Preserving Symmetry in Preconditioned Krylov Subspace Methods*

T. F. Chan[†], E. Chow[‡], Y. Saad[‡] and M. C. Yeung[†]

October 31, 1996

Abstract

We consider the problem of solving a linear system $Ax = b$ when A is nearly symmetric and when the system is preconditioned by a symmetric positive definite matrix M. In the symmetric case, one can recover symmetry by using M-inner products in the conjugate gradient (CG) algorithm. This idea can also be used in the nonsymmetric case, and near symmetry can be preserved similarly. Like CG, the new algorithms are mathematically equivalent to split preconditioning, but do not require M to be factored. Better robustness in a specific sense can also be observed. When combined with truncated versions of iterative methods, tests show that this is more effective than the common practice of forfeiting near-symmetry altogether.

1 Introduction

Consider the solution of the linear system

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

by a preconditioned Krylov subspace method. Assume at first that A is symmetric positive definite (SPD) and let M be an SPD matrix that is a preconditioner for the matrix A. Then one possibility is to solve either the left-preconditioned system

$$M^{-1}Ax = M^{-1}b \tag{2}$$

*This work was supported by the NSF under contract NSF/CCR-9214116, by the ONR under contract ONR-N00014-92-J-1890, and by the ARO under contract DAAL-03-9-C-0047 (Univ. of Tenn. subcontract ORA 4466.04, Amendment 1 and 2).

[†]Department of Mathematics, University of California Los Angeles, 405 Hilgard Avenue, Los Angeles, California, 90095-1555 (chan@math.ucla.edu and myeung@math.ucla.edu).

[‡]Department of Computer Science and Minnesota Supercomputer Institute, University of Minnesota, 4-192 EE/CSci Bldg., 200 Union St., S.E., Minneapolis, Minnesota, 55455-0154 (chow@cs.umn.edu and saad@cs.umn.edu).

or the right-preconditioned system

$$AM^{-1}u = b, \quad x = M^{-1}u \ . \tag{3}$$

Both of these systems have lost their symmetry. A remedy exists when the preconditioner M is available in factored form, e.g., as an incomplete Choleski factorization,

$$M = LL^T$$

in which case a simple way to preserve symmetry is to 'split' the preconditioner between left and right, i.e., we could solve

$$L^{-1}AL^{-T}u = L^{-1}b, \quad x = L^{-T}u \ , \tag{4}$$

which involves a symmetric positive definite matrix. This can also be done when M can be factored as $M = M^{1/2} \times M^{1/2}$. Unfortunately, the requirement that M be available in factored forms is often too stringent.

However, this remedy is not required. As is well-known, we can preserve symmetry by using a different inner product. Specifically, we observe that $M^{-1}A$ is self-adjoint for the M-inner product,

$$(x, y)_M = (Mx, y) = (x, My)$$

since we have

$$(M^{-1}Ax, y)_M = (Ax, y) = (x, Ay) = (x, M(M^{-1}A)y) = (x, M^{-1}Ay)_M \ .$$

If we rewrite the CG-algorithm for this new inner product, denoting by $r_j = b - Ax_j$ the original residual and by $z_j = M^{-1}r_j$ the residual for the preconditioned system, we would obtain the following algorithm, see, e.g., [4].

Algorithm **1.1** *Preconditioned Conjugate Gradient*

1. *Start:* *Compute* $r_0 = b - Ax_0$*;* $z_0 = M^{-1}r_0$*; and* $p_0 = z_0$.
2. *Iterate:* *For* $j = 0, 1, \ldots,$ *until convergence do,*
   (a) $\alpha_j = (r_j, z_j)/(Ap_j, p_j)$
   (b) $x_{j+1} = x_j + \alpha_j p_j$
   (c) $r_{j+1} = r_j - \alpha_j Ap_j$
   (d) $z_{j+1} = M^{-1}r_{j+1}$
   (e) $\beta_j = (r_{j+1}, z_{j+1})/(r_j, z_j)$
   (f) $p_{j+1} = z_{j+1} + \beta_j p_j$

Note that even though M-inner products are used, we do not need to multiply by M which may not be available explicitly; we only need to solve linear systems with the matrix coefficient M. It can also be seen that with a change of variables, the iterates produced by this algorithm are identical to both those of CG with split preconditioning, and CG with right-preconditioning using the M^{-1}-inner product. All three of these algorithms are thus equivalent.

The question we now raise is the following. When A is only nearly symmetric, it is often the case that there exists a preconditioner M which is SPD. In this situation, the use of either of the forms (2) or (3) is just as unsatisfactory as in the fully symmetric case. Indeed, whatever degree of symmetry was available in A is now entirely lost. Although the above remedy based on M-inner products is always used in the symmetric case, it is rather surprising that this problem is seldom ever mentioned in the literature for the nearly symmetric case. In the nonsymmetric case, when M exists in factored form, some form of balancing can also be achieved by splitting the preconditioner from left and right. However, there does not seem to have been much work done in exploiting M-inner products even when M is not available in factored form. This dichotomy between the treatment of the symmetric and the nonsymmetric cases is what motivated this study.

Ashby *et. al.* have fully considered the case of using alternate inner products when the matrix A is symmetric. Work of which we are aware that consider the use of alternate inner products when A is near-symmetric are Young and Jea [9] and Meyer [5]. In the latter, the $A^T M^{-1} A$ inner product (with M SPD) is used with ORTHOMIN and ORTHODIR.

This paper is organized as follows. In Section 2, it is shown how alternative inner products may be used to preserve symmetry in GMRES. Section 3 considers the use of truncated iterative methods when the preconditioned system is close to being symmetric. This has been hypothesized by many authors, for example, Axelsson [2] and Meyer [5]. In Section 4 we consider the symmetrically preconditioned Bi-CG algorithm. Section 5 tests these algorithms numerically using a Navier-Stokes problem parameterized by the Reynolds number, and thus nearness to symmetry. We conclude this paper in Section 6.

2 Symmetric preconditioning in GMRES

When A is nearly symmetric, split preconditioning may be used to preserve the original degree of symmetry. Alternatively, left-preconditioning with the M-inner product, or right-preconditioning with the M^{-1}-inner product may be used. These latter two alternatives will be developed for the Arnoldi process, and used as the basis of 'symmetric' preconditioned versions of GMRES. Like CG, it will be shown that these symmetric versions are mathematically equivalent to split preconditioning, but do not require the preconditioner to be symmetrically factored. We begin by exploring the options and implementation issues associated with left symmetric preconditioning.

2.1 Left symmetric preconditioning

The GMRES algorithm is based on the Arnoldi process. Without preconditioning, the Arnoldi algorithm based on the classical Gram-Schmidt process is as follows.

ALGORITHM **2.1** *Arnoldi-Classical Gram-Schmidt*

1. *Choose a vector* v_1 *of norm 1.*
2. *For* $j = 1, 2, \ldots, m$ *do:*
3. $\quad z = Av_j$
4. $\quad$ *Compute* $h_{ij} = (z, v_i)$ *for* $i = 1, 2, \ldots, j$
5. $\quad z = z - \sum_{i=1}^{j} h_{ij} v_i$
6. $\quad h_{j+1,j} = \|z\|_2$
7. $\quad$ *If* $h_{j+1,j} = 0$ *then Stop.*
8. $\quad v_{j+1} = z / h_{j+1,j}$
9. *EndDo*

Consider here the case when A is left-preconditioned, i.e., the matrix A involved in the algorithm is to be replaced by $B = M^{-1}A$, where M is some preconditioner, which we assume to be SPD. We wish to define a procedure to implement the above algorithm for the matrix $B = M^{-1}A$, using M-inner products, and if possible, avoid additional matrix-vector products, e.g., with M. Once this is accomplished, we will define the corresponding GMRES procedure.

In the preconditioned case, it is customary to define the intermediate vectors in the product $z = M^{-1}Av_j$ as

$$w_j = Av_j \ , \tag{5}$$

which is then preconditioned to get

$$z_j = M^{-1} w_j \ . \tag{6}$$

We now reformulate the operations of the above algorithm for $B = M^{-1}A$, with the M-inner product. Only the computation of the inner products changes. In the classical Gram-Schmidt formulation, we would first compute the scalars h_{ij} in Line 4 of Algorithm 2.1,

$$h_{ij} = (z_j, v_i)_M = (Mz_j, v_i) = (w_j, v_i) \ , \quad i = 1, \ldots, j. \tag{7}$$

Then we would modify the vector z_j to obtain the next Arnoldi vector (before normalization),

$$\hat{z}_j = z_j - \sum_{i=1}^{j} h_{ij} v_i \ . \tag{8}$$

To complete the orthonormalization step we must normalize the final $\hat{z}_j$. Because of the M-orthogonality of $\hat{z}_j$ versus all previous v_i's we observe that

$$(\hat{z}_j, \hat{z}_j)_M = (z_j, \hat{z}_j)_M = (M^{-1}w_j, \hat{z}_j)_M = (w_j, \hat{z}_j) \ . \tag{9}$$

Thus, the desired M-norm can be computed according to (9) and then computing

$$h_{j+1,j} = (\hat{z}_j, w_j)^{1/2} \quad \text{and} \quad v_{j+1} = \hat{z}_j / h_{j+1,j}. \tag{10}$$

One potentially serious difficulty with the above procedure is that the inner product $(\hat{z}_j, \hat{z}_j)_M$ as computed by (9) may be negative in the presence of round-off. There are two remedies. First, we can compute this M-norm explicitly at the expense of an additional matrix-vector multiplication with M, i.e., from

$$(\hat{z}_j, \hat{z}_j)_M \;=\; (M\hat{z}_j, \hat{z}_j) \ .$$

As was pointed out earlier, this is undesirable, since the operator M is often not available explicitly. Indeed in many cases, only the preconditioning operation M^{-1} is available from a sequence of operations, as is the case for multigrid preconditioning. Another difficulty with computing h_{ij} with (7) is that it is not immediately amenable to a modified Gram-Schmidt implementation. Indeed, consider the first step of a hypothetical modified Gram-Schmidt step, which consists of M-orthonormalizing z against v_1,

$$h_{i1} = (z, v_1)_M \ , \qquad \hat{z} = z - h_{i1} v_1 \ .$$

As was observed, the inner product $(z, v_1)_M$ is equal to (w, v_1) which is computable. Now we need $M\hat{z}$ to compute $(z - h_{i1}v_1, v_i)_M$ in the modified Gram-Schmidt process. However, no such vector is available, and we can only compute the $(z, v_i)_M$ of classical Gram-Schmidt.

An alternative is to save the set of vectors Mv_i (again, not computed by multiplying by M) which would allow us to accumulate inexpensively both the vector $\hat{z}_j$ and the vector $\hat{w}_j$ via the relation

$$\hat{w}_j \equiv M\hat{z}_j = w_j - \sum_{i=1}^{j} h_{ij} M v_i \ ,$$

which is obtained from (8). Now the inner product $(\hat{z}_j, \hat{z}_j)_M$ is given by

$$(\hat{z}_j, \hat{z}_j)_M = (M^{-1}\hat{w}_j, M^{-1}\hat{w}_j)_M = (M^{-1}\hat{w}_j, \hat{w}_j) \ .$$

In this form, this inner product is guaranteed to be nonnegative as desired. This leads to the following algorithm.

Algorithm **2.2** *Arnoldi-Classical Gram-Schmidt and M-inner products*

1. *Choose a vector w_1 such that $v_1 = M^{-1}w_1$ has M-norm 1.*
2. *For $j = 1, 2, \ldots, m$ do:*
3. $\quad w = Av_j$
4. $\quad$ *Compute $h_{ij} = (w, v_i)$ for $i = 1, 2, \ldots, j$*
5. $\quad \hat{w} = w - \sum_{i=1}^{j} h_{ij} w_i$
6. $\quad z = M^{-1}\hat{w}$.
7. $\quad h_{j+1,j} = (z, \hat{w})^{1/2}$. *If* $h_{j+1,j} = 0$ *then Stop.*
8. $\quad w_{j+1} = \hat{w}/h_{j+1,j}$
9. $\quad v_{j+1} = z/h_{j+1,j}$
10. *EndDo*

As is noted, the above algorithm requires that we save two sets of vectors: the v_j's and the w_i's. The v_i's form the needed Arnoldi basis, and the w_i's are required when computing the vector $\hat{w}_j$ in Line 5. If we do save these two sets of vectors we can also now easily formulate the algorithm with the modified Gram-Schmidt version of the Arnoldi procedure.

ALGORITHM **2.3** *Arnoldi-Modified Gram-Schmidt and M-inner products*

1. *Choose a vector w_1 such that $v_1 = M^{-1}w_1$ has M-norm 1.*
2. *For $j = 1, 2, \ldots, m$ do:*
3. $\quad w = Av_j$
4. $\quad$ *For $i = 1, \ldots, j$ do:*
5. $\quad\quad h_{ij} = (w, v_i)$
6. $\quad\quad w = w - h_{ij}w_i$
7. $\quad$ *EndDo*
8. $\quad z = M^{-1}w$
9. $\quad h_{j+1,j} = (z, w)^{1/2}$. *If $h_{j+1,j} = 0$ then Stop.*
10. $\quad w_{j+1} = w/h_{j+1,j}$
11. $\quad v_{j+1} = z/h_{j+1,j}$
12. *EndDo*

2.2 Right symmetric preconditioning

The matrix AM^{-1} is self-adjoint with the M^{-1}-inner product. The situation for right-preconditioning with this inner product is much simpler, mainly because $M^{-1}z$ is available when z needs to be normalized in the M^{-1} norm. However, $M^{-1}z$ is normally computed at the next iteration in the standard Arnoldi algorithm; a slight reorganization of the Arnoldi-Modified Gram-Schmidt algorithm yields the following.

ALGORITHM **2.4** *Arnoldi-Modified Gram-Schmidt and M^{-1}-inner products*

1. *Choose a vector v_1 of M^{-1}-norm 1, compute $w_1 = M^{-1}v_1$*
2. *For $j = 1, 2, \ldots, m$ do:*
3. $\quad z = Aw_j$
4. $\quad$ *For $i = 1, \ldots, j$ do:*
5. $\quad\quad h_{ij} = (z, w_i)$
6. $\quad\quad z = z - h_{ij}v_i$
7. $\quad$ *EndDo*
8. $\quad w = M^{-1}z$
9. $\quad h_{j+1,j} = (z, w)^{1/2}$. *If $h_{j+1,j} = 0$ then Stop.*
10. $\quad w_{j+1} = w/h_{j+1,j}$
11. $\quad v_{j+1} = z/h_{j+1,j}$
12. *EndDo*

Note that the preconditioned vector is computed in Line 8, while in the standard algorithm it is computed before Line 3. Again, both the v's and the w's need to be saved, where $w_j = M^{-1}v_j$ in this case.

The additional storage of the w's, however, makes this algorithm naturally 'flexible,' i.e., it accommodates the situation where M varies at each step as when $M^{-1}v$ is the result of some unspecified computation. If M^{-1} is not a constant operator, then a basis for the right-preconditioned Krylov subspace cannot be constructed from the v's alone. However, the vectors $w_j = M_j^{-1}v_j$ do form a basis for this subspace, where M_j^{-1} denotes the preconditioning operation at the j-th step. The use of this extra set of vectors is exactly how the standard flexible variant of GMRES is implemented [6].

2.3 Using M-inner products in GMRES

The vectors v_i form an orthonormal basis of the Krylov subspace. In the following we denote by $V_m = [v_1, \ldots, v_m]$ the matrix whose column vectors are the vectors v_i produced by the Arnoldi-Modified Gram-Schmidt algorithm with M-inner products (Algorithm 2.3). A similar notation will be used for the matrix W_m. We also denote by $\bar{H}_m$ the $(m+1) \times m$ upper Hessenberg matrix whose nonzero entries h_{ij} are defined by the algorithm. H_m denotes the top $m \times m$ portion of $\bar{H}_m$. These matrices satisfy a number of relations similar to the ones obtained from using the standard Euclidean inner product.

Proposition 2.1 *The following properties are satisfied by the vectors v_i and w_i in Algorithm 2.3:*

1. $M^{-1}AV_m = V_{m+1}\bar{H}_m$,
2. $AV_m = W_{m+1}\bar{H}_m$,
3. $V_m^T M V_m = I$,
4. $W_m^T M^{-1} W_m = I$,
5. *If A is Hermitian then H_m is Hermitian and tridiagonal.*

Consider now the implementation of a GMRES procedure based on the orthogonalization process of Algorithm 2.3. Since we are using M-inner products we should be able to minimize the M-norm of the residual vectors $M^{-1}b - M^{-1}Ax$ over all vectors of the affine subspace $x_0 + K_m$ in which,

$$K_m = \operatorname{span}\{z_0, M^{-1}Az_0, \ldots, (M^{-1}A)^{m-1}z_0\} \tag{11}$$

where $z_0 = M^{-1}r_0$, and $r_0 = b - Ax_0$. Define $\beta = \|r_0\|_M$ and e_1 as the first coordinate vector. Then we have,

$$\begin{aligned} b - Ax &= b - A(x_0 + V_m y) \\ &= r_0 - AV_m y \\ &= r_0 - W_{m+1}\bar{H}_m y \\ &= W_{m+1}(\beta e_1 - \bar{H}_m y). \end{aligned}$$

Therefore we have the equality,

$$
\begin{aligned}
\|M^{-1}(b-Ax)\|_M^2 &= \|M^{-1}W_{m+1}(\beta e_1 - \bar{H}_m y)\|_M^2 \\
&= (M^{-1}W_{m+1}(\beta e_1 - \bar{H}_m y), W_{m+1}(\beta e_1 - \bar{H}_m y)) \\
&= (W_{m+1}^T M^{-1} W_{m+1}(\beta e_1 - \bar{H}_m y), (\beta e_1 - \bar{H}_m y)) \\
&= \|\beta e_1 - \bar{H}_m y\|_2^2 \; . \qquad (12)
\end{aligned}
$$

A result of the equality (12) is that we can minimize the M-norm of the (preconditioned) residual vector $M^{-1}(b-Ax)$ by simply minimizing the 2-norm of $\beta e_1 - \bar{H}_m y$ as in the standard GMRES algorithm.

ALGORITHM **2.5** *Left-preconditioned GMRES with M-inner products*

0. *Compute* $r_0 = b - Ax_0$ *and* $z = M^{-1}r_0$
1. *Compute* $\beta = (r_0, z)^{1/2}$; $v_1 = z/\beta$ *and* $w_1 = r_0/\beta$
2. *For* $j = 1, 2, \ldots, m$ *do:*
3. $\quad w = Av_j$
4. $\quad$ *For* $i = 1, \ldots, j$ *do:*
5. $\quad\quad h_{ij} = (w, v_i)$
6. $\quad\quad w = w - h_{ij} w_i$
7. $\quad$ *EndDo*
8. $\quad z = M^{-1}w$.
9. $\quad h_{j+1,j} = (z, w)^{1/2}$. *If* $h_{j+1,j} = 0$ *then Stop.*
10. $\quad w_{j+1} = w/h_{j+1,j}$
11. $\quad v_{j+1} = z/h_{j+1,j}$
12. *EndDo*
13. *Compute the minimizer* y_m *of* $\|\beta e_1 - \bar{H}_m y\|_2$
14. *Compute the approximate solution* $x_m = x_0 + V_m y_m$
15. *If satisfied Stop; else set* $x_0 = x_m$ *and goto 1.*

An equality similar to (12) can be shown for the right-preconditioned case with M^{-1}-inner products. We can summarize with the following theorem, which we state without proof.

Theorem 2.1 *The approximate solution* x_m *obtained from the left-preconditioned GMRES algorithm with* M*-inner products minimizes the residual* M*-norm* $\|M^{-1}(b-Ax)\|_M$ *over all vectors of the affine subspace* $x_0 + K_m$ *in which*

$$K_m = \text{span}\{z_0, M^{-1}Az_0, \ldots, (M^{-1}A)^{m-1}z_0\} \qquad (13)$$

where $z_0 = M^{-1}r_0$. *Also, the approximate solution* x_m *obtained from the right-preconditioned GMRES algorithm with* M^{-1}*-inner products minimizes the residual* M^{-1}*-norm* $\|b - Ax\|_{M^{-1}}$ *over the same affine subspace.*

2.4 Equivalence of the algorithms

We can show that both left and right symmetric preconditioning are mathematically equivalent to split preconditioning. In the latter case, M must be factored into $M = LL^T$ and we solve

$$L^{-1}AL^{-T}u = L^{-1}b, \qquad x = L^{-T}u. \tag{14}$$

Denoting by B the preconditioned matrix $B = L^{-1}AL^{-T}$, the GMRES procedure applied to the above system for the u variable, minimizes the residual norm,

$$\|L^{-1}(b - AL^{-T}u)\|_2$$

over all vectors u in the space $u_0 + K_m^{(u)}$ with

$$K_m^{(u)} = \text{span}\{\tilde{r}_0, B\tilde{r}_0, \ldots, B^{m-1}\tilde{r}_0\}$$

in which

$$\tilde{r}_0 = L^{-1}(b - AL^{-T}u_0) = L^{-1}(b - Ax_0) = L^{-1}r_0.$$

Note that the variables u and x are related by $x = L^{-T}u$. As a result, this procedure minimizes

$$\|L^{-1}(b - Ax)\|_2 \tag{15}$$

over all x in the space $x_0 + K_m^{(x)}$ with

$$K_m^{(x)} = \text{span}\{L^{-T}\tilde{r}_0, L^{-T}B\tilde{r}_0, \ldots, L^{-T}B^{m-1}\tilde{r}_0\}.$$

We now make the following observation. For any $k \geq 0$ we have,

$$L^{-T}B^k\tilde{r}_0 = L^{-T}B^kL^{-1}r_0 = (M^{-1}A)^k z_0$$

where $z_0 = M^{-1}r_0$. Indeed, this can be easily proved by induction. Hence, the space $K_m^{(x)}$ is identical with the space

$$K_m^{(x)} = \text{span}\{z_0, M^{-1}Az_0, \ldots, (M^{-1}A)^{m-1}z_0\}$$

which is nothing but (13). In noting that

$$\|L^{-1}(b - Ax)\|_2 = \|b - Ax\|_{M^{-1}} = \|M^{-1}(b - Ax)\|_M \tag{16}$$

we have proved the following result.

Theorem 2.2 *Let $M = LL^T$. Then the approximate solution obtained by GMRES applied to the split preconditioned system (14) is identical with that obtained from the GMRES algorithm for the left preconditioned system (2) using the M-inner product.*

Again, the same statement can be made about right-preconditioning. All that must be noticed is that the same minimization (16) is taking place, and that the minimization is over the same subspace in each of the left, right, and split preconditioning options [7, Sec. 9.3.4]. We emphasize in particular that it is the split preconditioned residual that is minimized in all three algorithms.

3 Truncated iterative methods

Truncated iterative methods are an alternative to restarting, when the number of steps required for convergence is large and the computation and storage of the Krylov basis becomes excessive. When A is exactly symmetric, a three-term recurrence governs the vectors in the Arnoldi process, and it is only necessary to orthogonalize the current Arnoldi vector against the previous two vectors. If A is nearly symmetric, an incomplete orthogonalization against a small number of previous vectors may be advantageous over restarted methods. The advantage here may offset the cost of maintaining the extra set of vectors to maintain the initial degree of symmetry. The incomplete Arnoldi procedure outlined below stores only the previous k Arnoldi vectors, and orthogonalizes the new vectors against them. It differs from the full Arnoldi procedure only in Line 4, which would normally be a loop from 1 to j. It can be considered to be the full Arnoldi procedure when k is set to infinity.

ALGORITHM **3.1** *Incomplete Arnoldi Procedure*

1. *Choose a vector* v_1 *of norm 1.*
2. *For* $j = 1, 2, \ldots, m$ *do*
3. $w = Av_j$
4.   *For* $i = \max\{1, j-k+1\}, \ldots, j$ *do*
5. $h_{ij} = (w, v_i)$
6. $w = w - h_{ij} v_i$
7.   *EndDo*
8. $h_{j+1,j} = \|z\|_2$
9.   *If* $h_{j+1,j} = 0$ *then Stop.*
10. $v_{j+1} = w / h_{j+1,j}$
11. *EndDo*

The truncated version of GMRES uses this incomplete Arnoldi procedure and is called Quasi-GMRES [3]. The practical implementation of this algorithm allows the solution to be updated at each iteration, and is thus called a 'direct' version, or DQGMRES [8].

To suggest that truncated iterative methods may be effective in cases of near symmetry, we study the asymptotic behavior of the iterates of DQGMRES as the coefficient matrix A varies from nonsymmetry to (skew) symmetry. We first decompose A as

$$A = S + B$$

in which S is symmetric or skew symmetric, and set

$$\varepsilon = \|B\|_2 \,.$$

We will first establish asymptotic relations among the variables in the incomplete and full Arnoldi procedures. Then we will apply the incomplete procedure to A, and the full procedure to S, using the superscripts I and F to distinguish between the variables appearing in the two procedures. (Note that since S is (skew) symmetric, the full procedure on S is the same as the incomplete procedure with $k \geq 2$.)

Moreover, if we denote the degree of the minimal polynomial of v_1^F with respect to S by ν, then $h^F_{\nu+1,\nu} = 0$ and $h^F_{j+1,j} \neq 0$ for $1 \leq j < \nu$. In the proof of the following lemma, we also use $\hat{v}^I_j$ and $\hat{v}^F_j$ to denote the vectors w^I and w^F obtained at the end of Line 7 in the incomplete and complete Arnoldi procedures.

Lemma 3.1 *Assume the truncation parameter $k \geq 2$. If $v_1^I = v_1^F + O(\varepsilon)$, then*

$$h^I_{ij} = h^F_{ij} + O(\varepsilon)\,, \qquad v^I_j = v^F_j + O(\varepsilon)$$

where $1 \leq j \leq \nu$ and $\max\{1, j-k+1\} \leq i \leq j+1$.

Proof. The proof is by induction on the index j. By Lines 5 and 6 of the Arnoldi procedure,

$$h^I_{11} = (Av^I_1, v^I_1)\,, \quad \hat{v}^I_2 = Av^I_1 - h^I_{11}v^I_1\,, \quad h^I_{21} = \|\hat{v}^I_2\|_2\,,$$

we have

$$h^I_{11} = (Sv^F_1, v^F_1) + O(\varepsilon) = h^F_{11} + O(\varepsilon)\,,$$

$$\hat{v}^I_2 = Sv^F_1 - h^F_{11}v^F_1 + O(\varepsilon) = \hat{v}^F_2 + O(\varepsilon)\,,$$

$$h^I_{21} = \|\hat{v}^F_2\|_2 + O(\varepsilon) = h^F_{21} + O(\varepsilon)$$

and hence the lemma holds for $j = 1$. Assume that the lemma has been proved for $j < j_0 \leq \nu$. On that hypothesis, we prove it for $j = j_0$. By Line 10 of the Arnoldi procedure,

$$v^I_{j_0} = \hat{v}^I_{j_0} / h^I_{j_0,j_0-1}$$

which yields that

$$v^I_{j_0} = \frac{\hat{v}^F_{j_0} + O(\varepsilon)}{h^F_{j_0,j_0-1} + O(\varepsilon)} = \frac{\hat{v}^F_{j_0}}{h^F_{j_0,j_0-1}} + O(\varepsilon) = v^F_{j_0} + O(\varepsilon)$$

by the induction hypothesis. Therefore

$$w^I = Av^I_{j_0} = Sv^F_{j_0} + O(\varepsilon) = w^F + O(\varepsilon)$$

for the w^I and w^F in Line 3 of the Arnoldi procedures. Using another induction on the index i in Lines 5 and 6, and the induction hypothesis on j and, in the mean time, noting that $h^F_{ij_0} = 0$ for $1 \leq i \leq j_0 - 2$, we have

$$h^I_{ij_0} = h^F_{ij_0} + O(\varepsilon)$$

for $\max\{1, j_0 - k + 1\} \leq i \leq j_0$ and

$$\hat{v}^I_{j_0+1} = \hat{v}^F_{j_0+1} + O(\varepsilon)\,.$$

From the last equation,

$$h^I_{j_0+1,j_0} = \|\hat{v}^I_{j_0+1}\|_2 = \|\hat{v}^F_{j_0+1}\|_2 + O(\varepsilon) = h^F_{j_0+1,j_0} + O(\varepsilon)$$

and then the induction step is complete. QED.

We now turn to the DQGMRES algorithm. Consider the linear system

$$Sx = b$$

and denote by x^G_m and x^Q_m the approximate solutions by the GMRES and DQGMRES algorithms, respectively. Let μ be the degree of the minimal polynomial of the vector $b - Sx_0$ with respect to S. A result of the lemma can be stated as follows.

Theorem 3.1 *Given the same initial guess x_0 to GMRES and DQGMRES with $k \geq 2$, then at any given step m with $1 \leq m \leq \mu$,*

$$x^Q_m = x^G_m + O(\varepsilon)\,.$$

Proof. By the definitions of DQGMRES and GMRES, we have

$$x^Q_m = x_0 + \beta^Q V^I_m \left(\left(\bar{H}^I_m\right)^T \bar{H}^I_m\right)^{-1} \left(\bar{H}^I_m\right)^T e_1$$

and

$$x^G_m = x_0 + \beta^G V^F_m \left(\left(\bar{H}^F_m\right)^T \bar{H}^F_m\right)^{-1} \left(\bar{H}^F_m\right)^T e_1$$

where $\beta^Q = \|b - Ax_0\|_2$ and $\beta^G = \|b - Sx_0\|_2$. Since

$$v^I_1 = \frac{1}{\beta^Q}(b - Ax_0) = \frac{1}{\beta^G}(b - Sx_0) + O(\varepsilon) = v^F_1 + O(\varepsilon)\,,$$

we have by the lemma,

$$\bar{H}^I_m = \bar{H}^F_m + O(\varepsilon)\,, \qquad V^I_m = V^F_m + O(\varepsilon)$$

and therefore the desired equation holds. QED.

If we let x_A be the exact solution to $Ax = b$ and x_S be the exact solution to $Sx = b$, then it is obvious that $x_A = x_S + O(\varepsilon)$. Since, on the other hand, $x^G_\mu = x_S$ we immediately have the following corollary.

Corollary 3.1 *For any initial guess x_0 and any $k \geq 2$,*

$$x^Q_\mu = x_A + O(\varepsilon)\,.$$

The corollary suggests that we may use DQGMRES with small k when A is nearly symmetric or nearly skew symmetric.

4 Symmetric preconditioning in Bi-CG

The Bi-CG algorithm is based on Lanczos biorthogonalization. Both left-symmetric and right-symmetric preconditioning are relatively straightforward, and no extra vectors are required. For reference, Algorithm 4.1 gives the right-preconditioned Bi-CG algorithm with preconditioner M. The symmetric right-preconditioned Bi-CG algorithm (right-preconditioned Bi-CG using M^{-1}-inner products) is developed immediately afterward.

ALGORITHM **4.1** *Right-preconditioned Bi-CG*

1. *Compute* $r_0 = b - Ax_0$. *Choose* r_0^* *such that* $(r_0, r_0^*) \neq 0$.
2. *Set* $p_0 = r_0$, $p_0^* = r_0^*$
3. *For* $j = 0, 1, \cdots$, *until convergence Do:*
4. $\quad \alpha_j = (r_j, r_j^*)/(AM^{-1}p_j, p_j^*)$
5. $\quad x_{j+1} = x_j + \alpha_j M^{-1} p_j$
6. $\quad r_{j+1} = r_j - \alpha_j AM^{-1} p_j$
7. $\quad r_{j+1}^* = r_j^* - \alpha_j M^{-T} A^T p_j^*$
8. $\quad \beta_j = (r_{j+1}, r_{j+1}^*)/(r_j, r_j^*)$
9. $\quad p_{j+1} = r_{j+1} + \beta_j p_j$
10. $\quad p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*$
11. *EndDo*

Note in Line 7 of the above algorithm that the preconditioned coefficient matrix of the dual system is $(AM^{-1})^T = M^{-T}A^T$, i.e., the dual residual r^* is the residual of

$$M^{-T}A^T x^* = b^*,$$

which is a *left*-preconditioned version of some linear system with A^T.

To develop the symmetric right-preconditioned Bi-CG, M^{-1}-inner products are used in Algorithm 4.1 above. However, the preconditioned coefficient matrix of the dual system must be the adjoint of AM^{-1} in the M^{-1} inner product. This is $A^T M^{-1}$ as shown by

$$(AM^{-1}x, y)_{M^{-1}} = (M^{-1}AM^{-1}x, y) = (x, M^{-1}A^T M^{-1} y) = (x, A^T M^{-1} y)_{M^{-1}}.$$

The dual system thus involves the coefficient matrix $A^T M^{-1}$. Algorithm 4.2 gives the symmetric right-preconditioned Bi-CG algorithm with preconditioner M.

ALGORITHM **4.2** *Right-preconditioned Bi-CG with M^{-1}-inner products*

1. *Compute* $r_0 = b - Ax_0$. *Choose* r_0^* *such that* $(M^{-1}r_0, r_0^*) \neq 0$.
2. *Set* $p_0 = M^{-1}r_0$, $p_0^* = M^{-1}r_0^*$
3. *For* $j = 0, 1, \cdots$, *until convergence Do:*
4. $\quad \alpha_j = (M^{-1}r_j, r_j^*)/(Ap_j, p_j^*)$
5. $\quad x_{j+1} = x_j + \alpha_j p_j$
6. $\quad r_{j+1} = r_j - \alpha_j A p_j$

7. $\quad r^*_{j+1} = r^*_j - \alpha_j A^T p^*_j$
8. $\quad \beta_j = (M^{-1}r_{j+1}, r^*_{j+1})/(M^{-1}r_j, r^*_j)$
9. $\quad p_{j+1} = M^{-1}r_{j+1} + \beta_j p_j$
10. $\quad p^*_{j+1} = M^{-1}r^*_{j+1} + \beta_j p^*_j$
11. *EndDo*

Like GMRES, both left and right symmetric preconditioned versions of Bi-CG are equivalent to the split preconditioned version, and this can be shown by a change of variables. However, in both left and right symmetric preconditioned versions, the exact, rather than the split preconditioned residual is available.

The unpreconditioned Bi-CG algorithm cannot have a *serious breakdown* if A is SPD and r^*_0 is chosen to be r_0. This is because $r^*_j = r_j$ and $p^*_j = p_j$ for all j and the vectors Ap_j, p^*_j never become orthogonal. In fact, the cosine

$$\frac{(Ap_j, p^*_j)}{\|Ap_j\|\|p^*_j\|} = \frac{(Ap_j, p^*_j)}{\|p_j\|\|p^*_j\|}\frac{\|p_j\|}{\|Ap_j\|}$$

can be bounded below by the reciprocal of the condition number of A.

Similarly, in the symmetric right-preconditioned version of Bi-CG, if both A and M are SPD, and $r^*_0 = r_0$, then $r^*_j = r_j$ and $p^*_j = p_j$ for all j, and

$$\begin{aligned} \frac{(Ap_j, p^*_j)}{\|Ap_j\|\|p^*_j\|} &\geq \mathrm{cond}^{-1}(A) \\ \frac{(M^{-1}r_j, r^*_j)}{\|M^{-1}r_j\|\|r^*_j\|} &\geq \mathrm{cond}^{-1}(M). \end{aligned}$$

We measure the cosines rather than the quantities (Ap_j, p^*_j) and $(M^{-1}r_j, r^*_j)$ because the p and r vectors have magnitudes going to 0 as the algorithms progress. Recall that in the case when $(M^{-1}r_j, r^*_j) = 0$ and $r_j = 0$, we have a *lucky breakdown*.

For the case of regular right- or left-preconditioning, or if $r^*_0 \neq r_0$ in the symmetrically preconditioned cases, then no such lower bounds as the above exist, and the algorithms are liable to break down.

When A is near-symmetric, it is our hypothesis that the probability of breakdown is lower in the symmetrically preconditioned cases, and this will be shown by experiment in the next section.

5 Numerical Experiments

Section 5.1 tests the idea of using symmetric preconditionings with truncated iterative methods. Section 5.2 tests the breakdown behavior of symmetrically preconditioned Bi-CG.

5.1 Truncated iterative methods

To test the idea of using symmetric preconditionings with truncated iterative methods for nearly symmetric systems, we selected a standard fluid flow problem where the degree of symmetry in the matrices is parameterized by the Reynolds number. The flow problem is the two-dimensional square-lid driven cavity, and was discretized by the Galerkin Finite Element method. Rectangular elements were used, with biquadratic basis functions for velocities, and linear discontinuous basis functions for pressure. We considered a segregated solution method for the Navier-Stokes equations, where the velocity and pressure variables are solved separately; the matrices arising from a fully-coupled solution method are otherwise indefinite. In particular, we considered the expression of the conservation of momentum,

$$Re(u \cdot \nabla u) = -\nabla p + \nabla^2 u$$

where u denotes the vector of velocity variables, p denotes the pressure variable, and Re is the Reynolds number. The boundary conditions for the driven cavity problem over the unit square are $u = (1,0)^T$ on the top edge of the square, and $u = (0,0)^T$ on the other three sides and the corners. The reference pressure specified at the bottom-left corner is 0.

The matrices are the initial Jacobians at each Newton iteration, assuming a zero pressure distribution. For convenience, however, we chose the right-hand sides of the linear systems to be the vector of all ones. A mesh of 20 by 20 elements was used, leading to momentum equation matrices of order 3042 and having 91204 nonzero entries. The nodes corresponding to the boundaries were not assembled into the matrix, and the degrees of freedom were numbered element by element. For Reynolds number 0., the matrix is SPD, and is equal to the symmetric part of the matrices with nonzero Reynolds number.

We generated matrices with Reynolds number less than 10, which gives rise to the nearly symmetric case. For Reynolds number 1., the degree of symmetry measured by

$$\frac{\|A - A^T\|_F}{\|A + A^T\|_F}$$

has value 7.5102×10^{-4} and this measure increases linearly with the Reynolds number (at least up to Re=10).

In the numerical experiments below, we show the number of matrix-vector products consumed by GMRES(k) and DQGMRES(k) to reduce the *actual* residual norm to less than 10^{-6} of the original residual norm, with a zero initial guess. Several values of k are used. A dagger (†) in the tables indicates that there was no convergence within 500 matrix-vector products. The incomplete Choleski factorization IC(0) of the Re=0 problem was used as the preconditioner in all the problems.

For comparison, we first show in Table 1, the results using standard right-preconditioning. Table 2 shows the results using right-preconditioning with M^{-1} inner products, or equivalently, split preconditioning. In the latter case, care was taken to assure that GMRES did not stop before the *actual* residual norm was within twice the tolerance. For DQGMRES, since an accurate residual norm estimate is not available within the algorithm, the exact residual norm was computed and used for the stopping criterion for the purpose of this comparison. The right-preconditioned methods have a slight advantage in this comparison (by as many as 20 Mat-Vec's), since they directly minimize the actual residual norm, whereas the symmetrically preconditioned methods minimize a preconditioned residual norm.

	GMRES(k)			DQGMRES(k)								
Re.	5	10	∞	2	3	4	5	6	7	8	9	10
0	232	129	59	76	220	105	72	92	†	160	83	75
1	218	126	69	76	276	131	81	94	†	97	90	79
2	233	126	70	78	391	258	82	95	†	98	94	78
3	208	126	71	87	346	232	85	95	†	99	97	79
4	214	128	72	94	345	†	88	95	477	108	98	86
5	210	128	72	192	394	†	91	95	326	128	99	90
6	214	128	73	446	361	†	94	97	258	197	100	94
7	215	129	73	†	345	†	97	99	239	229	101	96

Table 1: Mat-Vec's for convergence for right-preconditioned methods.

	GMRES(k)			DQGMRES(k)								
Re.	5	10	∞	2	3	4	5	6	7	8	9	10
0	243	119	57	58	58	58	58	58	58	58	58	58
1	243	119	67	75	74	74	75	74	74	74	75	75
2	244	120	68	78	78	78	79	78	78	78	78	78
3	244	121	69	88	87	87	87	87	86	86	87	87
4	244	122	70	108	95	95	95	93	91	91	93	95
5	244	126	70	†	105	103	105	100	97	96	101	104
6	244	127	71	†	118	111	119	108	101	101	110	117
7	243	128	71	†	131	121	139	117	104	105	121	139

Table 2: Mat-Vec's for convergence for symmetric right-preconditioned methods.

The results in Table 1 show the irregular performance of DQGMRES(k) for these small values of k when the preconditioned system is not symmetric. The performance is entirely regular in Table 2, where the preconditioned system is near symmetric. For Reynolds numbers up to 3, the systems are sufficiently symmetric so that DQGMRES(2) behaves the same as DQGMRES with much larger k. The performance remains regular until beyond Reynolds number 7, when the number of steps to convergence begins to become irregular, like in the right-preconditioned case.

GMRES with either right or symmetric preconditioning does not show any marked difference in performance; apparently the symmetry of the preconditioned system is not as essential here for this problem. However, the results do show that DQGMRES(k) with small values of k may perform as well, in terms of number of steps, as GMRES(k) with large values of k, particularly if near-symmetry is preserved. Since the former is much more efficient, the combination of preserving symmetry and truncated iterative methods may result in a much more economical method, as well as the more regular behavior shown above.

We also performed the same experiments with orthogonal projection methods, namely the Full Orthogonalization Method (FOM) and its truncated variant, the Direct Incomplete Orthogonalization Method (DIOM) [7]. The results were very similar to the results above, and are not shown here. Indeed, the development of the algorithms and the theory above is identical for these methods.

For interest, we also performed tests where an ILU(0) preconditioner was constructed for each matrix and compared right and split preconditioning. For the near-symmetric systems here, there was very little difference in these results compared to using IC(0) constructed from the Re=0 case for all the matrices. Thus the deterioration in performance as the Reynolds number increases is not entirely due to a relatively less accurate preconditioner, but is more due to the increased nonsymmetry and non-normality of the matrices. Although the eigenvalues of the preconditioned matrices are identical, their eigenvectors and hence their degree of non-normality may change completely. Unfortunately, it is difficult to quantitatively relate non-normality and convergence.

5.2 Breakdown behavior of Bi-CG

To test the breakdown behavior of Bi-CG, MATLAB was used to generate random matrices of order 300 with approximately 50 percent normally distributed nonzero entries. The matrices were adjusted so that

$$A \leftarrow \frac{A + A^T}{2} - (\sigma_{min} + 10^{-5})I + \varepsilon \frac{A - A^T}{2},$$

i.e., the symmetric part was shifted so that the lowest eigenvalue was 10^{-5} and then ε times the skew-symmetric part was added back. The parameter ε was altered to get varying degrees of nonsymmetry.

For each ε that we tested, 100 matrices were generated, and the smallest value of the cosines corresponding to the denominators in the algorithms were recorded. In the right-preconditioned case, we recorded the minimum of

$$\frac{(AM^{-1}p_j, p_j^*)}{\|AM^{-1}p_j\| \|p_j^*\|} \quad \text{and} \quad \frac{(r_j, r_j^*)}{\|r_j\| \|r_j^*\|}$$

for all j, and for the symmetric right-preconditioned case, we recorded the minimum of

$$\frac{(Ap_j, p_j^*)}{\|Ap_j\|\|p_j^*\|} \quad \text{and} \quad \frac{(M^{-1}r_j, r_j^*)}{\|M^{-1}r_j\|\|r_j^*\|}$$

for all j. The relative residual norm reduction was 10^{-9} when the iterations were stopped. The initial guesses were 0, and r_0^* was set to r_0. IC(0) of the symmetric part was used as the preconditioner.

Table 3 shows the frequencies of the size of minimum cosines for the right-preconditioned (first row of each pair of rows) and the symmetrically-preconditioned cases (second row of each pair of rows). For example, all 100 minimum cosines were between 10^{-3} and 3×10^{-3} in the symmetrically-preconditioned case. The average number of Bi-CG steps and the average minimum cosine is also shown. The last column, labeled 'better', shows the number of times that the minimum cosine was higher in the improved algorithm.

The Table shows that the right-preconditioned algorithm can produce much smaller cosines, indicating a greater probability for breakdown. The difference between the algorithms is less as the degree of nonsymmetry is increased. For $\varepsilon = 0.1$, there is almost no difference in the breakdown behavior of the algorithms. The Table shows that the number of Bi-CG steps is not significantly reduced in the new algorithm, nor is the *average* minimum cosine of the modified algorithm significantly increased. It is the probability that a small cosine is not encountered that is better.

It is important to note that this behavior only applies when r_0^* is set to r_0. When r_0^* is chosen randomly, there is no gain in the symmetrically-preconditioned algorithm, as shown in Table 4.

Table 5 shows the number of steps and the minimum cosines for the two algorithms applied to the driven cavity problem described in Section 5.1 above. Figure 1 shows a plot of the minimum cosines as the two algorithms progress for the $Re = 1$ problem. Note that the minimum cosines are higher and much smoother in the symmetrically-preconditioned case. In the $Re = 7$ problem, the cosines are still higher, but the smoothness is lost.

6 Conclusions

When solving linear systems with matrices that are very close to being symmetric, this paper has shown that it is possible to improve upon the standard practice of using a (nonsymmetric) preconditioner for that matrix along with a left- or right-preconditioned iterative method. The original degree of symmetry may be maintained by using a symmetric preconditioner and an alternate inner product (or split preconditioning, if appropriate). By combining this idea with truncated iterative methods, solution procedures that converge more quickly and require less storage are developed. The truncated methods also seem to become more robust with the truncation parameter k when near-symmetry is maintained. The Bi-CG algorithm also seems to be more robust with respect to serious breakdown when near-symmetry is maintained.

ε $\left(\frac{\|A-A^T\|_F}{\|A+A^T\|_F}\right)$	steps	3e-6 1e-5	1e-5 3e-5	3e-5 1e-4	1e-4 3e-4	3e-4 1e-3	1e-3 3e-3	3e-3 1e-2	1e-2 3e-2	3e-2 1e-1	average $\times 10^{-3}$	better
0.000	32.51	1	4	9	19	26	30	11			1.35	
(0.)	31.77						100				1.87	74
0.005	30.57		1	2	8	16	34	34	5		3.51	
(2.3e-3)	29.97					2	1	82	15		8.54	92
0.010	29.27	1	0	3	2	10	29	32	20	3	7.25	
(4.5e-3)	28.94				1	2	1	6	88	2	15.79	77
0.050	27.53		1	3	8	13	36	31	8		4.15	
(2.3e-2)	27.32			2	3	7	18	39	26	5	9.47	69
0.100	26.38		1	11	18	39	27	4			0.88	
(4.5e-2)	26.42		3	4	15	40	27	11			1.26	57

Table 3: Frequencies of minimum cosines for right-preconditioned (first row of each pair of rows) and symmetrically-preconditioned (second row of each pair of rows) Bi-CG.

ε	steps	3e-6 1e-5	1e-5 3e-5	3e-5 1e-4	1e-4 3e-4	3e-4 1e-3	1e-3 3e-3	3e-3 1e-2	1e-2 3e-2	3e-2 1e-1	average $\times 10^{-3}$	better
0.000	33.05	1	4	11	24	26	30	4			0.92	
	32.54	1	1	3	11	24	56	4			1.31	63

Table 4: Frequencies of minimum cosines when r_0^* is chosen randomly.

	Bi-CG steps		min cosines	
Re.	right	symm	right	symm
0	70	62	1.52e-4	1.45e-1
1	71	68	1.08e-4	6.73e-3
2	74	72	2.44e-4	5.12e-4
3	73	72	2.02e-4	9.07e-3
4	77	72	1.93e-5	6.52e-3
5	80	75	5.54e-5	5.19e-4
6	80	78	1.91e-4	4.30e-5
7	80	80	1.87e-4	1.02e-3

Table 5: Steps and minimum cosines for the driven cavity problem.

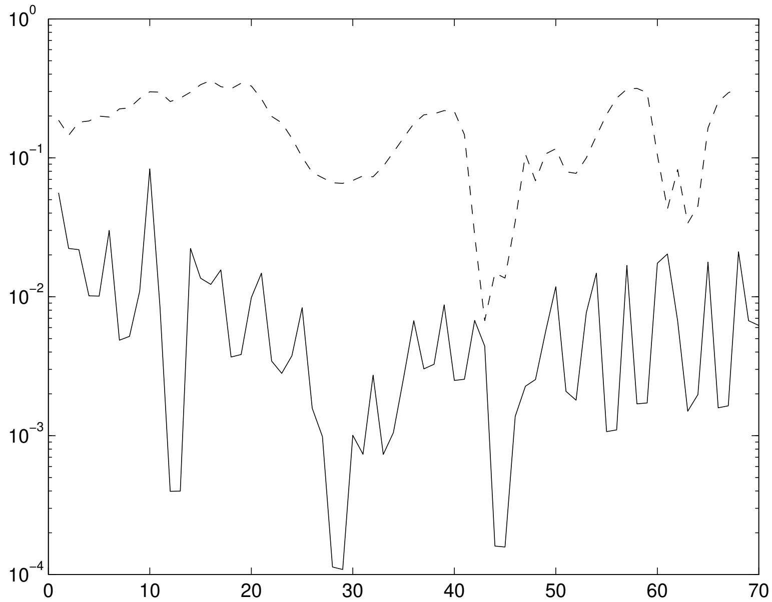

Figure 1: Minimum cosines in right-preconditioned Bi-CG (solid line) and symmetrically-preconditioned Bi-CG (dashed line) for the $Re = 1$ problem.

Acknowledgments

The first and third authors wish to acknowledge the support of RIACS, NASA Ames under Contract NAS 2-13721, where this collaboration originated. The second and third authors also wish to acknowledge the support of the Minnesota Supercomputer Institute.

References

[1] S. F. Ashby, T. A. Manteuffel, and P. E. Saylor. A taxonomy for conjugate gradient methods. *SIAM J. Numer. Anal.*, 27:1542–1568, 1990.

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

[3] P. N. Brown and A. C. Hindmarsh. Matrix-free methods for stiff systems of ODEs. *SIAM J. Numer. Anal.*, 23:610–638, 1986.

[4] J. A. Meijerink and H. A. Van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31(137):148–162, 1977.

[5] A. Meyer. The concept of special inner products for deriving new conjugate gradient-like solvers for non-symmetric sparse linear systems. *Num. Lin. Alg. Appl.*, 1, 1994.

[6] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Stat. Comput.*, 14:461–469, 1993.

[7] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing, New York, 1995.

[8] Y. Saad and K. Wu. DQGMRES: a direct quasi-minimal residual algorithm based on incomplete orthogonalization. *Num. Lin. Alg. Appl.*, to appear.

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