SIAM J. SCI. COMPUT.
Vol. 16, No. 4, pp. 917–933, July 1995

009

AN ITERATIVE METHOD FOR NONSYMMETRIC SYSTEMS WITH MULTIPLE RIGHT-HAND SIDES*

V. SIMONCINI† AND E. GALLOPOULOS‡

Abstract. We propose a method for the solution of linear systems $AX = B$ where A is a large, possibly sparse, nonsymmetric matrix of order n, and B is an arbitrary rectangular matrix of order $n \times s$ with s of moderate size. The method uses a single Krylov subspace per step as a generator of approximations, a projection process, and a Richardson acceleration technique. It thus combines the advantages of recent hybrid methods with those for solving symmetric systems with multiple right-hand sides. Numerical experiments indicate that in several cases the method has better practical performance and significantly lower memory requirements than block versions of nonsymmetric solvers and other proposed methods for the solution of systems with multiple right-hand sides.

Key words. nonsymmetric systems, iterative method, Krylov subspace, multiple right-hand sides, GMRES, hybrid method, Richardson

AMS subject classifications. 65F10, 65Y20

1. Introduction.

We consider techniques for the solution of linear systems

$$AX = B, \tag{1}$$

where A is a real nonsymmetric matrix of order n, and $B = [b_1, \ldots, b_s]$ is an arbitrary rectangular matrix of order $n \times s$ with s of moderate size. We will assume that the order of A is sufficiently large that it is possible to keep only a limited number of dimension-n vectors in memory without severe penalties due to heavy memory traffic. If one were able to solve 1 using direct methods, the first step would be to factorize A, and then solve for each right-hand side. The factorization step is more expensive than each solution (e.g., by a factor of n for dense A) but needs to be performed only once. Thus it can be regarded as an "information sharing" step across the systems. Our interest is in iterative techniques that allow the sharing of information during the solution of (1); the purpose is to solve these multiple nonsymmetric systems at a rate per system that is faster than solving each one separately. We would also like these methods to be applicable when all the right-hand sides b_j are not available simultaneously. It is worth noting that Lanczos in [14, §§4 and 8] devoted several paragraphs to (1), commenting on the merits of reusing basis vectors obtained from the iterative solution for one right-hand side in order to solve for the remaining ones, and comparing with the direct computation of the inverse as well as with repeated application of an iterative method.

When matrix A is symmetric, block versions of the conjugate gradient (CG) and minimum residual algorithms of O'Leary [19] as well as the block Davidson method by Sadkane and Vital [28] are appropriate for handling the problem (1), whereas the Lanczos schemes proposed by Parlett [22] and extended by Saad [25] and Papadrakakis and Smerou in [21], and the method of van der Vorst [40] are mostly suitable when not all b_j's are simultaneously available. Reference [21] is of particular interest as it contains numerical experiments from actual applications. See also [19] for useful references. The literature for nonsymmetric systems with multiple right-hand sides is less well developed. Two methods that have been proposed are block

*Received by the editors July 13, 1992, accepted for publication (in revised form) June 6, 1994.

†Istituto per lo Studio delle Metodologie Geofisiche Ambientali, Consiglio Nazionale delle Ricerche, Modena, Italy (valeria@bora.bo.cnr.it). This work was accomplished while the author was visiting the Center for Supercomputing Research and Development of the University of Illinois at Urbana-Champaign, supported by fellowship 2043597 from the Consiglio Nazionale delle Ricerche, Italy.

‡Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801-2307 (stratis@csrd.uiuc.edu). This research was supported by National Science Foundation grant NSF CCR-91-20105.

generalizations of solvers for nonsymmetric systems: the block biconjugate gradient algorithm (BBiCG) of O'Leary [19], and block GMRES (BGMRES) described by Vital [41]. We also note the recent communication by Kharchenko et al. on the use of restarted BGMRES for dense complex systems [13].

An alternative idea to block methods, suggested in the context of radar scattering applications and applied mostly to complex symmetric matrices, is to use a single "seed" system and some CG-type method as a generator of approximations for several right-hand sides. Smith [34] used a single seed system to solve multiple systems in electromagnetic scattering. Smith, Peterson, and Mittra also considered these methods [35] and Joly [9] extended the discussion. The underlying CG-like schemes are CG on normal equations, biconjugate gradients (BiCG) [9], [34], and a preconditioned generalized conjugate residual method [9]. If A is such that normal equations or BiCG approaches are successful, then the methods of [9] and [35] can be effective. We present one such method in §2.3.

In this paper we propose an alternative scheme for the solution of (1) and compare its performance with existing schemes. The method uses one subspace as a generator of approximations and accomplishes information sharing by means of a projection process and Richardson acceleration, thus utilizing some advantages of the hybrid method of Nachtigal, Reichel, and Trefethen [18]. Compared to BBiCG this method is transpose-free while compared to BGMRES, it needs substantially less memory, and it gives better overall performance. Furthermore, it does not exhibit the problems encountered when normal equations are used. In principle, the method could also be used when all the right-hand sides are not immediately available. We note that the method exploits the presence of common spectral information between the right-hand sides, and will be less effective whenever little useful common information can be found; this behavior is summarized in Theorem 3.2. Nevertheless, experiments in §5 indicate that the method is quite robust even under fairly adverse composition of the right-hand sides. In this study we do not address preconditioning; however, it is fair to say that it is an important issue and deserves a separate study. In particular, when s is large, expensive but effective forms of preconditioning may be justified because their cost is amortized over a large number of right-hand sides. We also point to [39] for preconditioners applied to methods having a polynomial component.

The structure of the paper is as follows. Section 2 reviews existing algorithms. Section 3 describes the new method, its implementation, and its convergence properties. The complexity of the methods is discussed in §4. Finally, numerical experiments and conclusions are presented in §§5 and 6.

2. Some existing methods. One approach for solving (1) is to apply a standard iterative method to each system. This is the simplest approach, but it makes no use of any information gathered during the solution of each system. Here we are interested in alternative methods which, by means of information sharing, accomplish convergence in fewer iterations than would have been the case if each system were to be solved separately. To justify our approach, we briefly describe some existing methods that can be used to solve (1), exploiting the fact that there are many right-hand sides to be solved for a single matrix.

2.1. Block BiCG. The BBiCG algorithm, proposed and described in [19], is a block generalization of BiCG [4]. In exact arithmetic, if BBiCG does not terminate before $\bar{k} = \lceil n/s \rceil$ steps, then the solution becomes available at step $\bar{k}$ [19, Thm. 1]. Even though the underlying BiCG method can be effective in several cases [10], [16], [17], [37], there are some drawbacks that seem to be inherited by the block version. These are erratic convergence behavior, possible division by zero, and sensitivity to the choice of the auxiliary starting vector. An organizational drawback is the use of the transpose of A. Recent research has been attempting to design methods which overcome some of these drawbacks [5], [10]. One problem due to the block

TABLE 1
Leading computational costs per iteration of BBiCG.

$M \times V$	$2s$	Mult. of blocks of dim. $n \times s$ and $s \times s$	5
n-vector DAXPY	$5s$	n-vector DOT (products)	$2s^2 + s$
Factor order s matrices	3	Mult. two order s matrices	4
Solve order s triang. system	$8s$	MGS on $n \times s$ block	2

TABLE 2
Leading computational costs per iteration of BGMRES *to restart.*

$M \times V$	$s(m+1)$	Mult. of blocks of dim. $n \times s$ and $s \times s$	$\frac{m(m+1)}{2}$
n-vector DAXPY	$s\frac{m(m+1)}{2} + s$	n-vector DOT (products)	$s^2\frac{m(m+1)}{2} + s$
Solve order sm triang. system	s	MGS on $n \times s$ block	$m-1$
Leading scalar costs	$6s^3m^2 + s^2m^3$		

nature of BBiCG is the possible rank deficiency or near rank deficiency of the search direction blocks. One suggested remedy is to restart BBiCG with deflated right-hand sides. Another remedy proposed to mitigate when possible the propagation of roundoff is to orthogonalize the direction matrices at each iteration [19].

The major computational costs during BBiCG are as shown in Table 1. We denote by $M \times V$ the cost for performing one matrix-vector multiplication with the (sparse) matrices A or A^T. Function MGS orthogonalizes a set of columns using modified Gram–Schmidt.

2.2. Block GMRES. The BGMRES algorithm was described in [41] and is repeated in the Appendix. Central to each iteration of the algorithm is a block Arnoldi process, which produces an orthogonal basis for $K_m(A, R) := \{R, AR, \ldots, A^{m-1}R\}$, where R is an $n \times s$ residual block (see also [24]), followed by a (block) minimization problem defined on that subspace. In exact arithmetic and under certain conditions on R and A, BGMRES achieves finite termination in $\lceil n/s \rceil$ iterations. It follows that, as for BBiCG, the number of iterations to termination for BGMRES is expected to decrease as the number of right-hand sides increases [33]. This property, combined with the built-in minimization of the block residual, make BGMRES mathematically attractive. There are, however, practical disadvantages such as high memory requirements; see [30] for comments on memory difficulties with block methods. It is also worth noting that similar memory difficulties arise in GMRES-like methods for the Sylvester equation [8][1]. The memory problem is handled by applying restarts. We denote by GMRES(m) and BGMRES(m, s) the restarted standard and block GMRES algorithms. Restarting entails, however, loss of the properties of finite termination and minimization over the entire Krylov subspace [27]. Experiments reported in [31] and §5 indicate that restarting has a deleterious effect on the purported nice convergence properties of BGMRES. In particular, for small m and large n, the number of right-hand sides s has to be large for an acceptable reduction of the number of iterations of BGMRES(m, s); increasing m seems to be a more effective way to reduce the number of iterations. Overall, the computational efficiency and numerical behavior of BGMRES(m, s) are strongly influenced by the relation between m and s; however, a full analysis remains to be done. The computational costs for one step of BGMRES(m, s) are shown in Table 2. It can be seen that in addition to the (order-n) matrix by vector multiplications, there is significant overhead arising from block Arnoldi and modified Gram–Schmidt, which are used to obtain a block basis $[U_1, \ldots, U_m]$ for $K_m(A, R)$; for a detailed complexity analysis, see [41].

[1] We thank one of the referees for pointing to the relevance of this reference.

2.3. Other methods. The electromagnetics literature describes alternatives to the block approach that share common features with our method [9], [35]. The success of these methods is usually measured by their effectiveness for complex symmetric matrices encountered in scattering. The method we describe is based on BiCG and was recently proposed by Joly [9]. We refer to the method as MJ3 [2]. The full algorithm is presented in the Appendix.

At each step of the algorithm, each current residual r_j for $j = 1, \ldots, s$ is projected onto the subspace generated by the direction vectors of a seed system, chosen among the systems that have not yet converged. The seed system is solved completely, while the remaining systems are suitably updated. Once converged, a new seed is chosen, namely the system with the maximum residual norm. Joly [9] shows that after starting with a new seed system, several orthogonality properties between old and new direction vectors are maintained, which permits information obtained during the solution of one system to be used for the next one; in exact arithmetic, this leads to finite termination in n steps, irrespective of s. The low cost per iteration of MJ3 ($2\ M \times V$, $3s + 2$ n-vector DAXPY and $3s + 2$ n-vector DOT products) makes the method attractive when the nonseed systems have significant components along the direction vectors generated by the seed system; otherwise, as indicated in §5, the total cost of the method becomes high. Furthermore, the disadvantages of BiCG (use of the transpose and erratic behavior) are still present.

3. MHGMRES: A hybrid algorithm for multiple right-hand sides. The aforementioned considerations together with recent advances in hybrid methods [18], [29], [36] led us to investigate alternatives for economical information sharing to solve (1). Some procedures for symmetric positive definite systems were described in [25], where the first system is solved using Lanczos iteration, the corresponding Krylov subspace is generated, and an approximation to the next system is computed using projection onto the previous Krylov subspace. Such an approach works well as long as the new right-hand side is near the subspace. In case this approximation is not satisfactory, a Lanczos run for the new system can be applied.

Our algorithm, referred to as MHGMRES, uses an Arnoldi process to generate a Krylov subspace for one of the residuals and then projects the remaining residuals onto that subspace. Unfortunately, except for special cases, there is no guarantee a priori for the quality of the approximation. Thus we consider additional methods to accomplish sharing of information among the systems. In particular, we use a hybrid scheme based on the method of Nachtigal, Reichel, and Trefethen [18], which allows us to exploit the generated GMRES residual polynomial in order to improve the approximation of all the systems.

The method described in [18] starts with an initial guess, runs GMRES until the residual has become small enough, constructs the GMRES residual polynomial, and applies it on the residual cyclically until convergence. As shown in [18], using the GMRES residual polynomial for the Richardson step for nonnormal matrices is more effective than using a polynomial built to be small in a domain determined by eigenvalue estimates.

The desire to have an effective method for multiple right-hand sides led us to modify the method of [18] to use restarting instead of either repeatedly applying the Richardson steps or continuing the GMRES process where it left off. As we make clear in the next section, by restarting, the method builds the solutions by appropriately using any information that can be shared between the right-hand sides. We will be using the name HGMRES whenever referring to the modified version of the algorithm used for a single right-hand side.

3.1. Description of the algorithm. We denote by e_j the jth element of the canonical basis and by $\|\cdot\|$ the 2-norm for vectors and matrices. Given a starting approximation to the solution $X^{(0)}$, the algorithm proceeds as follows (see Table 3). The initialization (lines 1–3)

[2] Postfix 3 acknowledges that this is the third of the algorithms proposed in [9].

TABLE 3
Algorithm MHGMRES.

Algorithm $X = \text{MHGMRES}(A, B, X^{(0)}, m, \epsilon)$
1. $X = X^{(0)}$
2. $R = B - AX$
3. $[\sigma, r_\sigma] = \text{SEED}(R)$
4. $\beta = \|r_\sigma\|$
5. while $\beta > \epsilon$
6. $\quad [V_{m+1}, H] = \text{ARNOLDI}(A, r_\sigma)$
7. $\quad \hat{b}_j = V_{m+1}^T(b_j - Ax_j), \quad j = 1, \ldots, s, \ j \neq \sigma$
8. $\quad \hat{b}_\sigma = \beta e_1$
9. $\quad$ compute y_j minimizing $\|Hy - \hat{b}_j\|$, $y \in \mathbb{R}^m$, $j = 1, \ldots, s$
10. $\quad X = X + V_m Y$, where $Y = [y_1, \ldots, y_s]$
11. $\quad \theta = \text{LEJA}(\text{QZ}(H))$
12. $\quad [x_j, r_j] = \text{RICHARDSON}(A, b_j, x_j, \theta) \quad j = 1, \ldots, s$
13. $\quad [\sigma, r_\sigma] = \text{SEED}(R)$
14. $\quad \beta = \|r_\sigma\|$
15. end

consists of the computation of initial residuals and convergence indicators and selection of a seed system with coefficient matrix A, in a manner to be specified later. Denoting the current approximate solution by $X = [x_1, \ldots, x_s]$, each iteration of the algorithm consists of four phases. In the first phase (line 6), an Arnoldi process is applied to build a basis V_{m+1} for the Krylov subspace $K_{m+1}(A, r_\sigma)$, where $r_\sigma = b_\sigma - Ax_\sigma$. The $(m + 1) \times m$ upper Hessenberg matrix H corresponding to the first m vectors of V_{m+1} is also generated.

In the second phase (lines 7–10), the solutions of all systems are approximated. This is done in the usual GMRES fashion for the seed system, while the nonseed solutions are approximated by projecting the residuals $r_j = b_j - Ax_j$ on $K_{m+1}(A, r_\sigma)$, and solving the least squares problems $\min_{y_j \in \mathbb{R}^m} \|Hy_j - V_{m+1}^T r_j\|$. This is done incrementally; that is, the required QR decomposition of H is carried out step by step for the seed system. Hence, for each additional right-hand side only multiplication by an orthogonal matrix and solution of an order-m triangular system needs to be done. In the third, Richardson phase (lines 11–12), parameters for the minimal residual polynomial corresponding to the first phase are generated, and the polynomial is applied on $R = [r_1, \ldots, r_s]$. In the fourth phase (lines 13–14), a new seed system is constructed from R, convergence information is obtained, and the process is restarted from the first phase for all those systems that have not yet converged. Since the first two phases of the algorithm correspond to a GMRES procedure applied to the seed and the projections of the other residuals, we refer to them collectively as the GMRES phase.

Function SEED is used to obtain the seed system. In our implementation, SEED applied on the $n \times s$ block R returns σ and r_σ, where σ is the index of the column of R having the maximum norm; this strategy is motivated in §3.3. Function ARNOLDI applies the Arnoldi procedure, using modified Gram–Schmidt re-orthogonalization, to build an orthogonal basis $V_{m+1} = [v_1, \ldots, v_{m+1}]$ for the Krylov subspace $K_{m+1}(A, r)$. Function QZ solves the generalized eigenvalue problem $H^T H z = \lambda \tilde{H}^T z$, with $\tilde{H} = [I_m, 0]H$. The values $\{\lambda_1, \ldots, \lambda_m\}$ are the roots of the GMRES polynomial. The choice of this approach for computing the roots is motivated by its good stability properties; see the discussion in [7]. Note that the procedure can be simplified by using the QR decomposition of H, which is already available. If $\mathcal{Q}$ and $\mathcal{R}$ are the factors of the decomposition at step m, values $\{\lambda_1, \ldots, \lambda_m\}$ are computed by solving $\mathcal{R}z = \lambda \tilde{\mathcal{Q}}^T z$, with $\tilde{\mathcal{Q}}$ the principal $m \times m$ submatrix of $\mathcal{Q}$. We observe that there are cheaper methods for deriving the λ_i's; one is to generate a triangular matrix C_m such

TABLE 4
Leading computational costs per iteration of MHGMRES *to restart.*

$M \times V$	$s(m+1)+m$	Mult. of vector by matrix of dim. $m \times (m+1)$	$s-1$
n-vector DAXPY	$3sm$	n-vector DOT (products)	$sm+2s-m-1$
Factor dim. $(m+1) \times m$ matrix	1	MGS on $n \times (m+1)$ block	1
Solve order m triang. system	s		
Other costs	*solve a generalized eigenvalue problem of order m*		

that $V_m = [v_1, Av_1, A^2v_1, \ldots, A^{m-1}v_1]C_m$, compute the coefficients $C_m y$ of the residual polynomial, and then use a rootfinder [18]. This approach suffers from the potential ill-conditioning of C_m, even though in some cases it can be cheaper and more effective than the QZ-based approach; see [32] for experiments. The reciprocals of the zeros of the minimal residual polynomial are used as coefficients of the Richardson acceleration procedure. We use Leja ordering, which is accomplished by means of function LEJA and is known to have several advantages [18], [23]. Reordered parameters are stored in vector θ and complex values are avoided by pairing complex roots and their conjugates [20]. Function RICHARDSON applies the GMRES polynomial to each residual r_j, $j = 1, \ldots, s$ using the following scheme:

$R = B - AX$
for $i = 1, m$
  $X = X + \lambda_i^{-1} R$
  $R = B - AX$
end

It is worth noticing that QZ is called once for each restart. This leads to considerable savings compared to the simple HGMRES method, where s polynomials must be considered per restart, necessitating s calls to the QZ procedure.

Table 4 shows the major computational costs associated with one iteration of MHGMRES to restart. The memory requirements are shown in Table 5. The algorithm is memory-conserving, in the sense that only one Krylov subspace is generated at each step. Furthermore, no multiplications with the transpose of A are required. During the algorithm, sharing of information is accomplished by: (a) the projection of all the residuals on the same subspace; (b) the solution of the least squares problems, using the common QR decomposition; (c) the application of the GMRES polynomial to all residuals. Experiments indicate that the application of the seed GMRES polynomial to all systems in each step is important to the success of the algorithm. This is related to the fact that the GMRES polynomial is small in a region that might include eigenvalues corresponding to eigendirections present in nonseed systems. The structure of the algorithm permits us to deal with the case when the right-hand sides are not all available at the same time. For example, new residuals can be introduced before the first phase, before step RICHARDSON, or before selecting the new seed. The method works, although experiments showed the need for a different seed selection strategy. This is because adding a new residual with the current approach delays the convergence of the current set of residuals and makes the overall solution time slower than if the systems were to be solved in strict sequence.

We also note a suggestion made in [18, §7] for handling multiple right-hand sides: to apply the GMRES component of the algorithm to all the residuals and then to select a polynomial for the hybrid method based on the cumulative information. However, this involves the application of the costly GMRES process s times. It is also possible to make MHGMRES more flexible, e.g., by varying m, or the number of times the Richardson phase is applied, or monitoring the application of the polynomial on nonseed residuals and stopping when the behavior is

unsatisfactory. In light of results in [11], [12], and [18] such flexibility is expected to enhance the performance of the algorithm; however, a full investigation remains to be done.

3.2. Error estimates for MHGMRES. We seek bounds for the residual norms at each iteration of the method to restart, both before and after the application of the Richardson phase. During every iteration, the seed system is approximated essentially by GMRES, for which error bounds are known [17]. We first bound the residual norms, in terms of information we have from the seed. Let Π_m be the space of polynomials of degree less than or equal to m. For a set S, and polynomial p on S, we define $\|p\|_S$ as $\sup_{z\in S}|p(z)|$. Since we deal with possibly nonnormal matrices, we use the notion of ϵ-pseudospectrum [38].

To simplify the notation, let $\sigma = 1$ be the index of the seed system. At each restart i, let $r_j^{(i)}$ denote the residual of the jth system, and $\tilde{r}_j^{(i+1)}$ the intermediate residual obtained after the GMRES phase. In the following, we denote by P_m the projection operator onto the Krylov subspace $K_m(A, r_1^{(i)}) = \{v_1, Av_1, \ldots, A^{m-1}v_1\}$ with $v_1 = r_1^{(i)}/\|r_1^{(i)}\|$. K_m denotes $K_m(A, r_1^{(i)})$. The following lemma will be needed.

LEMMA 3.1. *Let $\mathcal{L}$ be a subspace of $\mathbb{R}^n$ of dimension m, and let $v \in \mathbb{R}^n$ with $v \notin \mathcal{L}$. Given any $x \in v + \mathcal{L}$, denote by Fx the element of $\mathcal{L}$ for which $\|x - Fx\|$ is minimized. Then the error vector $x - Fx$ is independent of x.*

Proof. Let Pv be the projection of v in $\mathcal{L}$, and let $x = v + \tilde{x}$, with $\tilde{x} \in \mathcal{L}$. Then $\|x - Fx\| = \|v + \tilde{x} - Fx\|$, and $\|x - Fx\|$ is minimized by choosing Fx so that $\tilde{x} - Fx = -Pv$. Because the projection is unique, vector $x - Fx$ is unique and independent of x. □

Our use of this lemma will be for $\mathcal{L} := AK_m$ and $v := v_1$. Then we can write $x = v_1 + As(A)v_1$ and $Fx = Aq(A)v_1$ with $s, q \in \Pi_{m-1}$. Hence $x - Fx = p(A)v_1$ with $p \in \Pi_m$ and $p(0) = 1$. Lemma 3.1 tells us that polynomial p is independent of x.

For each $j = 1, \ldots, s$, the approximate solution after GMRES at step $i + 1$ is given by $\tilde{x}_j^{(i+1)} = x_j^{(i)} + \tilde{z}_j$, with $\tilde{z}_j \in K_m$. Rewriting the residual $\tilde{r}_j^{(i+1)} = b_j - A\tilde{x}_j^{(i+1)}$ as

$$\tilde{r}_j^{(i+1)} = r_j^{(i)} - A\tilde{z}_j,$$

the GMRES phase of the algorithm consists in determining vectors $\tilde{z}_j$'s for each $j = 1, \ldots, s$, such that

$$\tilde{z}_j = \arg\Big(\min_{z_j \in K_m} \|P_{m+1}r_j^{(i)} - Az_j\|\Big) \tag{2}$$

where $P_{m+1}r_1^{(i)} = r_1^{(i)}$.

THEOREM 3.2. *For each $j = 2, \ldots, s$, let $P_{m+1}r_j^{(i)} = q^{(j)}(A)v_1$ be the projection of residual $r_j^{(i)}$ on K_{m+1}, with $q^{(j)} \in \Pi_m$, $q^{(j)}(x) = \sum_{k=0}^{m}\alpha_{j,k}x^k$. For any $\epsilon \geq 0$, let $\Lambda_\epsilon = \{\xi \in \mathbb{C} : \|(\xi I - A)^{-1}\| \geq \frac{1}{\epsilon}\}$. Then, given any $\epsilon > 0$, there is some L_ϵ depending on Λ_ϵ such that the residuals $\tilde{r}_j^{(i+1)}$ satisfy*

$$\|\tilde{r}_j^{(i+1)}\| \leq \|(I - P_{m+1})r_j^{(i)}\| + |\alpha_{j,0}|\frac{L_\epsilon}{2\pi\epsilon}\delta_{m,\epsilon}, \quad j = 2, \ldots, s, \tag{3}$$

where

$$\delta_{m,\epsilon} = \min_{\substack{p\in\Pi_m \\ p(0)=1}} \|p\|_{\Lambda_\epsilon}.$$

Proof. For each j, from $\tilde{r}_j^{(i+1)} = r_j^{(i)} - A\tilde{z}_j$ and the triangle inequality we obtain

$$\|\tilde{r}_j^{(i+1)}\| \leq \|(I - P_{m+1})r_j^{(i)}\| + \|(P_{m+1}r_j^{(i)} - A\tilde{z}_j)\|. \tag{4}$$

For $z_j \in K_m$, $z_j = s^{(j)}(A)v_1$, and then we obtain

$$\min_{z_j \in K_m} \|P_{m+1} r_j^{(i)} - Az_j\| = \min_{s^{(j)} \in \Pi_{m-1}} \|q^{(j)}(A)v_1 - As^{(j)}(A)v_1\|$$
$$= |\alpha_{j,0}| \min_{\substack{p^{(j)} \in \Pi_m \\ p^{(j)}(0)=1}} \|p^{(j)}(A)v_1\|.$$

From the uniqueness of the minimum residual polynomial (refer to Lemma 3.1) and using the inequality

$$\min_{\substack{p \in \Pi_m \\ p(0)=1}} \|p(A)v_1\| \le \min_{\substack{p \in \Pi_m \\ p(0)=1}} \|p(A)\|,$$

we can write

$$\min_{z_j \in K_m} \|P_{m+1} r_j^{(i)} - Az_j\| \le |\alpha_{j,0}| \min_{\substack{p \in \Pi_m \\ p(0)=1}} \|p(A)\|. \tag{5}$$

It then follows from the integral representation of matrix functions that we have

$$p(A) = \frac{1}{2\pi\imath} \int_{\Gamma_\epsilon} p(\xi)(\xi I - A)^{-1} d\xi,$$

where Γ_ϵ is one or a union of positively oriented Jordan curves enclosing Λ_ϵ. Hence

$$\|p(A)\| \le \frac{L_\epsilon}{2\pi\epsilon} \|p\|_{\Lambda_\epsilon},$$

where L_ϵ is the length of Γ_ϵ. From (4) and (5) the result follows. □

Note that for $j = 1$, the first term of the right-hand side of (3) disappears. Furthermore, the Λ_ϵ coincides with the ϵ-pseudospectrum. Theorem 3.2 is a generalization of Theorem 3.1 of [25] and shows that the magnitude of the residual depends on the distance of the remaining residuals from the space spanned by the seed as well as on a polynomial approximation problem defined on Λ_ϵ. The next corollary follows trivially from Theorem 3.2.

COROLLARY 3.3. *The residual after the Richardson phase will satisfy*

$$\|r_j^{(i+1)}\| \le \|p(A)\| \left[\|(I - P_{m+1}) r_j^{(i)}\| + |\alpha_{j,0}| \frac{L_\epsilon}{2\pi\epsilon} \delta_{m,\epsilon} \right], \quad j = 2, \ldots, s.$$

Note that Richardson acceleration is most advantageous when $\|p(A)\| \approx \|\tilde{r}_1^{(i+1)}\| / \|r_1^{(i)}\|$ is of order $O(\delta_{m,\epsilon})$ [18]. If the polynomial constructed by the GMRES(m) step is such that $\|p(A)\| < 1$, the residuals $\|\tilde{r}_j^{(i+1)}\|$ obtained in (3) will decrease after the Richardson phase. This is guaranteed irrespective of the value of m when A has definite symmetric part [3]. Even then, however, $\|p(A)\|$ might be very near 1 unless m is taken very large, thus making the method impractical. Then other methods might be preferable; see [12], [17], [27].

Theorem 3.2 and its corollary show how the approximation is governed by the effectiveness of the projections and the underlying GMRES procedure, compounded by the application of the GMRES polynomial. We also note that solving (2) for $\tilde{z}_j$'s is straighforward. An interesting but less stable approach for generating the $\tilde{z}_j$'s, and which does not require the solution of $s - 1$ different least squares problems, was described in [32].

3.3. Seed selection heuristics. We next motivate the seed selection. Using $P_{m+1} = P_{m+1}^{(\sigma)}$ to denote the orthogonal projection onto the Krylov subspace of the seed, Theorem 3.2 shows how the seed affects the error. Then, for each $j \neq \sigma$, $\|(I - P_{m+1}) r_j\|$ is the distance of the

TABLE 5
Memory complexity for each method applied to (1).

MHGMRES(m, s)	$n(m+1+s)+O(m^2+sm)$	HGMRES(m)	$n(m+2)+O(m^2)$
BGMRES(m, s)	$ns(m+2)+\frac{1}{2}s^2m^2+O(s^2m)$	GMRES(m)	$n(m+2)+O(m^2)$
BBiCG	$5ns+O(s^2)$	MJ3	$n(2s+3)$

jth nonseed residual from the seed subspace[3], and it can be written as $\|(I - P_{m+1})r_j\|^2 = \|r_j\|^2 - \|P_{m+1}r_j\|^2 \geq 0$.

One possible way to minimize that distance, suggested in [34], is to choose as seed a linear combination of the current residuals, in order to increase the spectral content of the underlying subspace. However, this causes the GMRES step to solve an artificial system, which causes extra work. We thus restrict the seed to one of the current residuals. An appropriate choice is σ such that

$$\max_{j=1,\ldots,s} \{\|r_j\|^2 - \|P_{m+1}r_j\|^2\}$$

is minimized. Because P_{m+1} depends on the seed, the second term is not available until after the subspace has been built. On the other hand, the first term depends only on the magnitude of the residuals and is readily computable. Hence, in the absence of any other information, σ is chosen so that $\|r_j\| \leq \|r_\sigma\|$, $j = 1, \ldots, s$. Note that the seed directly affects the Richardson polynomial, which plays an important role in the reduction of the norm of the nonseed residuals. Overall, good seed selection merits further research, especially when application-dependent information is available. A similar problem, in connection with the Sylvester equation, was discussed in [8, §5]. We also note that a new seed system is selected whenever the current seed does not contain enough information to make the remaining systems converge. In practice, this alternation of seeds allows the generation of the components for all systems and was a primary move for restarting the GMRES phase instead of repeating the Richardson phase till convergence.

4. Complexity considerations. The costs of BBiCG and BGMRES rapidly increase with s, since most operations are performed on matrices whose order depends on s; see Tables 1 and 2. In contrast, MHGMRES does not suffer from this dependence. The number of right-hand sides s affects only the number of operations involved, not their complexity. The same holds for GMRES and HGMRES. We refer to the literature regarding complexity for each call to these methods.

Storage requirements for each method (excluding those for A, X, and B) are listed in Table 5. Methods GMRES(m) and HGMRES(m) have also been included, although we note that their costs are independent of s, unless a parallel implementation was desired. The leading storage requirements were accounted for as follows: In GMRES, the primary memory cost is due to the subspace V_m and the factors of the QR decomposition of H. Since the Richardson phase needs no extra memory, HGMRES requires only additional vectors of order m. For MHGMRES, storage is needed for $s - 1$ vectors of dimension $m + 1$ each (cf. line 7 of algorithm MHGMRES). The two leading factors in the complexity of BGMRES are due to the storage for $m + 1$ blocks U_i of dimension $n \times s$ and the storage of the elements of the QR decomposition of the matrix T of dimension $s(m + 1) \times sm$. The leading term, $5ns$, for BBiCG, is due to computations with rectangular iteration matrices of size $n \times s$ [19], while the cost for MJ3 is due to the matrices R and $\bar{R}$ of residuals and auxiliary residuals r_j and $\bar{r}_j$, respectively. All methods handling all

[3]To simplify notation, we omit the iteration superscript i on $r_j^{(i)}$.

right-hand sides simultaneously need to store the matrix of the residuals R. We refer to the Appendix for the precise meaning of the mentioned vectors.

Method BGMRES(m, s) requires a lot of memory, whereas BiCG-type methods enjoy a low memory advantage, when s is small relative to m. Note that the number of right-hand sides affects the storage of MHGMRES only through the space needed for the residuals. Otherwise, the requirements of MHGMRES are comparable to those of standard methods.

5. Numerical experiments. In this section we provide experimental results of using MHGMRES to solve (1) and compare its performance with the other methods described in this paper. More experiments have been reported in [31] and [32]. As we indicated at the beginning of §2, we also compare MHGMRES with standard GMRES and HGMRES; indeed, these proved to be the most formidable competitors. We avoided using other nonsymmetric solvers, since performance comparisons would depend on their effectiveness relative to HGMRES, which was not the purpose of our work.

All codes were written in Fortran, using 64-bit arithmetic. All the experiments were performed on an Alliant Concentrix-2800, running Concentrix 3.0. We are interested in evaluating the methods independently of their parallel processing potential. Hence, the compilation options were chosen not to use parallel processing. In particular, the codes were compiled and executed under the `-Ogv` and `-uniproc` options, respectively. Timings and random values were obtained using the Alliant library functions DTIME and DRAND, respectively. Sparse matrices were stored using compressed sparse row format [26]. Furthermore, the overhead for the construction of the coefficient matrices and right-hand sides was not taken into account. The right-hand sides were chosen as $B_1 = [e_1, \ldots, e_s]$ and $B_2 = \text{RAND}(n, s)$, where function RAND creates a random matrix of dimension $n \times s$ with values uniformly distributed in [0, 1].

In all examples the starting guess $X^{(0)}$ was taken to be zero. The stopping test for iteration index i is $\|r_j^{(i)}\|/\|r_j^{(0)}\| \leq 10^{-7}$ for $j = 1, \ldots, s$. Matrix $\tilde{R}$ in MJ3 was chosen as suggested in [9], namely $\tilde{R} = R$, where R is the initial residual matrix. A maximum number of 500 and 5000 iterations was allowed for BBiCG and MJ3, respectively. A dash signifies that the maximum allowed number of iterations was reached before convergence. We recall that for GMRES-based methods the number of iterations refers to the number of restarts. All algorithms were implemented using early-stopping; that is, as soon as any of the systems converged, these systems were marked and did not participate in any further iterations.

Reference [19] leaves several options open in the implementation of BBiCG. We list our choices next. Matrix $\tilde{R}$ was chosen differently for the two classes of right-hand sides: for B_1, we used $\tilde{R}_1 = A\text{RAND}(n, s)$, whereas for B_2 we used $\tilde{R}_2 = AB_2$. In our experiments these selections produced better results than $\tilde{R} = R$; also compare with [4] and [31]. We have also used orthogonalization of the search direction blocks but no restarting.

Experiments with a partial differential operator. The matrices for the first set of experiments correspond to the five-point discretization of operator

$$L(u) = -u_{xx} - u_{yy} + \beta(u_x + u_y) \tag{6}$$

on the unit square with homogeneous Dirichlet conditions on the boundary. First-order derivatives are discretized by central differences. Two matrices A_1 and A_2 are obtained, corresponding to parameters $\beta = 1$ and $\beta = 100$, respectively. The discretization was performed using a grid size of $h = 1/51$, yielding a matrix of size $n = 2500$. The matrices were scaled by multiplication with h^2. Such matrices were used in [15]. They are unsymmetric and have positive definite symmetric part. The Krylov subspace for GMRES-based methods was of maximum dimension $m = 20$.

Tables 6 and 7 show the number of iterations to convergence for each method for $\beta = 1$ and 100, respectively. One instance where BBiCG is effective is shown in Table 6: the number

TABLE 6

Number of iterations to convergence for operator (6) *with* $\beta = 1$; $n = 2500$; $m = 20$. $B_1 = [e_1, \ldots, e_s]$, $B_2 =$ RAND(n, s).

	s	1	4	8	12	16	20	24	28	32	36	40
B_1	GMRES	10	46	98	154	210	266	322	378	434	490	545
	BGMRES	10	12	12	10	9	9	8	8	8	7	7
	HGMRES	5	20	45	69	90	110	130	150	170	190	210
	MHGMRES	5	6	7	7	8	8	8	8	8	8	8
	BBiCG	135	—	—	—	—	—	—	—	—	—	—
	MJ3	123	426	789	1281	1844	2377	2924	3392	3975	4604	—
B_2	GMRES	21	84	168	252	336	420	504	588	672	756	840
	BGMRES	21	19	18	18	17	17	15	14	12	12	8
	HGMRES	8	30	60	88	118	148	176	206	236	265	293
	MHGMRES	8	11	10	10	10	10	10	11	11	12	11
	BBiCG	135	99	77	68	58	54	48	47	42	40	—
	MJ3	134	512	1161	2102	—	—	—	—	—	—	—

TABLE 7

Number of iterations to convergence for operator (6) *with* $\beta = 100$; $n = 2500$; $m = 20$. $B_1 = [e_1, \ldots, e_s]$, $B_2 =$ RAND(n, s).

	s	1	4	8	12	16	20	24	28	32	36	40
B_1	GMRES	15	60	117	170	217	259	299	335	368	397	421
	BGMRES	15	14	16	15	14	14	14	13	13	13	13
	HGMRES	10	37	72	106	134	160	183	203	220	236	249
	MHGMRES	10	12	13	13	11	12	12	12	12	12	12
	BBiCG	103	—	—	—	—	—	—	—	—	—	—
	MJ3	103	—	—	—	—	—	—	—	—	—	—
B_2	GMRES	10	38	75	112	150	188	225	259	297	335	374
	BGMRES	9	13	12	15	13	13	14	14	13	12	12
	HGMRES	5	21	41	62	83	105	126	146	167	188	208
	MHGMRES	5	10	10	10	11	11	12	12	11	11	11
	BBiCG	113	144	126	—	—	—	—	—	—	—	—
	MJ3	99	—	—	—	—	—	—	—	—	—	—

of iterations for BBiCG decreases as the number of right-hand sides increases, corroborating theoretical results in [19]. Unfortunately, even though we did not observe any loss of rank in the direction and residual matrices, BBiCG frequently failed to converge for some or all of the s right-hand sides in the set; in these cases the maximum residual appeared to persist at values that were a few orders of magnitude above the tolerance. MJ3 also frequently failed to converge; furthermore, it was found to be sensitive to the order in which seed systems are selected for processing, which depends in turn on B. The number of iterations for BGMRES decreased with s, albeit in a less pronounced manner than for BBiCG, whenever the latter was successful. As expected, the number of iterations for GMRES and HGMRES show an almost linear increase with s. In contrast, MHGMRES profits from sharing of information, as can be demonstrated by the very slow increase, and occasional decrease, of the number of restarts with s.

Figures 1(a) and (b) show the runtimes for A_1 and right-hand sides B_1 and B_2 as s varies. The runtimes for GMRES, HGMRES, and MHGMRES demonstrate a linear growth with s, whereas those for MJ3 and the block methods behave in a power-like fashion. In particular, because of the amount of work involved in each step of BGMRES, the drop in the number of iterations observed in the tables did not translate into competitive timings. When it converged, BBiCG outperformed the standard methods, as well as MJ3 and BGMRES. Among all methods,

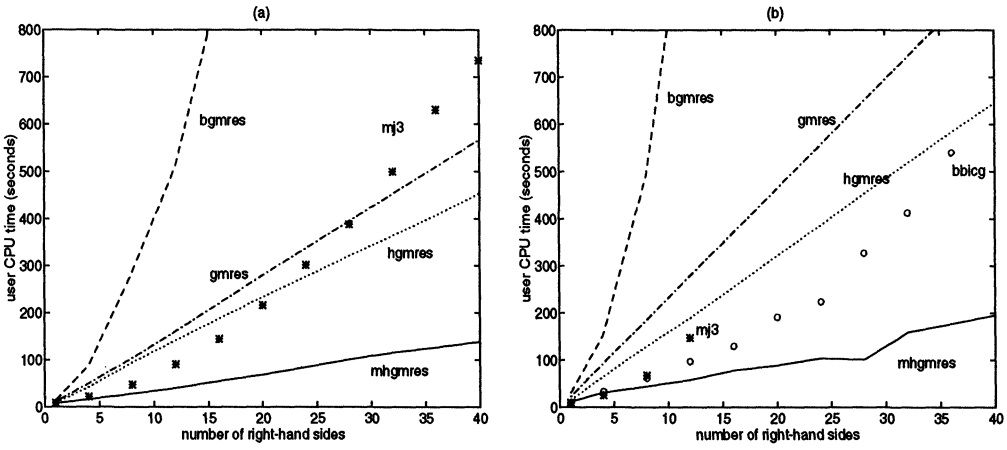

FIG. 1. *Performance of all methods for operator* (6) *with* $n = 2500$; $\beta = 1$; $m = 20$. (a) $B = [e_1, \ldots, e_s]$, (b) $B = \text{RAND}(n, s)$.

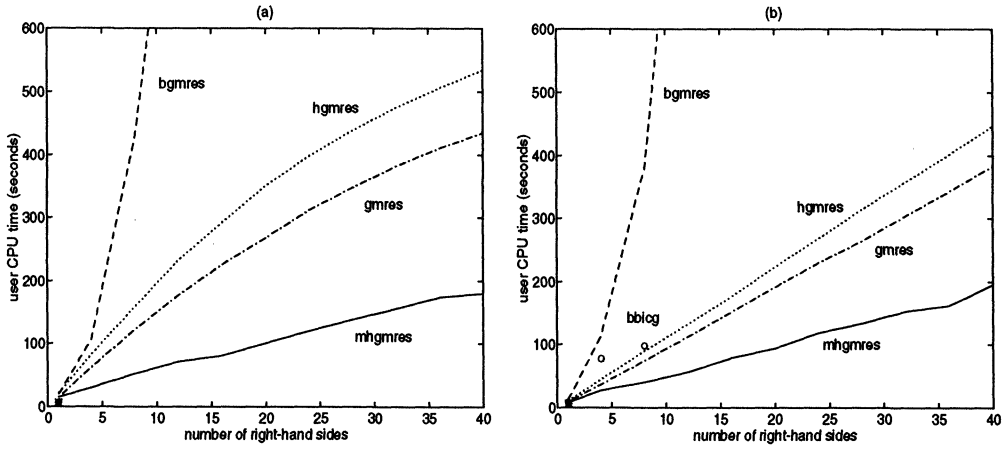

FIG. 2. *Performance of all methods for operator* (6) *with* $n = 2500$; $\beta = 100$; $m = 20$. (a) $B = [e_1, \ldots, e_s]$, (b) $B = \text{RAND}(n, s)$.

MHGMRES has the most satisfactory behavior. Figures 2 (a) and (b) show the behavior of all methods for matrix A_2 and right-hand sides B_1 and B_2. We point out that for $\beta = 100$, GMRES outperforms HGMRES. Results in [32] indicate that this is due to the QZ-based algorithm used at each restart of HGMRES. It is also interesting to see that the cost of using the expensive, QZ-based variant is amortized across the right-hand sides when MHGMRES is used (cf. §3.1).

Because a major goal in our work is to obtain methods that achieve better results than a repetitive application of a single right-hand side solver, we can use the ratio $T(s)/\bar{T}(1)$ as an indicator of the effectiveness of the multiple right-hand side approach. Here $T(s)$ is the time for a given method to converge when applied to s right-hand sides. $\bar{T}(1)$ is the average time to solve for one right-hand side using the underlying single right-hand side solver. For example, to obtain the effectiveness indicator for BGMRES we timed the method for s right-hand sides to obtain $T(s)$. We then computed $\bar{T}(s)$ by dividing by s the time needed for the s right-hand sides using repeated calls to GMRES. We note that using the average $\bar{T}(1)$ is necessary. Otherwise, the results would be skewed, depending on which right-hand side was used to measure $T(1)$.

TABLE 8

Effectiveness of the multiple right-hand side solvers measured by $T(s)/\bar{T}(1)$, where $\bar{T}(1)$ is the average runtime of the underlying single right-hand side solver for one right-hand side; $s = 12$.

		MHGMRES	BGMRES	MJ3	BBiCG
A_1	B_1	3.4	38.0	11.9	—
	B_2	3.6	47.7	17.6	18.9
A_2	B_1	3.7	63.9	—	—
	B_2	5.1	109.3	—	—

TABLE 9

Number of iterations and runtime (in seconds) of MHGMRES (m, s) *and analysis into its major components, with $n = 2500$, $\beta = 100$, and $m = 20$, $B = [e_1, \ldots, e_s]$.*

s	Iterations	Total runtime	GMRES + Projection	QZ + RICHARDSON
1	10	15.6	10.4	5.1
10	13	64.6	19.3	45.2
20	12	101.8	23.4	78.3
30	12	148.0	29.1	117.9
40	12	190.0	34.8	155.1

By definition, the effectiveness indicator for all single right-hand side methods is s. We consider a multiple right-hand side solver to be effective whenever a ratio of less than s is returned. The values of $T(s)/\bar{T}(1)$ for all multiple right-hand side methods for $s = 12$ are listed in Table 8. For all experiments with MHGMRES(m, s), the indicator $T(12)/\bar{T}(1)$ varied between 3.4 and 5.1, which is substantially better than the values observed for all other methods.

In order to better appreciate the cost of each of the components of MHGMRES, we present in Table 9 the runtimes for each of the GMRES and projection phases, and the Richardson phase. The data is from the experiment done with matrix A_2. The variation of the number of iterations with s underlines the benefit of information sharing. The increase in runtime for Richardson is much more pronounced than for the other component, whereas, in contrast to standard methods applied to each right-hand side, the runtime for the GMRES component increases far more slowly. We also note that the cost of the Richardson phase per right-hand side is small compared to that of the GMRES and projection. This indicates that an alternative approach based on s applications of the GMRES phase may be less competitive.

We next consider the performance of the methods when the right-hand sides are specifically chosen to degrade the performance of MHGMRES. We thus chose B to consist of s eigenvectors of matrix A when $\beta = 0$ (hence A is symmetric) and started iterations with zero initial guess. As expected, when $\beta = 0$ the residuals share no information; the remaining values of β cause the residuals to share very little information, in the sense that their dominant projections are on disjoint eigenvectors of A. We then observed the performance of each method for $s = 20, m = 10$ as β took values $0, 10^{-6}, 10^{-3}$, and 1. Results are presented in Table 10. For the extreme case of the symmetric operator, MHGMRES is slower than all other methods. MJ3 demonstrates the best performance when the matrix is very close to being symmetric. As β grew larger than 10^{-6}, compared to other methods, MHGMRES suffered a milder performance degradation, while BBiCG failed to converge (compare with Table 6).

Experiments with matrices from the Harwell–Boeing collection [2]. We present results from experiments with matrices PDE_9511, ORSREG_1, and SHERMAN4. It was observed in studies [16], [37] that for these matrices BiCG had better overall performance than restarted GMRES. Table 11 reports the CPU time for all methods except MJ3, using right-hand sides with random elements. MHGMRES returns the best performance among the GMRES-based methods.

TABLE 10

Runtimes to convergence (in seconds) for operator (6); $n = 2500$; $s = 20$; $m = 10$. Columns of B are eigenvectors of the 2D Poisson operator. A dagger (†) denotes convergence in one iteration. An asterisk (∗) denotes convergence with $m = 1$.

β	GMRES	BGMRES	MHGMRES	HGMRES	BBiCG	MJ3
10^0	239.7	185.3	73.87	104.4	—	—
10^{-3}	47.7	48.2	59.7	34.7	—	—
10^{-6}	2.5*	3.6*	35.6	2.4*	—	1.8
0	2.5*	3.6*	17.9	2.5*	4.6†	1.9

TABLE 11

Runtimes to convergence (in seconds) for Harwell–Boeing collection matrices; $m = 20$; $B = \text{RAND}(n, s)$. A dagger (†) means that the method broke down returning division by 0 or overflow.

Matrix	s	$(A + A^T)$ definite?	GMRES	BGMRES	MHGMRES	HGMRES	BBiCG	
							$\tilde{R} = AB$	$\tilde{R} = A^T B$
PDE_9511	5	yes	17	36	9	16	25	20
($n = 961$)	10		31	84	17	32	—	—
	15		47	206	29	49	30	—
	20		62	319	31	64	—	49
ORSREG_1	5	no	102	304	94	110	—	†
($n = 2205$)	10		222	1061	184	227	—	—
	15		345	2434	239	352	—	—
	20		445	3557	275	474	—	—
SHERMAN4	5	no	53	97	24	48	17	17
($n = 1104$)	10		149	153	35	90	21	21
	15		242	190	44	149	28	—
	20		311	289	56	190	39	41

We also report times for BBiCG using the auxiliary starting block $\tilde{R} = A^T B$. Results are similar to those obtained using $\tilde{R} = AB$. For SHERMAN4, BBiCG returns the best performance over all methods. It is worth observing that ORSREG_1 and SHERMAN4 have indefinite symmetric parts. In general, for such matrices, convergence of restarted GMRES for fixed m cannot be guaranteed a priori (cf. comments following Corollary 3.3). Thus BiCG and other Lanczos-type methods might be more suitable (see experiments in [10] and [17]). It would also be worth exploring adaptive versions of MHGMRESas suggested at the end of §3.1. As it stands, the indefinite problem merits further investigation.

6. Conclusions. MHGMRES seems to be an attractive method for solving nonsymmetric systems with multiple right-hand sides. The method performs best when the right-hand sides share spectral information. Block methods deserve further research. For example, a study of the influence of the block size on stability, the use of restarting to improve performance, and the building of solvers based on recent developments in transpose-free, nonsymmetric Lanczos methods [1], [6].

Appendix.

Block GMRES (see [41]). U_i $(i = 1, \ldots, m+1)$ is of dimension $n \times s$ and denotes the ith block of the orthogonal basis of the block Krylov subspace $K_m(A, R)$. Hence $U_i^T U_j = 0$ for $i \neq j$, and $U_i^T U_i = I$. We assume $(m + 1)s \leq n$. Denote by $H_i = [\chi_1^T, \ldots, \chi_i^T]^T$ a matrix of dimension $(i \cdot s) \times s$ $(i = 1, \ldots, m + 1)$, where χ_k is of order s. $\beta = [\beta_1^T, \ldots, \beta_{m+1}^T]^T$ is of dimension $s \cdot (m + 1) \times s$; $\bar{U}$ and Y are work matrices of dimension $n \times s$ and $s \cdot m \times s$, respectively. Matrix T is triangular of dimension $s \cdot (m + 1) \times s \cdot m$. NEWROT computes and

applies the s^2 rotations necessary to update the QR decomposition of T [41]. T is not stored; instead all computed rotations and the triangular matrix of the QR decomposition are saved.

Algorithm $X = \text{BGMRES}(A, B, X^{(0)}, m, \epsilon)$

$X = X^{(0)}$
$R = B - AX$; $T = 0$
while $\|R\|_\psi > \epsilon$
  $[\beta_1^T, U_1] = \text{MGS}(R)$
  for $i = 1, \ldots, m$
    $\bar{U} = AU_i$
    for $j = 1, \ldots, i$
      $\chi_j = U_j^T \bar{U}$
      $\bar{U} = \bar{U} - U_j \chi_j$
    end
    $[\chi_{i+1}^T, U_{i+1}] = \text{MGS}(\bar{U})$
    $[T, \beta_{i+1}] = \text{NEWROT}(T, H_{i+1})$
  end
  solve $TY = \beta$
  $X = X + [U_1, \ldots, U_m]Y$
  $R = B - AX$
end

Method MJ3 (see [9]). Vectors p^k and $\bar{p}^k$ represent the direction and pseudo-direction vectors, respectively. Parameters α_j^k and β^k are computed in order to minimize the projected nonseed residuals over the subspace spanned by the direction vectors p^k.

Algorithm $X = \text{MJ3}(A, B, X^{(0)}, \epsilon)$

$r_j^0 = \bar{r}_j^0 = b_j - Ax_j^0$, for $j = 1, \ldots, s$
$p^0 = \bar{p}^0 = r_{\bar{j}}^0$, $\bar{j}$ chosen system
$k = 0$
while $(\max \|r_j^k\| > \epsilon)$
  for $j = 1, \ldots, s$
    $\alpha_j^k = ((r_j^k, \bar{p}^k) + (\bar{r}_j^k, p^k))/2(\bar{p}^k, Ap^k)$
    $x_j^{k+1} = x_j^k + \alpha_j^k p^k$
    $r_j^{k+1} = r_j^k - \alpha_j^k Ap^k$
    $\bar{r}_j^{k+1} = \bar{r}_j^k - \alpha_j^k A^T \bar{p}^k$
  end
  test convergence for each $\|r_j\|$
  choose a new $\bar{\bar{j}}$ if $\|r_{\bar{j}}\|$ has converged, otherwise $\bar{\bar{j}} = \bar{j}$
  $\beta^{k+1} = ((r_{\bar{\bar{j}}}^{k+1}, \bar{r}_{\bar{j}}^{k+1}) + (\bar{r}_{\bar{\bar{j}}}^{k+1}, r_{\bar{j}}^{k+1}))/2(r_{\bar{j}}^k, \bar{r}_{\bar{j}}^k)$
  $p^{k+1} = r_{\bar{\bar{j}}}^{k+1} + \beta^{k+1} p^k$
  $\bar{p}^{k+1} = \bar{r}_{\bar{\bar{j}}}^{k+1} + \beta^{k+1} \bar{p}^k$
  $\bar{j} = \bar{\bar{j}}$
  $k = k + 1$
end

Acknowledgments. We thank Miloud Sadkane for providing us with the BGMRES programs, Jacques Laminie for drawing our attention to [9], Dianne O'Leary for her explanations

on [19], Randall Bramley for his criticism and suggestions, and Merle Levy for her editorial comments. We thank Gene Golub, Youcef Saad, Paul Saylor, and David Schneider for several discussions, Raj Mittra for readily providing us with [34], and Qasim Seikh for [13]. We also thank the referees and Howard Elman for criticism that helped us improve the paper.

REFERENCES

[1] A. T. CHRONOPOULOS, *On the squared unsymmetric Lanczos method*, J. Comput. Appl. Math, 54 (1994), pp. 65–78.

[2] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *User's Guide for the Harwell-Boeing Sparse Matrix Collection (Release* I), Tech. Rep. TR/PA/92/86, CERFACS, Toulouse Cedex, France, Oct. 1992.

[3] S. C. EISENSTAT, H. C. ELMAN, AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[4] R. FLETCHER, *Conjugate Gradient Methods for Indefinite Linear Systems*, in Proc. Dundee Biennial Conf. Numer. Anal., G. A. Watson, ed., Lect. Notes Math. 506, Springer-Verlag, Berlin, 1976, pp. 73–89.

[5] R. FREUND, G. GOLUB, AND N. NACHTIGAL, *Iterative solution of linear systems*, Acta Numerica, 1 (1992), pp. 57–100.

[6] ———, *Recent advances in Lanczos-based iterative methods for nonsymmetric linear systems*, in Algorithmic Trends in Computational Fluid Dynamics, M. Hussaini, A. Kumar, and M. Salas, eds., Springer-Verlag, New York, 1993, pp. 137–162.

[7] R. W. FREUND, *Quasi-kernel polynomials and their use in non-Hermitian matrix iterations*, J. Comput. Appl. Math., 43 (1992), pp. 135–158.

[8] D. Y. HU AND L. REICHEL, *Krylov subspace methods for the Sylvester equation*, Linear Algebra Appl., 172 (1992), pp. 283–313.

[9] P. JOLY, *Résolution de systèmes linéaires avec plusieurs seconds membres par la méthode du gradient conjugué*, Tech. Report R-91012, Publications du Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, Paris, March 1991.

[10] W. D. JOUBERT, *Lanczos methods for the solution of nonsymmetric systems of linear equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 926–943.

[11] ———, *A robust GMRES-based adaptive polynomial preconditioning alorithm for nonsymmetric linear systems*, SIAM J. Sci. Comput., 15 (1994), pp. 427–439.

[12] ———, *On the convergence behavior of the restarted GMRES algorithm for solving nonsymmetric linear systems*, J. Numer. Linear Algebra Appl., 1 (1994), pp. 427–448.

[13] S. KHARCHENKO, P. KOLESNIKOV, A. NIKISHIN, A. YEREMIN, M. HEROUX, AND Q. SEIKH, *Iterative solution methods on the Cray YMP/C90. Part* II*: Dense linear systems*, Presented at 1993 Simulation Conference: High Performance Computing Symposium, Washington D.C.

[14] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 33–53.

[15] T. A. MANTEUFFEL, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 183–208.

[16] U. MEIER-YANG, *Preconditioned Conjugate Gradient-Like Methods for Nonsymmetric Linear Systems*, Tech. Report 1210, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, April 1992.

[17] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations*?, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.

[18] N. M. NACHTIGAL, L. REICHEL, AND L. N. TREFETHEN, *A hybrid GMRES algorithm for nonsymmetric matrix iterations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 796–825.

[19] D. P. O'LEARY, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[20] G. OPFER AND G. SCHOBER, *Richardson's iteration for nonsymmetric matrices*, Linear Algebra Appl., 58 (1984), pp. 343–361.

[21] M. PAPADRAKAKIS AND S. SMEROU, *A new implementation of the Lanczos method in linear problems*, Internat. J. Numer. Methods Engnrg., 29 (1990), pp. 141–159.

[22] B. N. PARLETT, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 323–346.

[23] L. REICHEL, *The application of Leja points to Richardson iteration and polynomial preconditioning*, Linear Algebra Appl., 154-156 (1991), pp. 389–414.

[24] Y. SAAD, *Numerical Methods for Large Eigenvalue Problems*, Halstead Press, New York, 1992.

[25] ———, *On the Lanczos method for solving symmetric systems with several right hand sides*, Math. Comp., 48 (1987), pp. 651–662.

[26] ———, *SPARSKIT: A Basic Tool Kit for Sparse Matrix Computation*, Tech. Rep. 1029, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, August 1990.

[27] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[28] M. SADKANE AND B. VITAL, *Davidson's method for linear systems of equations. Implementation of a block algorithm on a multi-processor*, Tech. Report TR/PA/91/60, CERFACS, Toulouse, Sept. 1991.

[29] P. E. SAYLOR AND D. C. SMOLARSKI, *Implementation of an adaptive algorithm for Richardson's method*, Linear Algebra Appl., 154-156 (1991), pp. 615–646.

[30] B. I. SCHNEIDER AND L. A. COLLINS, *The linear algebraic method for the scattering of electrons from atoms and molecules: Computational techniques*, Comput. Phys. Rep., 10 (1991), pp. 51–75.

[31] V. SIMONCINI AND E. GALLOPOULOS, *A memory-conserving hybrid method for solving linear systems with multiple right hand sides*, in Proc. Copper Mountain Conf. Iterative Methods, April 1992; Tech. Rep. No. 1203, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Feb. 1992.

[32] ———, *An Iterative Method for Nonsymmetric Systems with Multiple Right-Hand Sides*, Tech. Report 1242, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, July 1992.

[33] ———, *Convergence Properties of Block GMRES for Solving Systems with Multiple Right-Hand Sides*, Tech. Rep. 1316, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Oct. 1993.

[34] C. F. SMITH, *The performance of preconditioned iterative methods in computational electromagnetics*, Ph.D. thesis, Dept. of Electrical Engineering, Univ. of Illinois at Urbana-Champaign, 1987.

[35] C. F. SMITH, A. F. PETERSON, AND R. MITTRA, *A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields*, IEEE Trans. Antennas and Propagation, 37 (1989), pp. 1490–1493.

[36] G. STARKE AND R. S. VARGA, *A hybrid Arnoldi-Faber iterative method for nonsymmetric systems of linear equations*, Numer. Math., 64 (1993), pp. 213–240.

[37] C. H. TONG, *A Comparative Study of Preconditioned Lanczos Methods for Nonsymmetric Linear Systems*, Tech. Report SAND91-8240 UC404, Sandia National Laboratories, Albuquerque, NM, March 1992.

[38] L. N. TREFETHEN, *Approximation theory and numerical linear algebra*, in Algorithms for Approximation II, J. C. Mason and M. G. Cox, eds., Chapman and Hall, London, 1990, pp. 336–360.

[39] H. A. VAN DER VORST, *Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems*, J. Comput. Phys., 44 (1981), pp. 1–19.

[40] ———, *An iterative solution method for solving $f(A)x = b$ using Krylov subspace information obtained for the symmetric positive definite matrix A*, J. Comput. Appl. Math., 18 (1987), pp. 249–263.

[41] B. VITAL, *Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur*, Ph.D. thesis, Université de Rennes I, Rennes, France, Nov. 1990.