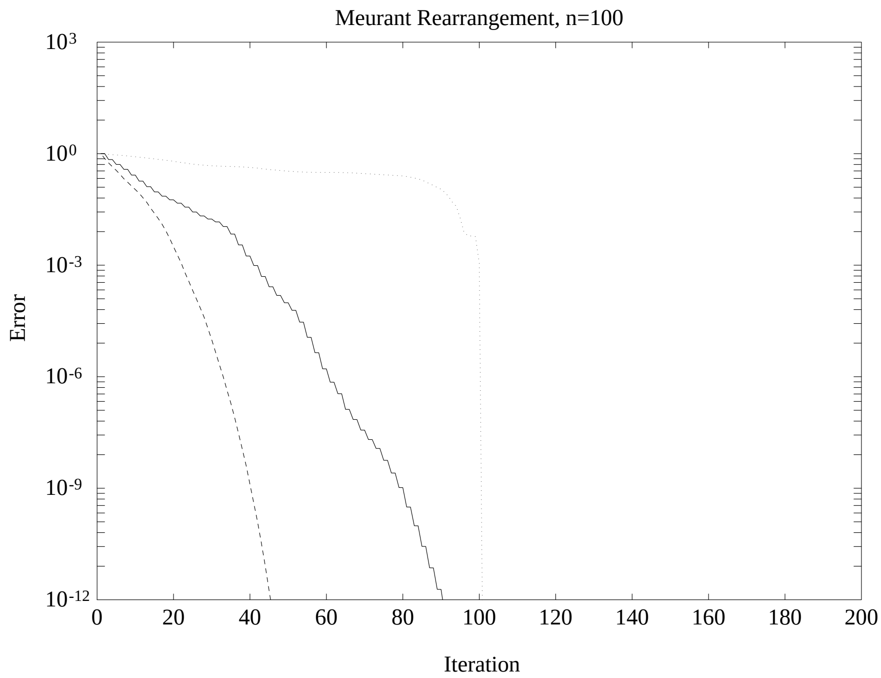

Figure 9: Meurant Rearrangement on Tests 2–4. Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.

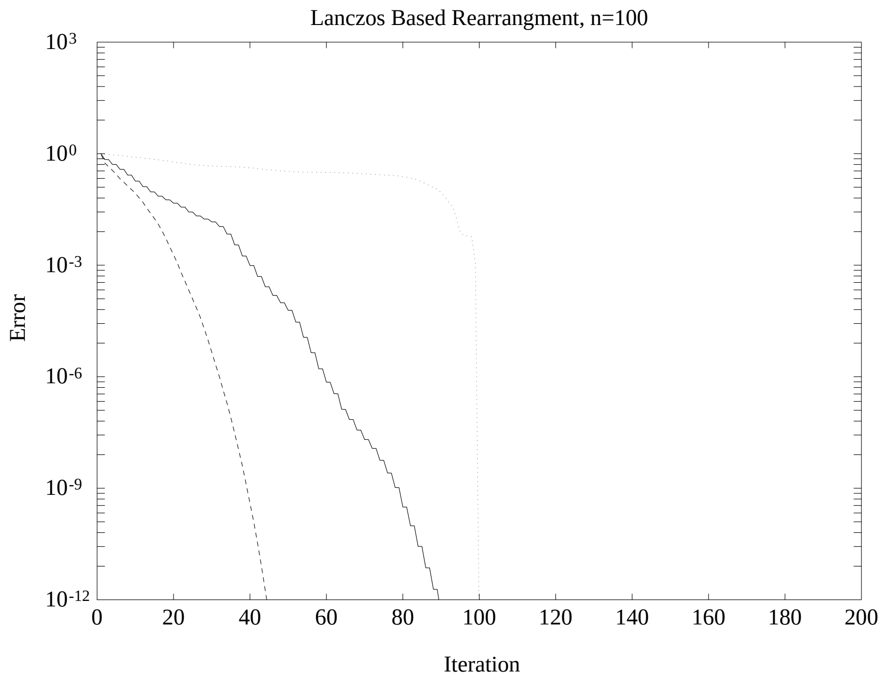

Figure 10: Lanczos Rearrangement on Tests 2–4. Solid curve: Test 2; dashed curve: Test 3; dotted curve Test 4.

Table 1: Description of test problems.

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

Figures 1–5, display the convergence results from Test 1. Note that for $\rho = 0.8, 0.9$ both the standard and modified CG procedures exhibit similar slow convergence behavior. Figures 6–10 display the convergence results on Tests 2–4. For Test 1 with $\rho = 0.6$, the standard CG algorithm shows the best convergence properties. Eijkhout's rearrangement has slightly better stability properties than modified CG. The other results are essentially the same.

All the results on Tests 2–4 again show similar convergence behavior among the standard CG and the different rearrangements of CG.

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 modified CG. The test matrices are chosen from the Harwell-Boeing Test Collection [3]. The experiments are performed on 16 nodes of the iPSC/860 hypercube. Each matrix is first reordered by the bandwidth reducing Reverse Cuthill-McKee ordering [9]. The matrix is 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 are used, and convergence is assumed when

$$||r_k||_2 \leq 10^{-8} ||r_0||_2. \tag{17}$$

The conjugate gradient procedure is rarely used without some form of preconditioning to accelerate convergence. In the tests described below, we use 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 $M = \text{diag}(M_1, M_2, \ldots, M_p)$, 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. This preconditioner requires no added communication among the processors when implemented in parallel.

Table 1 is a brief description of the problems selected from the Harwell-Boeing Test Collection. Table 2 shows the number of iterations and time (in

Table 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

seconds) required to solve the corresponding problems. In all cases, the modified CG shows an improvement in the time required for solution, ranging from 5% to 13%. Moreover, the modified CG rearrangement shows no unstable behavior since it takes almost exactly the same number of iterations as standard CG.

6 Conclusion

We have presented a rearrangement of the standard conjugate gradient procedure that eliminates one synchronization point by performing two inner products at once. The rearrangement has a natural connection with the Lanczos process for solving linear equations. Although not a proof, `MATLAB` simulations indicate that the rearrangement is stable. Moreover, computational experiments using parallel versions of both the modified and standard conjugate gradient algorithms 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 J. Bunch and D. Rose, editors, *Sparse Matrix Computations*, pages 309–322. Academic Press, New York, 1976.

[3] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15(1):1–14, 89.

[4] Gene H. Golub and Charles F. van Loan. *Matrix Computations.* John Hopkins University Press, Baltimore, Maryland, 1983.

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

[6] A. Greenbaum and Z. Strakos. Predicting the behavior of finite precision Lanczos and conjugate gradient computations. *SIAM J. Matrix Anal. Appl.*, 13(1):121–137, 1992.

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

[8] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49(6), 1952.

[9] Joseph W-H Liu and Andrew H. Sherman. Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, 13:198–213, 1975.

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

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

[12] C. C. Paige. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *J. Inst. Maths. Applics.*, 18:341–349, 1976.

[13] C. C. Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *J. Linear Algebra Appl.*, 34:235–258, 1980.

[14] J. R. Rice and R. F. Boisvert. *Solving Elliptic Problems Using ELLPACK*. Springer-Verlag, New York, 1985.

[15] Yocef Saad. practical use of polynomials preconditionings for the conjugate gradient method. *SIAM J. Sci. Statist. Comput.*, 6:865–881, 1985.

[16] Youcef Saad. Krylov subspace methods on supercomputers. *SIAM J. Sci. Statist. Comput.*, 10(6):1200–1232, 1989.

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

[18] Henk A. van der Vorst. High performance preconditioning. *SIAM J. Sci. Statist. Comput.*, 10(6):1174–1185, 1989.

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