Iterative methods for complex symmetric systems with multiple right-hand sides

V. Simoncini and E. Gallopoulos

December 1993

CSRD Report No. 1322

Center for Supercomputing Research and Development
University of Illinois at Urbana-Champaign
1308 West Main Street
Urbana, Illinois 61801

ITERATIVE METHODS FOR COMPLEX SYMMETRIC SYSTEMS WITH MULTIPLE RIGHT-HAND SIDES*

V. SIMONCINI† AND E. GALLOPOULOS‡

Abstract. The aim of this paper is to introduce and analyze block and nonblock iterative methods to solve $A[x^{(1)},\ldots,x^{(s)}] = [b^{(1)},\ldots,b^{(s)}]$, where A is complex symmetric. The block methods are based on short term recurrences combined with quasi-minimization of the residual. The inner iteration of the nonblock method applies one residual polynomial to several right-hand sides. Implementation of all methods on a shared-memory parallel computer is discussed and results from numerical experiments reported. Comparisons with a single right-hand side solver are included and conditions under which each approach demonstrates superior performance are analyzed.

Key words. complex symmetric matrices, multiple right-hand sides, block methods, nonsymmetric Lanczos algorithm, conjugate gradient algorithm, quasi-minimal residual, parallel processing, sparse matrices, Krylov methods, residual polynomial, computational electromagnetics

AMS subject classifications. 65F10, 65Y20

1. Introduction. The design of methods for the solution of complex symmetric systems with multiple right-hand sides,

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

$A \in \mathbb{C}^{n\times n}$, $A^T = A$, $X, B \in \mathbb{C}^{n\times s}$, and $s \ll n$ is a challenging problem with important applications in areas such as electromagnetics and chemistry [4][24][22][40][44][35]. When n is very large, iterative methods become necessary. Two important issues need to be addressed.

(i) Matrix A cannot always be put in the form $e^{\iota\theta}(T + \iota\rho I)$, where $T = T^H$ and $\rho \in \mathbb{R}$. Therefore, it is not possible, in general, to build iterates based on short recurrences, and at the same time satisfy monotonicity properties for some measure of the error [8][43].

(ii) There are s systems and they all share the same coefficient matrix A. Is it possible to exploit this fact and design methods that are more effective than applying a good single right-hand side solver to each system independently?

One approach for (i) is to extend over the complex field techniques used for real nonsymmetric systems [11]; examples are CG on the normal equations, biconjugate gradient (BiCG), truncated Orthomin; see [40] [33][39] [3] for references from the electromagnetics literature. If we abandon the short recurrence requirement, one could also use the complex version of the generalized minimum residual method of Saad and Schultz (GMRES) [31]; see [4]. Joly and Meurant [18] review several iterative methods and report on experiments for general complex matrices. Related methods are presented in [16][23]; the method of [20] uses polynomial

* This research was supported by the National Science Foundation under grant NSF CCR-9120105 and by ARPA under a subcontract from the University of Minnesota of Grant No. DARPA/NIST-60NANB2D1272. Preliminary results from this paper were first presented at the 1993 Householder Symposium.

† Dipartimento di Matematica Pura ed Applicata, Universitá di Padova, Italy. E-mail simoncin@deis20.cineca.it.

‡ Center for Supercomputing Research and Development and Coordinated Science Laboratory, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois 61801. E-mail stratis@csrd.uiuc.edu.

acceleration on an equivalent set of coupled real systems. It is worth noting that CG-type methods have been proposed and used for the single right-hand side case of (1) in electromagnetics without the usual step of converting to normal equations; see for example [33][34] and the analysis of some of these methods in [41]. It seems, however, that the most thorough study of the problem to date is due to Freund [10]. He proposes and analyzes algorithms that are based on a complex symmetric version of the Lanczos recursions and produce residuals which satisfy a quasi-minimization property. Key is the use of the indefinite bilinear form $\langle x, y\rangle = x^T y$ (termed "natural inner product" for the problem by Craven [6]), instead of the standard inner product $(x, y) = x^* y$ (the superscript "*" denotes the conjugate transpose). Breakdowns due to quasi-null vectors, that is vectors $x \neq 0$ such that $x^T x = 0$, are handled using look-ahead. Under exact arithmetic the algorithm terminates in a finite number of steps unless there is breakdown.

Therefore, in addressing (ii), which is the primary motivation for our work, we start from the methods described in [10] and propose iterative techniques that allow the sharing of information during the solution of (1). An ideal multiple right-hand side method should solve for all systems at a rate that is in general much better than and at worst comparable with solving each system independently. Direct solvers are the methods of choice for solving the single right-hand side problem when the size of the systems is small; see numerical experiments in [18]. The situation is even more favorable for multiple right-hand sides, since the costly factorization of A needs to be performed only once. However, when the size of the systems is very large it becomes necessary to use iterative methods. We consider two approaches. One is to apply block iteration to approach the solution by making use of information extracted from all the systems in each step. The "nonblock" approach is to choose one system at a time and use the extracted information to help the remaining systems proceed toward a solution.

We discussed the real nonsymmetric problem in [37]. The methods we present here are different from those of [37] even though the objectives are the same. Block methods have been investigated in several contexts: [25][26][36] discuss the case of real linear systems; [14][17][32] the eigenvalue problem; [2][15] block methods in control, the latter reference being of particular interest because of its use of block QMR; and [27], which provides a theoretical investigation of the mapping problem of block CG algorithms on parallel computers. We note that we are not aware of any other published presentations of iterative solvers for complex symmetric systems with multiple right-hand sides. This is the subject of our paper, which is structured as follows. In Sections 2.1 and 2.2 the block quasi-minimum residual implementation of CG-like and Lanczos methods are presented; a modified version of the block Lanczos algorithm is described in Section 2.3. In Section 3, we propose a nonblock approach to solve problem (1), and discuss the theoretical justification in Section 3.2. Computational and memory requirements are presented in Section 4. Section 5 contains numerical experiments and actual performance results on a shared-memory parallel computer. Concluding remarks are offered in Section 6.

For the rest of the paper, unless otherwise stated, $A \in \mathbb{C}^{n\times n}$ will be assumed nonsingular. $\sigma_i(A)$ will indicate the i^{th} singular value of A. We denote by 0_m and I_m the zero and identity matrices of order m, by $0_{n,s}$ the zero rectangular matrix, and by $I_{n,s} = [e_1, \ldots, e_s]$ the leading $n \times s$ section of the identity matrix, where e_j denotes the j-th unit vector. $\mathbb{P}_k$ denotes the space of scalar-valued complex polynomials of maximum degree k, and $\bar{\mathbb{P}}_k$ the subset of $\mathbb{P}_k$ with elements satisfying $p(0) = 1$. Similarly $\mathbb{P}_{k,s} := \{\Theta(\lambda) \equiv \alpha_0 + \alpha_1\lambda + \ldots \alpha_k\lambda^k \mid \alpha_0, \ldots, \alpha_s \in \mathbb{C}^{s\times s}\}$ will denote the space of matrix-valued complex polynomials of maximum degree k and order s, and $\bar{\mathbb{P}}_{k,s}$ the subset of $\mathbb{P}_{k,s}$ with elements satisfying $\Theta(0) = I_s$.

Unless otherwise specified, $\|\cdot\|$ will denote the Euclidean norm for vectors and Frobenius norm for matrices. We denote n by s matrices using both their name and column vector notation, e.g., V_k and $[v_k^{(1)}, \ldots, v_k^{(s)}]$. We use semicolons to delimit adjacent rows of matrices, e.g., $[\alpha_{11}, \alpha_{12}; \alpha_{21}, \alpha_{22}]$ for a matrix of order 2. For a matrix with elements $(t_{i,j})$ we denote by $t_{:,j}$ the j^{th} column and by $t_{i,:}$ the i^{th} row. We denote by $\langle\cdot,\cdot\rangle_L$ the bilinear form defined by

$$\langle X, Y\rangle_L := X^T L Y, \quad \text{for } X, Y \in \mathbb{C}^{n\times s}, \text{ and } L \in \mathbb{C}^{n\times n} \text{ symmetric.} \tag{2}$$

We omit the subscript whenever $L = I_n$. Matrices or vectors of random elements are picked from a uniform distribution in $[0, 1]$. Function RAND(k, m) returns a k by m matrix of real random values. Except where noted, operation counts correspond to complex floating-point arithmetic.

2. Block methods. We first discuss some design issues for Krylov-based block methods. In analogy with the single right-hand side solvers of [10], after k steps of our block methods, and assuming that no breakdown has occurred, two sets of matrices would have been generated, namely $\mathcal{P}_k = [P_0, \ldots, P_k]$ and $\mathcal{V}_k = [V_0, \ldots, V_k]$, $P_i, V_i \in \mathbb{C}^{n\times s}$. These matrices satisfy

$$A\mathcal{P}_{k-1} = \mathcal{V}_k \mathcal{T}_k, \tag{3}$$

with the block tridiagonal matrix $\mathcal{T}_k$

$$\mathcal{T}_k = \begin{bmatrix} \delta_0 & \varsigma_1 & \\ \xi_0 & \delta_1 & \ddots \\ & \xi_1 & \ddots \end{bmatrix}, \qquad \mathcal{T}_k \in \mathbb{C}^{(k+1)s\times ks}, \quad \delta_i, \varsigma_i, \xi_i \in \mathbb{C}^{s\times s}. \tag{4}$$

The approximate solution of (1) is given by $X_{k+1} = X_0 + \mathcal{P}_{k-1}y$, for some matrix $y \in \mathbb{C}^{ks\times s}$. Then the residual can be written as $R_{k+1} = R_0 - \mathcal{V}_k\mathcal{T}_k y$.

In exact arithmetic and under special conditions on the starting vectors, the above approach computes the exact solution after $\lceil n/s \rceil$ steps [26]. In practice n/s is very large, and we would like to obtain an approximate solution earlier.

When A is diagonalizable and the starting residual can be written as linear combination of non quasi-null eigenvectors of A, then incurable breakdowns cannot occur in the complex symmetric (nonblock) Lanczos algorithm [10]. In particular, incurable breakdowns are avoided whenever all eigenvalues of A are distinct. Henceforth we will be assuming that the starting residuals have no projection in the directions of quasi-null eigenvectors. We next observe that our algorithms use the indefinite bilinear form (2) and so no minimization is performed, the consequence being that the residual error curve might be very erratic. We thus introduce a block quasi-minimization procedure and investigate its implementation and effect for complex symmetric versions of block Lanczos and CG-like methods.

Let $V_0 := B - AX_0$ with A complex symmetric. Defining $\mathcal{E}_k := [I_s, 0_s, \ldots, 0_s]^T \in \mathbb{R}^{ks\times s}$, the residual can be written as $R_{k+1} = \mathcal{V}_k(\mathcal{E}_{k+1} - \mathcal{T}_k y)$. In general matrix $\mathcal{V}_k$ is not unitary so a scaling diagonal matrix $\mathcal{W}_k = \text{diag}(\omega_0, \ldots, \omega_k)$ is used to normalize the columns of $\mathcal{V}_k = [V_0, \ldots, V_k]$, with $\omega_i = \text{diag}(\|v_i^{(1)}\|, \ldots, \|v_i^{(s)}\|)$; it follows that $R_{k+1} = (\mathcal{V}_k\mathcal{W}_k^{-1})(\mathcal{W}_k(\mathcal{E}_{k+1} - \mathcal{T}_k y))$. Then y is chosen in order to solve

$$(5) \qquad \min_{y\in\mathbb{C}^{ks\times s}} \|\mathcal{W}_k(\mathcal{E}_{k+1} - \mathcal{T}_k y)\|.$$

Note that other choices, yielding different convergence histories, are possible for the scaling matrix $\mathcal{W}_k$; for example we can choose $\mathcal{W}_k$ to be block diagonal. Problem (5) is solved using factorization $\mathcal{W}_k\mathcal{T}_k = \mathcal{Q}_k\mathcal{R}_k$ with $\mathcal{Q}_k$ unitary and $\mathcal{R}_k$ upper triangular. Matrix $\mathcal{T}_k$ is block Hessenberg, thus the QR decomposition is done using Givens rotations. Clearly, these steps are of higher computational complexity than the corresponding ones for the usual QMR, as one has to deal with order-s matrices instead of scalars. For example, in generating $\mathcal{R}_k$, the triangular matrix ξ_k needs to be annihilated at each iteration; see [37][42] for a similar procedure applied to block GMRES. The approximate solution at step k becomes $X_{k+1} = X_0 + \mathcal{S}_k t_k$, with $\mathcal{S}_k = \mathcal{P}_{k-1}\mathcal{R}_k^{-1}$ and $t_k = \mathcal{Q}_k^T\mathcal{E}_{k+1}$. Exploiting the banded structure of $\mathcal{W}_k\mathcal{T}_k$, X_{k+1} can be updated at each iteration by $X_{k+1} = X_k + S_{k+1}\tau_k$, after the computation of the new components S_{k+1} and τ_k of $\mathcal{S}_{k+1} = [S_1, \ldots, S_{k+1}]$ and $t_k = [\tau_0^T, \ldots \tau_k^T]^T$ [10].

We are now in a position to design algorithms based on quasi-minimization of block CG-like and Lanczos-like procedures for complex symmetric matrices. These algorithms correspond to specific choices for $\mathcal{T}_k, \mathcal{V}_k$, and $\mathcal{P}_k$. We note that block Lanczos and CG algorithms for Hermitian systems were first described in [26].

2.1. Block QMRCG-like method. In this section we design a CG-like algorithm for complex symmetric systems, based on the indefinite bilinear form (2), for suitable choice of L, and we combine it with a quasi-minimization procedure in order to improve the behavior of the residual norm.

From $R_{k+1} = R_k - AP_k\alpha_k$, the quantities in (3) are $\mathcal{V}_k = [R_0, \ldots, R_k]$ and $\mathcal{P}_k = [P_0\alpha_0, \ldots, P_k\alpha_k]$. Matrix $\mathcal{T}_k$ is block bidiagonal with $\delta_k = I_s$ and $\xi_k = -I_s$. By using the weight matrix $\mathcal{W}_k = \text{diag}(\omega_0, \ldots, \omega_k)$ we actually compute $\delta_k = \omega_k$, $\xi_k = -\omega_{k+1}$, with $\omega_k = \text{diag}(\|r_k^{(1)}\|, \ldots, \|r_k^{(s)}\|)$. The new approximate solution is updated using the factors of the QR decomposition of $\mathcal{W}_k\mathcal{T}_k$.

The details of the algorithm together with counts for leading floating-point operations of the iteration steps are shown below. The estimate of the total cost per step is presented in Section 4.

Algorithm 2.1
BQMRCG(A, B, X_0, L)
$R_0 = B - AX_0, \quad S_0 = 0_{n,s}$
$[\gamma_0, P_0] = \text{ORTH}(R_0)$
$\omega_0 = \text{diag}(\|r_0^{(1)}\|, \ldots, \|r_0^{(s)}\|), \quad \tau_0 = \omega_0$
$\zeta_0 = \langle R_0, R_0\rangle_L, \quad G_{-1} = I_{2s}$
for $\quad k = 0, 1, \ldots$

$$
\begin{array}{ll}
\epsilon_k = \langle AP_k, P_k \rangle_L, \quad \alpha_k = \epsilon_k^{-1} \gamma_k^T \zeta_k & 2ns^2 \\
R_{k+1} = R_k - AP_k \alpha_k & 2ns^2 + ns + 14/3s^3 \\
\omega_{k+1} = \text{diag}(\|r_{k+1}^{(1)}\|, \ldots, \|r_{k+1}^{(s)}\|) & 2ns \\
\varsigma_k = 0_s, \quad \delta_k = \omega_k, \quad \xi_k = -\omega_{k+1} & \\
[\varsigma_k, \delta_k]^T = G_{k-1}[\varsigma_k, \delta_k]^T & 2s \\
[G_k, \tau_k, \tau_{k+1}, \delta_k] = \text{GIVENS1}(\delta_k, \xi_k, \tau_k) & \\
S_{k+1} = (P_k \alpha_k - S_k \varsigma_k) \delta_k^{-1} & 2s^2 n + 3ns \\
X_{k+1} = X_k + S_{k+1} \tau_k & 2ns \\
\zeta_{k+1} = \langle R_{k+1}, R_{k+1} \rangle_L & 2ns^2 \\
\hat{P}_{k+1} = R_{k+1} + P_k \gamma_k^{-1} \zeta_k^{-1} \zeta_{k+1} & 2ns^2 + ns + 11/3s^3 \\
[\gamma_{k+1}, P_{k+1}] = \text{ORTH}(\hat{P}_{k+1}) & 2ns^2
\end{array}
$$

end

Functions ORTH and GIVENS1 are computational primitives. The application of ORTH on a block of s column n-vectors $\hat{Y}$ generates Y and γ such that $Y = \hat{Y}\gamma$ and $Y^T Y = I_s$ [5]. We observe that such orthogonalization does not ensure the column rank of Y to be equal to the column rank of $\hat{Y}$. Therefore the process is not equivalent to Gram-Schmidt orthogonalization with respect to the usual inner product. Function GIVENS1 applies Givens rotations as described in Appendix A.2. An important feature of the algorithm above is that matrices δ_k and ξ_k are diagonal, so that the block quasi-minimization part of the computation is decoupled among the right-hand sides.

As presented, Algorithm 2.1 does not compute the true residual R_{k+1}^{true} at each step. This can be obtained by using the update

$$
R_{k+1}^{\text{true}} = R_k^{\text{true}} g_{3,k} g_{3,k}^H + R_{k+1} \omega_{k+1}^{-1} g_{4,k}^H \tau_{k+1}, \tag{6}
$$

where $G_k = [g_{1,k}, g_{2,k}; g_{3,k}, g_{4,k}]$, with $g_{j,k} \in \mathbb{C}^{s \times s}$. Furthermore, an approximation for the norm of each residual $r_{k+1}^{(i)}$ is available from $\|r_{k+1}^{(i)}\| \leq s\sqrt{k+2}|t_{i,i}^{(k+1)}|$ with $\tau_k = (t_{i,j}^{(k)})$ [10].

From $\langle AP_k, P_j \rangle_L = 0_s$ for $k \neq j$ and $\langle AP_k, P_k \rangle_L \neq 0_s$ it follows that $\langle A\mathcal{P}_k, \mathcal{P}_k \rangle_L = \text{diag}(\epsilon_0, \ldots, \epsilon_k)$, where $\text{diag}(\epsilon_0, \ldots, \epsilon_k)$ is a block diagonal matrix, with ϵ_i $(0 \leq i \leq k)$ not necessarily diagonal. Finite precision arithmetic causes loss of orthogonality in Algorithm 2.1 and matrices ϵ_k, ζ_k become very ill-conditioned (see Section 5).

An equivalent approach in exact arithmetic would be to force ϵ_k to be diagonal by orthogonalizing in ORTH with respect to $\langle \cdot, \cdot \rangle_{AL}$ instead of $\langle \cdot, \cdot \rangle$. This approach, however, seems to be more sensitive to ill-conditioning of A. Regarding the choice of L, our experiments were conducted using $L = I$, that is, with the quasi-minimal residual version of block CG. We could also choose $L = A$ and thus obtain the quasi-minimal residual version of the block minimum residual method (adapted to complex symmetric matrices).

2.2. Block QMR Lanczos method. In this section we present a block quasi-minimal residual method based on the Lanczos recurrence for complex symmetric matrices [10]. For this algorithm, the quantities in (3) are $\mathcal{P}_k = [Q_0, \ldots, Q_k]$ and $\mathcal{V}_k = \mathcal{P}_k$. Matrix $\mathcal{T}_k$ is block tridiagonal with columns

$$\begin{pmatrix} \beta_k^T \\ \alpha_k \\ \beta_{k+1} \end{pmatrix}, \quad \beta_k, \alpha_k, \beta_{k+1} \in \mathbb{C}^{s\times s},$$

leading to a block three-term recurrence for the Q_i's. The weight matrix $\mathcal{W}_k$ is defined as in BQMRCG.

The algorithm and leading floating-point operation counts are shown below. Function ORTH is as described in the context of BQMRCG. Function GIVENS carries out the incremental block QR decomposition of $\mathcal{T}_k$ and is described in Appendix A.2. Matrices G_{k-1}, G_k contain the rotation coefficients of two preceding steps.

Algorithm 2.2

BQMRLAN(A, B, X_0)
$\hat{Q}_0 = B - AX_0, \quad Q_{-1} = S_{-1} = S_{-2} = 0_{n,s}$
$G_{-2} = G_{-1} = I_{2s}$
$[\beta_0^{-1}, Q_0] =$ ORTH$(\hat{Q}_0)$
$\omega_0 =$ diag$(\|q_0^{(1)}\|, \ldots, \|q_0^{(s)}\|)$
$\tau_0 = \omega_0\beta_0, \quad \omega_{-1} = 0_s$
for $\quad k = 0, 1, \ldots$

$\hat{Q}_{k+1} = AQ_k - Q_{k-1}\beta_k^T \quad \alpha_k = Q_k^T\hat{Q}_{k+1}$	$4ns^2 + ns$
$\hat{Q}_{k+1} = \hat{Q}_{k+1} - Q_k\alpha_k$	$2ns^2 + ns$
$[\beta_{k+1}^{-1}, Q_{k+1}] =$ ORTH$(\hat{Q}_{k+1})$	$2ns^2 + s^3$
$\omega_{k+1} =$ diag$(\|q_{k+1}^{(1)}\|, \ldots, \|q_{k+1}^{(s)}\|)$	$2ns$
$\epsilon_k = 0_s, \quad \varsigma_k = \omega_{k-1}\beta_k^T \quad \delta_k = \omega_k\alpha_k, \quad \xi_k = \omega_{k+1}\beta_{k+1}$	$3s^2$
$[\epsilon_k, \varsigma_k]^T = G_{k-2}[\epsilon_k, \varsigma_k]^T$	$4s^3$
$[\varsigma_k, \delta_k]^T = G_{k-1}[\varsigma_k, \delta_k]^T$	$8s^3$
$[G_k, \tau_k, \tau_{k+1}, \delta_k] =$ GIVENS$(\delta_k, \xi_k, \tau_k)$	
$S_{k+1} = (Q_k - S_k\varsigma_k - S_{k-1}\epsilon_k)\delta_k^{-1}$	$6s^2n + 2ns + s^3$
$X_{k+1} = X_k + S_{k+1}\tau_k$	$2s^2n + ns$

end

The true residual can be computed from

$$R_{k+1}^{\text{true}} = R_k^{\text{true}}\tau_k^{-1}g_{3,k}^H\tau_{k+1} + Q_{k+1}\omega_{k+1}^{-1}g_{4,k}^H\tau_{k+1}, \tag{7}$$

and $\|r_{k+1}^{(i)}\| \leq s\sqrt{k+2}\|t_{:,i}^{(k+1)}\|$ [10]. We note however that unlike the case of BQMRCG the block quasi-minimization procedure in Algorithm 2.2 cannot be decoupled into its s components.

In exact arithmetic, BQMRLAN generates the set $[Q_0, \ldots, Q_k]$ of $(k+1)s$ vectors that are orthogonal with respect to $\langle\cdot,\cdot\rangle$. A three-term recurrence leads to orthogonality between Q_i and Q_j $(i \neq j)$, whereas orthogonality within the columns of Q_i is forced by ORTH; unfortunately finite precision arithmetic causes early loss of orthogonality.

2.3. Stability issues. While it is known that Lanczos and CG-like algorithms are sensitive to roundoff, problems seem to be amplified in the block versions of these algorithms. As we observed, the quasi-minimization computations in BQMRCG are completely decoupled between right-hand sides. Hence any deterioration in accuracy is primarily due to the underlying CG procedure. In particular, ill-conditioning of ϵ_k and ζ_k can cause stagnation (see Section 5). In order to ameliorate the situation we will be using a technique first proposed in [1] for block CG and Hermitian A which uses orthogonalization of the columns of R_k. In that implementation each block P_k is orthogonalized with respect to AP_k. In Sections 4 and 5 We will be using this modification referring to the resulting algorithm as MBCG.

The problem of loss of orthogonality for block Lanczos was discussed in [21]. Errors are amplified both because of computations with full matrices of order s as well as because of three-term recurrences in Q_k and S_k. Lewis showed that

$$\theta_k \leq \left(\|\alpha_k\| \|I_{k+1} - Q_k^T Q_k\| + \|Q_k^T A Q_k - \alpha_k\| + \theta_{k-1} \|\beta_k\| \right) \|\beta_{k+1}^{-1}\|, \tag{8}$$

where $\theta_k = \|Q_k^T Q_{k+1}\|$. Expanding (8) we see that θ_k can grow as the product of cond(β_j) for $j \leq k$. Furthermore, if the norms of the differences to the right of (8) are not 0, there will be further deterioration. One cure for avoiding numerical loss of rank of Q_k is to deflate linearly dependent columns of the matrix but continue updating the corresponding approximate solutions [26]. Deflation can be also useful for eliminating converged systems. In our experiments, however, failure of BQMRLAN was not caused by loss of rank within the block.

Error accumulation from the three-term recurrence can be alleviated by converting to a coupled two-term one as proposed in [12][13] for Lanczos QMR. This helps preserve orthogonality between different Q_k's and mitigates the growth of θ_k at the cost of some additional work. We call the block complex symmetric version of this algorithm BQMRLAN2. Up to scaling, the recurrence matrices are the same as those computed by CG, so that in exact arithmetic BQMRLAN2 and BQMRCG generate the same solution iterates [12] though in practice the two methods give widely different results; see Section 5. Algorithm BQMRLAN2 is described in Appendix A.1; see [12] for a detailed description of the nonblock version of the algorithm from which BQMRLAN2 is derived.

2.4. Error bounds. We next present residual error bounds for Algorithms 2.1 and 2.2 using tools that we developed to analyze block GMRES [38]. First note that from (3) and setting $\mathcal{V}_k = \mathcal{P}_k \mathcal{L}_k$ and $\mathcal{H}_k = \mathcal{T}_k \mathcal{L}_{k-1}$ we have the relation

$$A\mathcal{V}_{k-1} = \mathcal{V}_k \mathcal{H}_k, \tag{9}$$

where $\mathcal{H}_k$ is block tridiagonal. Let $\phi_i \in \mathbb{P}_{i,s}$ be such that

$$V_i = \phi_i(A) \circ R_0 \equiv \sum_{j=0}^{i} A^j R_0 \xi_j^{(i)},$$

where $\xi_j^{(i)} \in \mathbb{C}^{s \times s}$ $(0 \leq i \leq k)$. Then we can write the residuals from Algorithms 2.1 and 2.2 as

$$R_k = \Phi_k(A) \circ R_0 \tag{10}$$

with $\Phi_k(\lambda) = f_k(\lambda)\rho_k \in \bar{\mathbb{P}}_{k,s}$ where $\rho_k = \mathcal{W}_k(\mathcal{E}_{k+1} - \mathcal{H}_k y)$ and $f_k(\lambda) = [\phi_0(\lambda), \ldots, \phi_k(\lambda)]$. As discussed earlier in this section the $\mathcal{W}_k$s are chosen to make the columns of $\mathcal{V}_k$ of unit length, hence $\|V_k\| = s$. Thus we have

$$\|f_k(\lambda)\|^2 = \sum_{j=0}^{k} \|\phi_j(\lambda)\|^2 = s^2(k+1)$$

and it follows that

$$\|\Phi_k(\lambda)\| \leq s\sqrt{k+1}\|\Phi_k^B(\lambda)\|, \tag{11}$$

where Φ_k^B is the minimum residual matrix-valued polynomial that solves

$$\min_{\Theta_k \in \bar{\mathbb{P}}_{k,s}} \|\Theta_k(A) \circ R_0\|.$$

Let $\Lambda_\varepsilon(A) := \{\zeta \in \mathbb{C} : \|(\zeta I - A)^{-1}\| \geq \varepsilon^{-1}\}$ and define $\|\Phi_k\|_{\Lambda,F} := \sup_{\lambda\in\Lambda} \|\Phi_k(\lambda)\|$. The following result can be proved:

THEOREM 2.1 ([38]). *Let $R_k = \Phi_k(A) \circ R_0$ with $R_0, R_k \in \mathbb{R}^{n\times s}$, $A \in \mathbb{R}^{n\times n}$ and $\Phi_k \in \bar{\mathbb{P}}_{k,s}$. Then*

$$\|R_k\| \leq \frac{L_\varepsilon}{2\pi\varepsilon}\|\Phi_k\|_{\Lambda_\varepsilon}\|R_0\|. \tag{12}$$

When A is diagonalizable, by combining with (11) one can obtain a tighter bound [38]. From (11) and (12) it follows that

$$\|R_k\| \leq \frac{L_\varepsilon}{2\pi\varepsilon} s\sqrt{k+1}\|R_0\| \min_{\Theta_k \in \bar{\mathbb{P}}_{k,s}} \sup_{\lambda\in\Lambda_\varepsilon} \|\Theta_k(\lambda)\|,$$

which differs by a factor of $s\sqrt{k+1}$ from the bound obtained using complete minimization; see [9] for a similar result for the nonblock case.

3. A nonblock polynomial algorithm. In contrast to block methods, which in their course generate and share information among all right-hand sides, we study the viability of a scheme which selects a system at a time, extracts information from it, and uses that information to partially solve any remaining systems. Similar principles have guided the creation of solvers for real symmetric [28][29][30][40] and nonsymmetric systems [37].

3.1. Algorithm description. We proceed as follows. Among the systems (1), a "seed" system $Ax^{(\sigma)} = b^{(\sigma)}, \sigma \in \{1, \ldots, s\}$ is selected. Then an approximation $x_k^{(\sigma)}$ to the solution is computed by means of a base algorithm, which we choose to be the complex symmetric version of the minimum residual algorithm. We emphasize here that the complex symmetric version of the algorithm does not produce minimum residuals.

After k iterations, the base algorithm produces $r_k^{(\sigma)} = \phi_k(A)r_0^{(\sigma)}$, $\phi_k \in \bar{\mathbb{P}}_k$. Polynomial ϕ_k is also applied to the non-seed residuals so that $r_k^{(i)} = \phi_k(A)r_0^{(i)}$, $(1 \leq i \leq s)$. Once the seed system has converged, a new seed system is selected and the process is repeated. We observe that the process is implemented by updating the residual $r_k^{(i)}$ and the solution $x_k^{(i)}$, $(1 \le i \le s)$ during each iteration.

The algorithm and leading computational costs are shown below. The total computational cost and comparisons with other methods are presented in Section 4.

Algorithm 3

MULTI.MR(A, B, X_0, ε)
$E = \{1, \ldots, s\}$
$r_0^{(i)} = b^{(i)} - Ax_0^{(i)}, \quad p_0^{(i)} = r_0^{(i)}, \quad i \in E$
$[r_0^{(\sigma)}, \sigma] = \max_{i \in E}(\|r_0^{(i)}\|)$
for $\quad k = 0, 1, \ldots$
  while$(\frac{\|r_k^{(\sigma)}\|}{\|r_0^{(\sigma)}\|} > \varepsilon)$
    $\alpha_k = (r_k^{(\sigma)}, Ap_k^{(\sigma)})/(Ap_k^{(\sigma)}, Ap_k^{(\sigma)})$ $\quad 4n$
    $x_{k+1}^{(i)} = x_k^{(i)} + \alpha_k p_k^{(i)}, \quad i \in E$ $\quad 2ns$
    $r_{k+1}^{(i)} = r_k^{(i)} - \alpha_k Ap_k^{(i)}, \quad i \in E$ $\quad 2ns$
    $\beta_k = -(Ar_{k+1}^{(\sigma)}, Ap_k^{(\sigma)})/(Ap_k^{(\sigma)}, Ap_k^{(\sigma)})$ $\quad 2n$
    $p_{k+1}^{(i)} = r_{k+1}^{(i)} + \beta_k p_k^{(i)}, \quad i \in E$ $\quad 2ns + 2n$
    $I_k := \{i \ : \ \frac{\|r_{k+1}^{(i)}\|}{\|r_0^{(i)}\|} \le \epsilon, i \in E\}$ $\quad 2ns$
    $E = E \setminus I_k$
  end
  $\sigma = \max_{i \in E}(\|r_{k+1}^{(i)}\|/\|r_0^{(i)}\|)$
  if $(\|r_{k+1}^{(\sigma)}\|/\|r_0^{(\sigma)}\|) \le \varepsilon$ then stop
  $p_{k+1}^{(i)} = r_{k+1}^{(i)}, \quad i \in E$
end

Set E consists of indices of systems that have not yet converged. The seed system is chosen to be the one with largest residual norm; we refer to [37] for heuristics on the seed selection strategy.

The residual polynomial is applied implicitly at each iteration, thus preventing the introduction of instabilities from any explicit generation of polynomial coefficients. The method inherits the properties of the underlying single right-hand side solver. Incidentally, there is freedom in the choice of the base polynomial-building algorithm; for instance, one can use the polynomial generated by quasi-minimization applied on the complex symmetric version of CG or Lanczos algorithms. A computational advantage of MULTI.MR is that the number of inner products necessary to solve (1) can be significantly smaller than when applying the base algorithm on each right-hand side. In a parallel implementation, this also implies a reduction in the amount of synchronization and communication.

Roughly speaking, the degree of success of the algorithm depends on the amount by which the polynomials ϕ_k reduce the norm of residuals $r_k^{(i)}$ other than the seed. We next discuss this issue in detail.

3.2. The behavior of the residual polynomial. We next use the properties of the residual polynomial generated by the base algorithm to motivate the design of MULTI.MR. We will be assuming throughout A to be diagonalizable with definite Hermitian part, and will be using notation and results from [10]. For this subsection in particular we will be discussing the behavior of the residual polynomial for a single right-hand side, which permits us to omit the superscript right-hand side identifier. To illustrate the theory, we will be using the order 100 complex symmetric matrix

$$A = T + i\Delta \tag{13}$$

Matrix T corresponds to the scaled 2-D Poisson operator on $[0,1] \times [0,1]$, and matrix Δ is diagonal consisting of random elements. Matrix A has eigenvalues in $(0,8] \times [0,1]$. All experiments in this section were done using Sparse Matlab v. 4.0 on a Sun Sparcstation.

The residual of the minimum residual algorithm can be written as $r_k = \phi_k(A)r_0$, where $\phi_k \in \bar{\mathbb{P}}_k$. When A is Hermitian positive definite, ϕ_k is a normalized kernel polynomial satisfying

$$\phi_k(\lambda) = \arg(\min_{\phi \in \bar{\mathbb{P}}_k} \|\phi\|_\Lambda), \quad \text{with} \quad \|\phi\|_\Lambda := \sup_{\lambda \in \Lambda} |\phi(\lambda)|. \tag{14}$$

For A complex symmetric, ϕ_k is not a kernel polynomial and the method does not minimize the residual at each step. In exact arithmetic, the recurrence of the base algorithm generates a Krylov subspace $\mathbb{K}_m = \text{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$ with $m \leq m_*$, where m_* is the lowest integer for which $\mathbb{K}_m$ becomes an invariant subspace of A, so that $A^{-1}b \in x_0 + \mathbb{K}_{m_*}$. Since A is diagonalizable, the starting residual can be written as $r_0 = \sum_{j=1}^{m_*} c_j v_j$, with $c_j \neq 0$, and $V_{m_*} = [v_1, \ldots, v_{m_*}]$ is the matrix of m_* eigenvectors of A, one for each distinct eigenvalue, such that $V_{m_*}^T V_{m_*} = \text{diag}(\delta_1, \ldots, \delta_{m_*})$ [10]. Let $\Lambda_{m_*} := \{\lambda_1, \ldots, \lambda_{m_*}\}$ with $\lambda_i \neq \lambda_j$. Since the subspace $\mathbb{K}_{m_*}$ is spanned by $\{v_1, \ldots, v_{m_*}\}$, we have $r_{m_*} = \phi_{m_*}(A)r_0 = 0$, and ϕ_{m_*} is the minimal polynomial for r_0.

Consider the discrete functional analog of the bilinear form $\langle x, y \rangle$, that is

$$\langle f, g \rangle_\rho := \frac{1}{m_*} \sum_{i=1}^{m_*} f(\lambda_i) g(\lambda_i) \rho(\lambda_i)$$

with $\rho(\lambda)$ some weight function. The ϕ_k's are orthogonal with respect to $\rho(\lambda) = \lambda\omega(\lambda)$, with $\omega(\lambda)$ the Jacobi weight $\omega(\lambda) = \lambda$, that is, for $k \neq j$,

$$\langle \phi_k, \phi_j \rangle_\rho = \frac{1}{m_*} \sum_{i=1}^{m_*} \phi_k(\lambda_i) \phi_j(\lambda_i) \lambda_i^2 = 0.$$

Thus, the residual polynomial satisfies some properties of kernel polynomials, in particular orthogonality with respect to $\langle f, g \rangle_\rho$ and a three-term recurrence. The lack of minimality is due to the indefiniteness of the bilinear form $\langle f, g \rangle_\rho$.

Our point is that in practice ϕ_k demonstrates a behavior that can be exploited in order to build iterative methods for complex symmetric systems. This is illustrated in Figure 1 where the zeros of ϕ_k are plotted for matrix (13) starting from a vector of real random elements. For comparison, we also plotted the zeros of the kernel minimal residual polynomial of GMRES. We see that the polynomials corresponding to both methods have similar zero patterns, the similarity being stronger in the vicinity of those eigenvalues of A having smallest and largest real parts. It is especially interesting to observe that $|\phi_k(\lambda)|$ is small on Λ_{m_*} even though ϕ_k does not satisfy (14). Figure $2(a)$ shows the values of $|\phi_k(\lambda)|$ as λ takes values in the domain $(0, 8] \times [0, 1]$ that contains the spectrum of A.

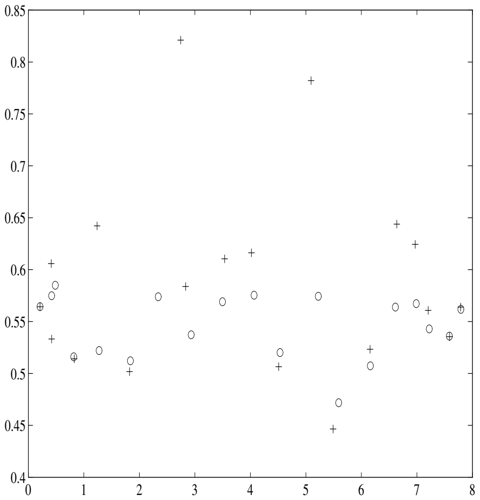

FIG. 1. *Zeros of the* GMRES *kernel polynomial of degree* $k = 19$ *('o') and zeros of the residual polynomial for the complex symmetric version of the minimum residual algorithm of degree* $k = 19$ *('+') for* A *in (13) and random starting vector.*

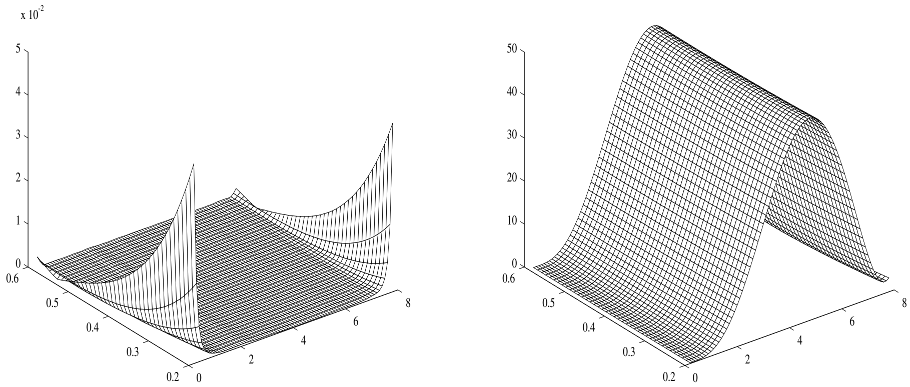

FIG. 2. *Plot of* $|\phi_k(\lambda)|$ *for* $\lambda \in (0, 8] \times [0, 1]$*; (a)* $k = 21$ $(\|r_{21}\|/\|r_0\| < 10^{-5})$*,* $r_0 = \text{RAND}(n, 1)$*. (b)* $k = 5$ $(\|r_5\|/\|r_0\| < 10^{-14})$*,* $r_0 = \sum_{j=1}^{5} v_i$*.*

We next show that a condition similar to (14) holds for the complex symmetric polynomial method, in particular that $|\phi_k(\lambda)|$ varies on Λ_{m_*} within some factor of $\|r_k\|$; furthermore, this factor only depends on the eigencomponents of the starting residual. Let V_{m_*} be free of quasi-null vectors and construct the square matrix $\mathcal{B} := [V_{m_*} V_{n-m_*}] \in \mathbb{C}^{n\times n}$, where V_{n-m_*} is some n by $n - m_*$ block. Then we can write the polar decomposition $\mathcal{B} = M\mathcal{U}$ where $\mathcal{U} \in \mathbb{C}^{n\times n}$ is unitary and $M \in \mathbb{C}^{n\times n}$ is positive semidefinite. Hence $\mathcal{B}I_{n,m_*} = M\mathcal{U}I_{n,m_*}$ so that $V_{m_*} = MU_{m_*}$ with $U_{m_*} := \mathcal{U}I_{n,m_*}$ and $U_{m_*}^H U_{m_*} = I_{m_*}$. Writing the residual $r_k = \sum_{j=1}^{m_*} c_j\phi_k(\lambda_j)v_j$ as $r_k = M\sum_{j=1}^{m_*} c_j\phi_k(\lambda_j)u_j$ we have

$$\|r_k\| \geq \sigma_{\min}(M)\|\sum_{j=1}^{m_*} c_j\phi_k(\lambda_j)u_j\|,$$

from which

$$\left(\sum_{j=1}^{m_*} |c_j|^2|\phi_k(\lambda_j)|^2\right)^{\frac{1}{2}} \leq (\sigma_{\min}(M))^{-1}\,\|r_k\|.$$

We have thus proved the following result.

THEOREM 3.1. *Let A be diagonalizable with eigenvalues $\Lambda_{m_*} = \{\lambda_1, \ldots, \lambda_{m_*}\}$ and let $M \in \mathbb{C}^{n\times n}$ satisfy $V_{m_*} = MU_{m_*}$ with $U_{m_*}^H U_{m_*} = I_{m_*}$. Let also $r_0 = \sum_{j=1}^{m_*} c_jv_j$, and for functions f, g define $(f, g)_\omega := \frac{1}{m_*}\sum_{j=1}^{m_*} f(\lambda_j)\overline{g(\lambda_j)}\omega(\lambda_j)$, with $\omega(\lambda_j) = |c_j|^2$, and $\|f\|_\omega = (f, f)_\omega^{\frac{1}{2}}$. Then*

$$\|\phi_k\|_\omega \leq (\sigma_{\min}(M))^{-1}\,\|r_k\|.$$

We remark that the inner product $(f, g)_\omega$ introduced in Theorem 3.1 defines a norm in $\mathbb{C}$.

3.3. Using a single residual polynomial ϕ_k on multiple right-hand sides. Let A be diagonalizable and let $m_*^{(i)}$ be the dimension of the smallest invariant subspace of A generated by $r_0^{(i)}$. Define $\Lambda^{(i)} := \Lambda_{m_*^{(i)}}$ to be the part of spectrum of A whose eigenvectors are the components of $r_0^{(i)}$. After k iterations of MULTI.MR on $r_0^{(i)} = \sum_{j=1}^{m_*^{(i)}} c_j^{(i)}v_j$, $i = 1, \ldots, s$, we have $r_k^{(i)} = \phi_k(A)r_0^{(i)}$, that is

$$r_k^{(i)} = \sum_{j=1}^{m_*^{(i)}} c_j^{(i)}\phi_k(\lambda_j)v_j.$$

From Theorem 3.1 it follows that $|\phi_k(\lambda_j)|$ can be small for $\lambda_j \in \Lambda^{(i)} \cap \Lambda^{(\sigma)}$. Thus, whenever $\Lambda^{(i)} \cap \Lambda^{(\sigma)} \neq \emptyset$, that is whenever $r_0^{(i)}$ has projection on the spectral components of $r_0^{(\sigma)}$, the application of ϕ_k on non-seed systems can reduce $\|r_k^{(i)}\|$.

Figure $2(b)$ shows $|\phi_k(\lambda)|$ for matrix A in (13) using b with eigencomponents corresponding to some of the largest and smallest eigenvalues of A. In accordance with Theorem (13), $|\phi_k(\lambda)|$ is small in the neighborhood of eigenvalues with eigenvectors participating in the initial residuals. Furthermore $|\phi_k(\lambda)|$ is also small in the vicinity of each $\lambda_i \in \Lambda^{(\sigma)}$. Thus, even for $\lambda_j \notin \Lambda^{(i)} \cap \Lambda^{(\sigma)}$, $|\phi_k(\lambda_j)|$ can decrease with $\|r_k^{(i)}\|$, as long as the distance $\mathrm{dist}(\lambda_j, \Lambda^{(\sigma)})$ remains small. We summarize our observations in the following theorem.

THEOREM 3.2. *Let A be diagonalizable with $\Lambda^{(i)}$ the set of eigenvalues of A corresponding to the eigencomponents of $r_0^{(i)}$. For each $\lambda_l \in \Lambda^{(i)}$, let $d_{j_l} := |\lambda_l - \lambda_{j_l}| = \min_{\lambda_j \in \Lambda^{(\sigma)}} |\lambda_l - \lambda_j|$. Define $\mathcal{D}(d_{j_l})$ such that $\phi_k(\lambda_l) = \phi_k(\lambda_{j_l}) + \mathcal{D}(d_{j_l})$ with $r_k^{(\sigma)} = \phi_k(A) r_0^{(\sigma)}$. Let also M_i be such that $V_{m_*^{(i)}} = M_i U_{m_*^{(i)}}$.*

Then, for each i, $1 \leq i \leq s$ there exists a constant $L_{i,\sigma}$ such that

$$\|r_k^{(i)}\| \leq \left[L_{i,\sigma} m_*^{(i)} \|r_k^{(\sigma)}\|^2 + \|M_i\|^2 \sum_{\lambda_l \in \Lambda^{(i)} \setminus \Lambda^{(\sigma)}} |c_l^{(i)}|^2 |\mathcal{D}(d_{j_l})|^2 \right]^{\frac{1}{2}}, \qquad \lim_{\lambda_l \to \lambda_j} |\mathcal{D}(d_{j_l})| = 0.$$

Proof. Using the definition of $\mathcal{D}(d_{j_l})$ and the fact that $\mathcal{D}(d_{j_j}) = 0$, the residual $r_k^{(i)} = \sum_{\lambda_l \in \Lambda^{(i)}} c_l^{(i)} \phi_k(\lambda_l) v_l$ can be written as

$$r_k^{(i)} = \sum_{\lambda_l \in \Lambda^{(i)}} c_l^{(i)} \phi_k(\lambda_{j_l}) v_l + \sum_{\lambda_l \in \Lambda^{(i)} \setminus \Lambda^{(\sigma)}} c_l^{(i)} \mathcal{D}(d_{j_l}) v_l.$$

Matrix U_{m_*} is unitary, thus

$$\|r_k^{(i)}\| \leq \|M_i\| \left[\sum_{\lambda_l \in \Lambda^{(i)}} |c_l^{(i)}|^2 |\phi_k(\lambda_{j_l})|^2 + \sum_{\lambda_l \in \Lambda^{(i)} \setminus \Lambda^{(\sigma)}} |c_l^{(i)}|^2 |\mathcal{D}(d_{j_l})|^2 \right]^{\frac{1}{2}}.$$

Moreover,

$$\begin{aligned} \sum_{\lambda_l \in \Lambda^{(i)}} |c_l^{(i)}|^2 |\phi_k(\lambda_{j_l})|^2 &\leq \left(\max_{l, j_l} \frac{|c_l^{(i)}|^2}{|c_{j_l}^{(\sigma)}|^2} \right) \sum_{\lambda_l \in \Lambda^{(i)}} |c_{j_l}^{(\sigma)}|^2 |\phi_k(\lambda_{j_l})|^2 \\ &\leq \left(\max_{l, j_l} \frac{|c_l^{(i)}|^2}{|c_{j_l}^{(\sigma)}|^2} \right) m_*^{(i)} \sum_{\lambda_j \in \Lambda^{(\sigma)}} |c_j^{(\sigma)}|^2 |\phi_k(\lambda_j)|^2 \end{aligned}$$

where the summation is now performed over all eigenvalues in $\Lambda^{(\sigma)}$. From Theorem 3.1,

$$\sum_{\lambda_j \in \Lambda^{(\sigma)}} |c_j^{(\sigma)}|^2 |\phi_k(\lambda_j)|^2 \leq (\sigma_{\min}(M_\sigma))^{-2} \|r_k^{(\sigma)}\|^2$$

Letting $L_{i,\sigma} = \left(\max_{l, j_l} \frac{|c_l^{(i)}|^2}{|c_{j_l}^{(\sigma)}|^2} \right) \left(\frac{\|M_i\|}{\sigma_{\min}(M_\sigma)} \right)^2$ the result follows. □

Theorem 3.2 suggests that the performance of MULTI.MR is affected by both the dimension of $\Lambda^{(i)} \cap \Lambda^{(\sigma)}$, $i \neq \sigma$ and the clustering of the spectrum of A.

Clearly, for λ_l not in the neighborhood of eigenvalues whose eigenvalues participate in the seed residual, $\mathcal{D}(d_{j_l})$ can be very large. In those cases, the application of the polynomial will lead to a deterioration of the residual. While this seems to render ineffective the shared polynomial approach, the problem is caused by the current seed's relation to the remaining residuals. Once this seed system has been solved, conditions for the next seed are likely to be different. In such circumstances, it is critical to avoid using the polynomial for any residuals for which its behavior becomes noncontracting. One effective strategy consists of monitoring

TABLE 1
Leading complex floating-point operations for some basic linear algebra operations

operation	$Y\alpha$	$\alpha\beta, \alpha^T\beta$	Y^TY	$\alpha^{-1}\beta$	$\epsilon^{-1}\beta$	$\delta\beta$	$Y+Y$	$Y\delta$	diag($\|y_i\|$)
additions	ns^2	s^3	ns^2	$4/3s^3$	$s^3/2$	-	ns	-	ns
multiplications	ns^2	s^3	ns^2	$4/3s^3$	$s^3/2$	s^2	-	ns	ns

the residuals and stopping the application of the current polynomial on those residuals that become very large. An early detection of such problems also helps reduce unnecessary work on residuals that are not helped by the current polynomial. It is worth noting that block methods are also sensitive to the composition of the starting residuals [38]. This point will be discussed further in Section 5.

4. Cost analysis. The aim of this section is to evaluate the cost per iteration of all methods presented so far.

Let $Y \in \mathbb{C}^{n\times s}, \alpha, \beta, \epsilon, \delta \in \mathbb{C}^{s\times s}$, with ϵ triangular and δ diagonal matrices. The basic costs for kernel matrix operations in terms of complex floating-point operations are listed in Table 1.

Table 2 shows the leading computational costs per iteration per method, excluding the cost for calculating the residual norms. For comparison purposes we have also included the costs for the single right-hand side method TFQMR and the complex symmetric MBCG. Since the quasi-minimization phase of block methods requires operations with full matrices of order s, both BQMRLAN and BQMRLAN2 have large coefficients for terms ns^2 and s^3. We also note that when s is not very small relative to n, the s^3 terms become comparable to the ns^2 ones. The coefficient for the ns term in BQMRCG is due to computations with diagonal matrices.

TABLE 2
Computational cost per iteration

method	complex flops	$A\cdot x$
BQMRCG	$12ns^2 + 25/3s^3 + 9ns$	s
MBCG	$14ns^2 + 6s^3 + 3ns$	s
BQMRLAN	$16ns^2 + 14s^3 + 7ns$	s
BQMRLAN2	$14ns^2 + 26/3s^3 + 6ns$	s
MULTI.MR	$8ns + 8n$	s
QMR$\times s$	$18ns$	s

Under favorable circumstances, block methods require fewer iterations as s increases, but at greater cost per iteration. In particular, the cost increase is polynomial in s; thus a large reduction in the number of iterations will be needed for them to be competitive.

The density of matrix A provides an indication for the expected performance of block methods in comparison to other nonblock solvers. Let "nnz" denote the total number of nonzero elements in A. Then nnz$/n$ is the average number of nonzero elements per row. The cost of forming $A\cdot X$ with $X \in \mathbb{C}^{n\times s}$ is approximately $2s$nnz complex operations[1].

[1] To keep a uniform treatment we lump complex additions and multiplications together; however, the cost of the former is smaller than the cost of the latter.

TABLE 3
Memory requirements

method	matrices $n \times s$	matrices $s \times s$	matrices diag(s)	rotation coefs.† s, s^2	vectors n
BQMRCG	P_k, R_k S_k	$\epsilon_k, \gamma_k, \zeta_k$ wrk^*	$\omega_k, \varsigma_k, \delta_k$ τ_k, τ_{k+1}	$\mathcal{C}, \mathcal{S}$	
MBCG	P_k, R_k	$\beta_k, \gamma_k, wrk*$			
BQMRLAN	Q_k, Q_{k+1} S_k, S_{k+1}	$\alpha_k, \beta_k, \varsigma_k, \delta_k$ $\xi_k, \epsilon_k, \tau_k, \tau_{k+1}$	ω_k	$\mathcal{C}_{k-1}, \mathcal{S}_{k-1}$ $\mathcal{C}_k, \mathcal{S}_k$	
BQMRLAN2	P_k, V_k S_k	$\rho_k, \epsilon_k, \varsigma_k,$ $\xi_k, \delta_k, \tau_k, \tau_{k+1}$	ω_k	$\mathcal{C}, \mathcal{S}$	
MULTI.MR	P_k, R_k				$v_k^{(\sigma)}$
QMR$\times s$	V_k, V_{k-1} P_k, P_{k-1}		$\alpha_k, \beta_k, \eta_k$ $\theta_k, \xi_k, \tau_k, \tau_{k+1}$	$\mathcal{C}, \mathcal{S}$	

* work matrix. † see Appendix A.2.

From Table 2 we see that the leading cost of block methods is a multiple of ns^2, whereas the costs for the nonblock method and the single right-hand side solver are multiples of ns. Then if we denote by i_B and i_S the number of iterations for a block and nonblock method to converge and compare BQMRCG with MULTI.MR, we see that their costs will be similar when $(12ns^2 + 2s\text{nnz})i_B \approx (8ns + 2s\text{nnz})i_S$. This means that for the block method to be successful, at first approximation we should have

$$i_B \leq \frac{8ns + 2s\text{nnz}}{12ns^2 + 2s\text{nnz}} i_S \;\; = \left(1 - \frac{6s - 4}{6s + \frac{\text{nnz}}{n}}\right) i_S. \tag{15}$$

It follows that for $\frac{\text{nnz}}{n}$ small and s large, the block method achieves better total complexity only for very small i_B; on the other hand, a large density per row allows the block method a number of iterations which could be sufficient to attain convergence at lower total cost than the nonblock method. Table 3 shows the leading memory requirements, excluding storage for common matrices A, X and B. Depending on implementation, additional work vectors may be needed. We refer to [1] and [10] for the meaning of the variables of MBCG and QMR respectively. The column "rotation coefs." lists vectors used to store the Givens rotation coefficients; see Appendix A.2 for details. As before, we note the rapid increase with s in the cost of the block methods.

5. Experiments. In this section we present experiments to illustrate the performance of all methods described earlier, including MBCG. We also report the performance of QMR applied to each right-hand side, for the sake of comparison with a state-of-the-art solver for complex symmetric systems. The matrices in Examples $1, 2, 4$ originate in electromagnetic scattering and computational chemistry. In both cases, the boundary conditions are selected to lead to a complex symmetric matrix. The matrix characteristics are shown in Table 4. As right-hand sides we used random values $B_1 = \text{RAND}(n, s)$ and sections of identity matric $B_2 = I_{n,s}$ for several values of s. Random values were computed using the Alliant intrinsic function `drand`. Observe that both B_1 and B_2 are real. In all cases the starting guess was

TABLE 4
Characteristics of matrices used in the experiments

Example	*matrix name*	*matrix order*	nnz: *# nonzero elements*	*origin*
1	RWG	635	7,881	em-scattering
2	RXS	10,180	154,102	em-scattering
3	YOUNG1C	841	4,089	Harwell-Boeing set
4	ELQ	1,197	61,373	magnetic resonance
5	diagonal matrix	1,000	1000	artificial

chosen to be $X_0 = 0_{n,s}$.

5.1. Implementation details. All experiments were conducted on an Alliant FX/2800 shared memory multiprocessor running Concentrix 3.0, using at most 12 processors. Each processor has peak rate of 40 Mflops for real double precision arithmetic. Processors are based on the RISC architecture and offer extensive pipelining which will be referred to as vectorization. The system command `execute -c`p was used to force the system to use a specific number (p) of processors. Computations were performed in double precision using complex arithmetic.

Matrix A is is stored in compressed row format so that no special structure is assumed. An important kernel in all algorithms is multiplication of dense blocks of s columns by A. This operation was accomplished by vectorizing across the right-hand sides and parallelizing across rows. It is worth noting that the presence of multiple right-hand sides improves data locality, the absence of which is a computational deficiency of sparse matrix operations on computer systems with hierarchical memories. In particular, computing $A \cdot X$, where $A \in \mathbb{C}^{n\times n}$ and $X \in \mathbb{C}^{n\times s}$ requires a total of $s(\text{nnz} - n)$ complex additions and $s\text{nnz}$ complex multiplications on a total of nnz $+ \, ns$ elements. Counting one complex addition as two real additions and one complex multiplication as four real multiplications and two real additions, there is an average of

$$\text{loc}(s) = \frac{s(8\text{nnz} - 2n)}{2\text{nnz} + sn}$$

real floating-point operations per data element. It follows that $\text{loc}(s)$ increases with s. We note that in our implementation QMR (the single right-hand side solver of choice) is applied on s right-hand sides at a time. Consequently QMR uses the same sparse matrix by dense matrix computational kernel as the remaining algorithms presented here. All kernels listed in Table 1 involve operations with dense blocks of order s. Parallelization was achieved by parallelizing across the s columns of the right-hand operand. Whenever possible we used the Alliant scientific library intrinsic functions `matmul`, `dotproduct` and `transpose`. We must qualify our experiments, however, with the comment that the parallel implementation deserves a separate comprehensive study and remains to be done.

The convergence test used for all methods was

$$\max_{i=1,\ldots,s} \frac{\|b^{(i)} - Ax_k^{(i)}\|}{\|b^{(i)} - Ax_0^{(i)}\|} \leq 10^{-7}. \tag{16}$$

Occasionally the number of iterations for a method is suffixed by an asterisk ("*"); this means that even though the method has converged, systems with very ill-conditioned matrices (spectral condition number larger than 10^{13}) had to be solved. All methods stopped iterating for those right-hand sides for which convergence was detected.

For each method we obtained the number of iterations required and the total elapsed time, in seconds, as measured by the intrinsic function `dtime`; however, we report timings in terms of average time per right-hand side in order to highlight the performance of each method as the number of right-hand sides increases. The number of iterations for each method also provides a measure of effectiveness for each method class (block and nonblock).

From Theorem 3.2 it follows that the polynomial generated by the seed system may have large values on the spectral components of some of the non-seed systems, increasing the residual enormously. A residual monitoring procedure is thus essential for MULTI.MR to succeed. Our implementation stops updating systems for which the magnitude of the residual norm becomes larger than $\epsilon_1 = 10^7$, until a new seed system is generated. For stability reasons, coefficients β_k in algorithm BQMRLAN were computed explicitly, instead of reporting the values from the orthogonalization of the previous iteration.

5.2. Test problems. The first two problems arise in the solution of the three-dimensional Maxwell equations using finite elements in a square plate shaped scatterer in the $x - y$ plane of side $1m$, centered at the origin. The region was tesselated using two types of edge-elements and a Galerkin approximation was carried out.

Example 1. The matrix name is RWG and was created using 697 tetrahedral edge-elements, for a total of 1069 edges, and 434 edges on the perfect electric conductor (PEC) [22]. The number of right-hand sides ranged from 1 to 40. Results are reported in Table 5 and Figures 3(a) and 3(b).

Most methods converge for both types of right-hand sides; however BQMRCG failed for B_2 and $s > 6$. This is due to ill-conditioning of the order s matrices involved in the computation; see Section 2.3. This can also be seen from results for B_1 and $s \geq 20$. The results for MBCG imply that it is worthwhile to orthogonalize the residual matrices. Furthermore, they show that the loss of rank involved in BQMRCG is due to roundoff and would not have occurred in exact arithmetic; see Section 2.3. Round-off error also affects the convergence of BQMRLAN. We also note that the number of iterations for MULTI.MR increases sharply for $s > 20$. These were cases where the monitoring procedure detected deterioration of some residual norms and frequently had to stop the application of the seed polynomial on these residuals.

From the figures we see that the performance of MULTI.MR stabilizes for large values of s. This is expected, since a fixed number of iterations is needed for the seed system to converge.

Overall, MULTI.MR has the best performance and all methods perform better than QMR, implying that information sharing is very effective in this problem.

Example 2. The next matrix, named RXS, comes from the finite element approximation of the three-dimensional Maxwell equations using 7,916 tetrahedral edge-elements having

TABLE 5

Number of iterations to satisfy (16). matrix RWG. $n = 635$. B_1=RAND(n, s), $B_2 = I_{n,s}$.

rhs	s / method	1	4	8	12	16	20	24	28	32	36	40
B_1	BQMRCG	493	158	92	62	47	51*	30*	32*	22*	21*	18*
	MBCG	498	147	86	59	44	35	29	24	21	19	17
	BQMRLAN	492	148	82	-	43	35	28	24	21	18	17
	BQMRLAN2	497	147	83	57	44	35	29	24	21	19	17
	MULTI.MR	496	508	508	502	501	501	501	501	510	510	510
	QMR	499	501	506	509	509	512	512	512	513	513	513
B_2	BQMRCG	464	162	-	-	-	-	-	-	-	-	-
	MBCG	468	145	86	62	48	40	30	25	21	19	17
	BQMRLAN	465	147	83	60	45	38	29	24	21	18	17
	BQMRLAN2	465	146	83	60	46	38	29	24	21	19	17
	MULTI.MR	462	476	493	493	493	494	923	938	964	964	965
	QMR	467	472	478	482	488	488	488	488	488	488	499

* *ill-conditioned.*

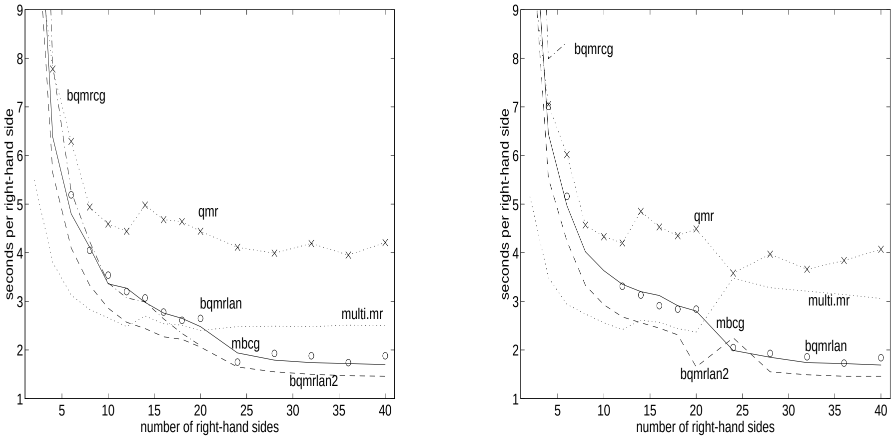

FIG. 3. *Time per right-hand side, in seconds, using matrix* RWG. *(a)* B = RAND(n, s)*, (b)* $B = I_{n,s}$*. Symbols: '.-.'* BQMRCG*, '–'* MBCG*, 'o'* BQMRLAN*, '- -'* BQMRLAN2*, '...'* MULTI.MR*, '..x..'* QMR.

10,268 edges in total, out of which 88 are perfect electric conductor (PEC) edges [22]. Exceptionally this problem was solved using diagonal preconditioning. In particular we solved $DADD^{-1}X = DB$, where $D := (\text{diag}(\text{real}(A)))^{-1/2}$. Table 6 shows the number of iterations for $s \leq 10$. We observe the numerical effectiveness of block methods from the reduction in the number of iterations as s increases. Unfortunately, Figure 4 shows that their overall performance is disappointing. In particular it is worse than QMR. Once again, the best method seems to be MULTI.MR. This shows that MULTI.MR achieves the best trade-off between information sharing and computational work.

The inferior performance of block methods is due to the size of n. From our Table 2 and the discussion leading to formula (15), we see that the reduction in the number of iterations for the block methods is only moderate, and they cannot equal the performance of

TABLE 6
Number of iterations to satisfy (16). matrix RXS $n = 10180$. B_1=RAND(n, s), $B_2 = I_{n,s}$.

rhs	s / method	1	2	4	6	8	10
B_1	BQMRCG	1188	1042	945	869	833	763
	MBCG	1085	1039	959	881	832	768
	BQMRLAN	1190	1039	918	850	-	-
	BQMRLAN2	1049	959	848	782	722	687
	MULTI.MR	1129	1187	1180	1149	1149	1146
	QMR	1052	1063	1090	1123	1123	1123
B_2	BQMRCG	1083	986	901	851	790	757
	MBCG	1085	997	884	849	797	753
	BQMRLAN	1086	978	857	814	739	715
	BQMRLAN2	938	905	769	742	688	653
	MULTI.MR	1035	1075	1067	1114	1114	1115
	QMR	940	1014	1014	1014	1014	1014

MULTI.MR. Note that the high cost of MBCG is due to the double orthogonalization required at each iteration.

This experiment corroborates our earlier discussion that for very large matrices, block methods cannot be effective unless the computation of $A \cdot X$ (the predominant kernel in non-block methods) becomes very expensive compared to inner products and other computations with dense n by s blocks, as would be the case for example when A is dense [19][21].

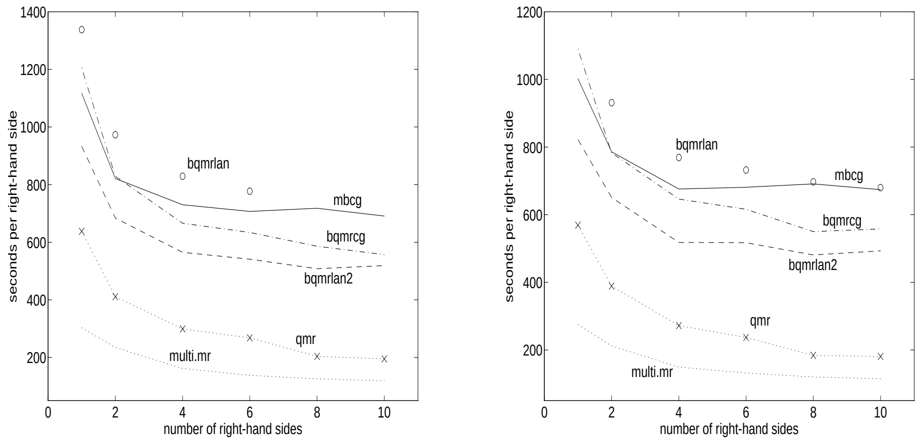

FIG. 4. *Time per right-hand side, in seconds, with matrix* RXS. *(a)* $B =$ RAND(n, s), *(b)* $B = I_{n,s}$. *Symbols:* '.-.' BQMRCG, '–' MBCG, 'o' BQMRLAN, '- -' BQMRLAN2, '...' MULTI.MR, '..x..' QMR.

Example 3. The matrix in this example is called YOUNG1C and was extracted from the class of complex symmetric matrices of the Harwell-Boeing collection [7]. All elements with nonzero imaginary part lie on the main diagonal.

As before, MULTI.MR has the best performance. Table 7 shows that block methods BQMRCG and BQMRLAN are sensitive to the choice of right-hand sides, and fail in several cases.

TABLE 7
Number of iterations to satisfy (16). matrix YOUNG1C $n = 814$. B_1=RAND(n, s), $B_2 = I_{n,s}$.

rhs	s / method	1	4	8	12	16	20	24	28	32	36	40
B_1	BQMRCG	458	178	98	70	55	45	37	33	28*	24*	22*
	MBCG	464	168	99	69	55	44	37	32	27	24	22
	BQMRLAN	464	-	-	-	-	-	-	-	27	-	-
	BQMRLAN2	458	168	93	69	53	44	37	32	27	24	22
	MULTI.MR	451	456	456	456	456	456	469	472	472	470	470
	QMR	461	492	496	496	498	498	498	498	498	498	501
B_2	BQMRCG	396	-	-	-	-	-	-	-	-	-	-
	MBCG	400	169	106	89	89	74	70	65	-	-	-
	BQMRLAN	402	170	98	84	75	63	60	55	-	-	-
	BQMRLAN2	397	166	99	85	80	66	59	55	-	-	-
	MULTI.MR	388	400	417	456	440	452	449	451	451	451	451
	QMR	397	406	429	429	429	429	429	429	429	429	429

* *ill-conditioned.*

More reliable results are obtained from BQMRLAN2, MBCG; even so, for B_2 and $s \geq 32$ all block methods fail. On the other hand, whenever block methods converge, their performance demonstrate satisfactory improvement as s increases, approaching that of MULTI.MR. We note that the failure of BQMRLAN was apparently due to loss of orthogonality. Comparing with the performance of QMR we conclude that once again, some form of information sharing is effective.

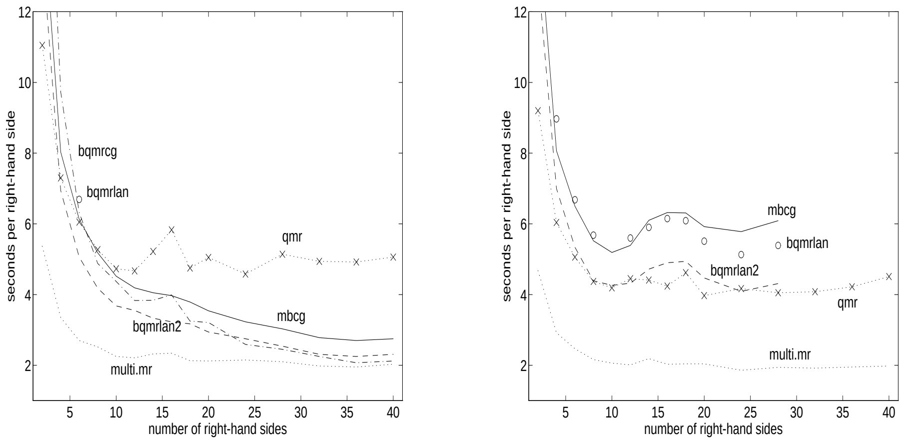

FIG. 5. *Time per right-hand side, in seconds, with matrix* YOUNG1C. *(a)* $B =$ RAND(n, s). *(b)* $B = I_{n,s}$, *Symbols: '.-.'* BQMRCG, *'–'* MBCG, *'o'* BQMRLAN, *'- -'* BQMRLAN2, *'...'* MULTI.MR, *'..x..'* QMR.

Example 4. The matrix in this example is called ELQ and originates from the discretization of the stochastic Liouville equation associated with slow-motional magnetic resonance line-shape simulations. The discretization is performed using a direct product basis set, consisting of a complete set of orthogonal idempotent spin operators and a set of orthogonal functions

[35]. In order to measure the parallelism of the codes, a different type of experiment was performed. We fixed the size of the problem, holding the number of right-hand sides to be equal to the maximum number of processors used, so $s = 12$ and let the number of processors vary from 1 to 12. For each method, figure 6(a) shows the time per right-hand side as p varies. Figure 6(b) shows the corresponding speedup $T(1)/T(p)$, where $T(p)$ is the time per right-hand side when p processors are used. Overall, block methods demonstrate a similar performance, with a maximum speed-up of 4. Methods MULTI.MR and QMR seem to profit more from additional processors reaching speed-ups of 7.5 and 6.8 respectively.

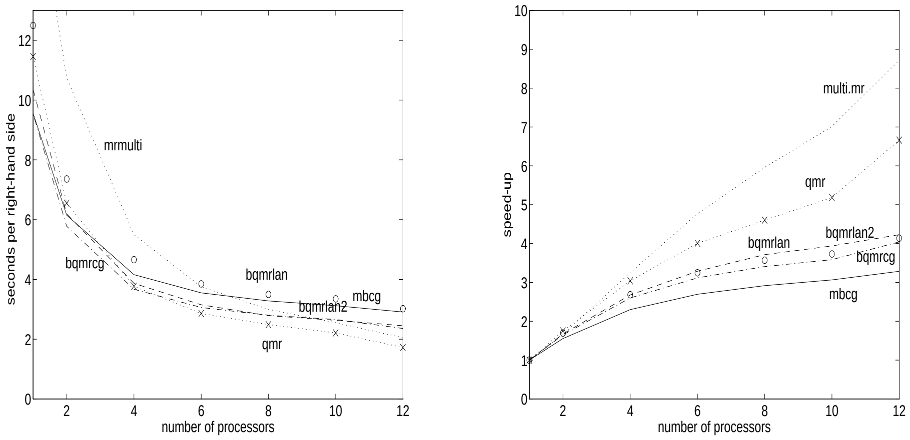

FIG. 6. *Parallel performance of methods for matrix* ELQ, $B = \text{RAND}(n, 12)$. *'.-.'* BQMRCG, *'–'* MBCG, *'o'* BQMRLAN, *'- -'*,BQMRLAN2, *'...'* MULTI.MR, *'..x..'* QMR. *(a) Time in seconds. (b) Speedup* $T(1)/T(p)$.

Example 5. The performance of block methods is strongly affected by the content of the starting residuals (B for our case of zero starting vectors) in eigenvectors of A [26][17][38]. In Theorem 3.2 we have shown that MULTI.MR too is sensitive to the eigenvector components found in B. This experiment is designed to illustrate these points. We use the diagonal matrix $D = \text{diag}(\text{RAND}(n, 1) + i\,\text{RAND}(n, 1))$, with $n = 1000$. Clearly all eigenvalues lie in the right-hand plane and a set of eigenvectors is $[e_1, \ldots, e_n]$. In order to achieve a better approximation of the invariant subspaces spanned by the right-hand sides, we increased the convergence tolerance in (16) to 10^{-12}. Four sets of right-hand sides are used. These are labeled B_a, B_b, B_c and B_d; each set consists of $s = 4$ right-hand sides. These sets were generated as follows:

$$B_a: \quad b_1 = \sum_{i=1}^{200} \rho_i e_i, \quad b_2 = \sum_{i=1}^{200} \rho_i e_i, \quad b_3 = \sum_{i=1}^{200} \rho_i e_i, \quad b_4 = \sum_{i=1}^{200} \rho_i e_i$$

$$B_b: \quad b_1 = \sum_{i=1}^{200} \rho_i e_i, \quad b_2 = \sum_{i=100}^{300} \rho_i e_i, \quad b_3 = \sum_{i=1}^{50} \rho_i e_i + \sum_{i=201}^{250} \rho_i e_i + \sum_{i=301}^{400} \rho_i e_i,$$

$$b_4 = \sum_{i=101}^{150} \rho_i e_i + \sum_{i=301}^{350} \rho_i e_i + \sum_{i=501}^{600} \rho_i e_i$$

$$B_c: \quad b_1 = \sum_{i=1}^{200} \rho_i e_i, \quad b_2 = \sum_{i=101}^{300} \rho_i e_i, \quad b_3 = \sum_{i=401}^{600} \rho_i e_i, \quad b_4 = \sum_{i=501}^{700} \rho_i e_i$$

TABLE 8
Number of iterations and time per right-hand side. Diagonal matrix of Example 5. $n = 1000$, $s = 4$.

type method	B_a	B_b	B_c	B_d
BQMRCG	48(2.92)	75(4.53)	83(4.96)	95(5.69)
MBCG	48(3.24)	75(5.03)	83(5.57)	95(6.36)
BQMRLAN	48 (3.63)	75 (5.65)	83(6.23)	96 (7.20)
BQMRLAN2	48(2.76)	74(4.24)	83(4.74)	95(5.45)
MRMULTI	90†(0.67)	124(0.96)	147(1.12)	169(1.25)

† all systems converge when seed system is solved

$$B_d : \quad b_1 = \sum_{i=1}^{200} \rho_i e_i, \quad b_2 = \sum_{i=201}^{400} \rho_i e_i, \quad b_3 = \sum_{i=401}^{600} \rho_i e_i, \quad b_4 = \sum_{i=601}^{800} \rho_i e_i$$

The scalars ρ_i are real random values. We observe that each right-hand side in B_a has nonzero projection to every basis vector in $\{e_1, \ldots, e_{200}\}$. In other words, the smallest invariant subspace for each of the columns in B_a is the span of $\{e_1, \ldots, e_{200}\}$. Sets B_b, B_c represent cases where the right-hand sides (hence their smallest invariant subspaces) have fewer common components. Finally, the smallest invariant subspaces for each of the right-hand sides in B_d are disjoint. Table 8 shows the number of iterations and, in parentheses, time per right-hand side. With B_a, all block methods reach their best performance of $\lceil n/s \rceil$ iterations to converge. As expected MULTI.MR demonstrates excellent performance; all systems were well approximated using the polynomial from the first seed. Block methods converge in the largest number of iterations with set B_d; however, the number of iterations remains smaller than the dimension (200) of the invariant subspace of each right-hand side. It is interesting to see MULTI.MR achieve good performance despite the structure of B_d. This was due to the fact that the residual polynomial is small in the region $[0, 1] \times [0, 1]$, causing a decrease of the norm of non-seed residuals; see Theorem 3.2. Although not shown here, we can also conclude that MULTI.MR will be faster than the complex symmetric minimal residual method applied to each system. We finally note that in light of the discussion in Section 4, the superior performance of MULTI.MR versus the block methods is expected; diagonal matrices have the maximum sparsity allowed in a nonsingular matrix.

6. Conclusions. We started with the objective to build methods that are based on short recurrences and that can solve complex symmetric systems with multiple right-hand sides faster than if a single right-hand side solver were to be applied independently to each right-hand side. We presented two approaches: (i) Block methods, which are analogues of CG-like single right-hand side solvers and are combined with a block quasi-minimal residual procedure of moderate cost, and (ii) a nonblock method, called MULTI.MR, based on the application of a seed residual polynomial on several right-hand sides. A study of the properties of the residual polynomial was undertaken and motivated the design of MULTI.MR. All methods were implemented on a shared-memory parallel machine and their performance compared with a single right-hand side solver (QMR).

For some application problems, block methods were found to perform well. However, algorithms BQMRCG and BQMRLAN stagnated in several cases; it is then worthwhile to use the modified algorithms MBCG and BQMRLAN2. For MBCG, the improved numerical properties were achieved at a substantial additional cost per iteration. Our analysis and experiments support the conclusion that block methods are appealing when applied to systems which are not very sparse, in particular to cases where the cost of sums and products of dense block matrices becomes comparable to that of multiplication with the coefficient matrix. The eigenvector composition of the starting vectors influences the performance of all methods. With a good monitoring strategy to avoid breakdown, MULTI.MR performs very well even for right-hand sides consisting of disjoint eigenvector components, provided distinct eigenvalues are clustered.

It thus seems that MULTI.MR provides a promising approach in the design of methods which perform better in general and at worst comparably with the application of a single right-hand side solver on each system. Finally, further research is warranted to improve the robustness of block methods for complex symmetric matrices as well as to achieve effective implementation on parallel architectures; see [17][25][38][36] for recent work in these directions.

Acknowledgments. D. Schneider provided us with much valuable information, references, and the matrix in Example 4. B. Philippe and R. Freund have made several useful suggestions. R. Mittra and K. Mahadevan provided us with the matrices in Examples 1 and 2. M-C. Brunet made comments on an earlier version of the manuscript. M. Levy provided editorial assistance. They all have our thanks.

A. Appendix.

A.1. Algorithm BQMRLAN2.

Algorithm 8.1 from [12] applies the quasi-minimal residual procedure to the Lanczos recurrence by first computing an incremental LU factorization of the associated tridiagonal matrix. This variant results in a more stable evaluation of recurrence terms than the original complex symmetric version of QMR. Algorithm BQMRLAN2 implements the block version of this method. Its computational cost was presented in Section 4. Function ORTH$(\hat{Y})$ orthogonalizes matrix $\hat{Y}$ with respect to $\langle x, y\rangle$, and function GIVENS applies Givens rotations as described in Appendix A.2.

Algorithm BQMRLAN2(A, B, X_0)
$\hat{V}_0 = B - AX_0, \quad P_0 = S_0 = 0_{n,s}, \quad G_{-1} = I_{2s}$
$[\rho_0^{-1}, V_0] = \mathrm{ORTH}(\hat{V}_0)$
$\omega_0 = \mathrm{diag}(\|v_0^{(1)}\|, \ldots, \|v_0^{(s)}\|)$
$\tau_0 = \omega_0\rho_0, \quad \epsilon_0 = I_s$
for $k = 0, 1, \ldots$

$P_{k+1} = V_k - P_k\epsilon_k^{-1}\rho_k^T$	$2ns^2 + ns + 8/3s^3$
$\epsilon_{k+1} = P_{k+1}^T A P_{k+1}$	$2ns^2$
$\hat{V}_{k+1} = AP_{k+1} - V_k\epsilon_{k+1}$	$2ns^2 + ns$
$[\rho_{k+1}^{-1}, V_{k+1}] = \mathrm{ORTH}(\hat{V}_{k+1})$	$2ns^2 + s^3$
$\omega_{k+1} = \mathrm{diag}(\|v_{k+1}^{(1)}\| \ldots, \|v_{k+1}^{(s)}\|)$	$2ns$
$\varsigma_k = 0_s, \quad \delta_k = \omega_k\epsilon_{k+1}, \quad \xi_k = \omega_{k+1}\rho_{k+1}$	$2s^2$
$[\varsigma_k, \delta_k]^T = G_{k-1}[\varsigma_k, \delta_k]^T$	$4s^3$

$[G_k, \tau_k, \tau_{k+1}, \delta_k] = \text{GIVENS}(\delta_k, \xi_k, \tau_k)$
$S_{k+1} = (P_{k+1} - S_k \varsigma_k)\delta_k^{-1}$ $\qquad 4ns^2 + ns + s^3$
$X_{k+1} = X_k + S_{k+1}\tau_k$ $\qquad 2ns^2 + ns$
end

As mentioned in Section 2.3, matrix S_k is updated by means of a two-term recurrence. On the other hand, matrices ς_k, δ_k, and τ_k are either full or triangular matrices, so that the BQMR procedure is slightly more expensive than for BQMRCG-like algorithms.

A.2. Function GIVENS**.** The block form of the function GIVENS was described in [42] and is repeated below. It carries out the operation

$$\begin{bmatrix} \delta \\ \xi \end{bmatrix} \longrightarrow \begin{bmatrix} \tilde{\delta} \\ 0_s \end{bmatrix} \qquad \delta, \tilde{\delta}, \xi \in \mathbb{C}^{s\times s}, \quad \tilde{\delta}, \xi \quad \text{upper triangular.}$$

At the same time, other iterates are updated, and the rotation coefficients stored.

function $[G, \tilde{\tau}, \tilde{\tilde{\tau}}, \tilde{\delta}] = \text{GIVENS}(\delta, \xi, \tau)$
$E = \begin{bmatrix} \delta \\ \xi \end{bmatrix} \quad T = \begin{bmatrix} \tau \\ 0_s \end{bmatrix}, \quad G = I_{2s}$
for $\;i = 1 : s$
  for $\;j = i + s : -1 : i + 1$
    $[\mathcal{C}, \mathcal{S}] = \text{ROT}(E_{j-1,i}, E_{j,i})$
    $y = \mathcal{C}E_{j-1,i:s} + \mathcal{S}E_{j,i:s}$
    $E_{j,i:s} = -\mathcal{S}^* E_{j-1,i:s} + \mathcal{C}E_{j,i:s}$
    $E_{j-1,i:s} = y$
    $y = \mathcal{C}T_{j-1,1:s} + \mathcal{S}T_{j,1:s}$
    $T_{j,1:s} = -\mathcal{S}^* T_{j-1,1:s} + \mathcal{C}T_{j,1:s}$
    $T_{j-1,1:s} = y$
    $y = \mathcal{C}G_{j-1,1:2s} + \mathcal{S}G_{j,1:2s}$
    $G_{j,1:2s} = -\mathcal{S}^* G_{j-1,1:2s} + \mathcal{C}G_{j,1:2s}$
    $G_{j-1,1:2s} = y$
  end
end
$\begin{bmatrix} \tilde{\tau} \\ \tilde{\tilde{\tau}} \end{bmatrix} = T, \quad \tilde{\delta} = E_{1:s,1:s}$

Function $\text{ROT}(x, y)$ computes the scalar Givens rotation coefficients to annihilate y. A total of s^2 Givens parameters are calculated. When both δ_k and ξ_k are diagonal, the routine can be significantly simplified, and only s Givens parameters are computed.

function $[G, \tilde{\tau}, \tilde{\tilde{\tau}}, \tilde{\delta}] = \text{GIVENS1}(\delta, \xi, \tau)$
for $i = 1 : s$
  $[\mathcal{C}, \mathcal{S}] = \text{ROT}(\delta_{i,i}, \xi_{i,i})$
  $\delta_{i,i} = \mathcal{C}\delta_{i,i} + \mathcal{S}\xi_{i,i}$
  $\tilde{\tilde{\tau}}_{i,i} = -\mathcal{S}^*\tau_{i,i}, \quad \tau_{i,i} = \mathcal{C}\tau_{i,i}$

$$G_{i,i} = \mathcal{C}, \quad G_{i+1,i+1} = \mathcal{C}, \quad G_{i+1,i} = -\mathcal{S}^*, \quad G_{i,i+1} = \mathcal{S}$$

end

$\tilde{\tau} = \tau, \quad \tilde{\delta} = \delta$

A cheaper implementation consists of storing the rotations parameters in vectors of length s^2 and s, for GIVENS and GIVENS1, respectively.

REFERENCES

[1] M. ARIOLI, I. DUFF, D. RUIZ, AND M. SADKANE, *Techniques for accelerating the block Cimmino method*, in Proc. Fifth SIAM Conf. Parallel Processing for Scientific Computing, J. J. Dongarra, K. Kennedy, P. Messina, D. C. Sorensen, and R. G. Voigt, eds., SIAM, Philadelphia, 1992, pp. 98–104.

[2] D. L. BOLEY AND G. H. GOLUB, *The Lanczos-Arnoldi algorithm and controllability*, Systems & Control Letters, 4 (1984), pp. 317–324.

[3] F. X. CANNING, *Physical and mathematical structure determine convergence rate of iterative techniques*, IEEE Trans. Magn., 25 (Jul. 1989), pp. 2825–2827.

[4] D. C. CHATFIELD, M. S. REEVES, D. G. TRUHLAR, C. DUNECZKY, AND D. W. SCHWENKE, *Complex generalized minimal residual algorithm for iterative solution of quantum mechanical reactive scattering equations*, J. Chem. Phys., 97 (1992), pp. 8322–8333.

[5] D. CHOUDHURY AND R. A. HORN, *An analog of the Gram-Schmidt algorithm for complex bilinear forms and diagonalization of complex symmetric matrices*, Tech. Report 454, The Johns Hopkins University, Jan. 1986.

[6] B. D. CRAVEN, *Complex symmetric matrices*, J. Austral. Math. Soc., 10 (1969), pp. 351–354.

[7] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *User's guide for the Harwell-Boeing sparse matrix collection (release I)*, Tech. Report TR/PA/92/86, CERFACS, Toulouse Cedex, France, Oct. 1992.

[8] V. FABER AND T. MANTEUFFEL, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal., 21 (April 1984), pp. 352–362.

[9] R. W. FREUND, *Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations*, in Numerical Methods in Approximation Theory, Vol. 9, D. Braess and L. L. Schumaker, eds., Birkhäuser Verlag, Basel, 1992, pp. 77–95.

[10] ——, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Stat. Comput., 13 (Jan. 1992), pp. 425–448.

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

[12] R. W. FREUND AND N. M. NACHTIGAL, *An implementation of the QMR method based on coupled two-term recurrences*, Tech. Report 92-15, RIACS, NASA Ames Research Center, Jun. 1992.

[13] ——, *Implementation details of the coupled QMR algorithm*, Tech. Report 92-19, RIACS, NASA Ames Research Center, Oct. 1992.

[14] G. GOLUB AND R. UNDERWOOD, *The block Lanczos method for computing eigenvalues*, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977, pp. 364–377.

[15] M. HOCHBRUCK AND G. STARKE, *Preconditioned Krylov subspace methods for Lyapunov matrix equations*, tech. report, Eidgenössische Technische Hochschule, Zürich, Oct. 1992. IPS Research Report 92-17.

[16] D. A. H. JACOBS, *A generalization of the conjugate-gradient method to solve complex systems*, IMA J. Numer. Anal., 6 (1986), pp. 447–452.

[17] Z. JIA, *Generalized block Lanczos methods for large unsymmetric eigenproblems.* Submitted for publication, 1993.

[18] P. JOLY AND G. MEURANT, *Complex conjugate gradient methods*, Numer. Alg., 4 (1993), pp. 379–406.

[19] 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.

[20] A. B. KUCHEROV, *A fast iterative method for solving in real arithmetic a system of linear equations with a complex symmetrix matrix*, Soviet Math. Dokl., 43 (1991), pp. 377–379.

[21] J. G. LEWIS, *Algorithms for sparse matrix eigenvalue problems*, Tech. Report STAN-CS-77-595, Dept. Comput. Sci., Stanford Univ., Palo Alto, CA, 1977.

[22] K. MAHADEVAN AND R. MITTRA, *Private communication*. Electromagnetic Communication Lab., University of Illinois at Urbana-Champaign, 1992.

[23] G. MARKHAM, *Conjugate gradient methods for indefinite, asymmetric, and complex systems*, IMA J. Numer. Anal., 10 (1990), pp. 155–170.

[24] E. K. MILLER, *Assessing the impact of large-scale computing on the size and complexity of first-principles electromagnetic models*, in Directions in Electromagnetic Wave Modeling, H. L. Bertoni and L. B. Felsen, eds., New York, 1991, Plenum Press, pp. 185–196.

[25] A. A. NIKISHIN AND A. Y. YEREMIN, *Variable block CG algorithms for solving large sparse symmetric positive definite systems on parallel computers, I: General iterative scheme*, Tech. Report EE-RR 1/92, Elegant Mathematics, Inc., 1992, Rev. Jan. 1993.

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

[27] ——, *Parallel implementation of the block conjugate gradient algorithm*, Parallel Comput., 5 (1987), pp. 127–139.

[28] M. PAPADRAKAKIS AND S. SMEROU, *A new implementation of the Lanczos method in linear problems*, Int'l. J. Numer. Meth. Engng., 29 (1990), pp. 141–159.

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

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

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

[32] M. SADKANE, *A block Arnoldi-Chebyshev method for computing the leading eigenpairs of large sparse unsymmetric matrices*, Numer. Math., 64 (1993), pp. 181–193.

[33] T. K. SARKAR, *On the application of the generalized biconjugate gradient method*, J. Electromagnetic Waves and Applications, 1 (1987), pp. 223–242.

[34] T. K. SARKAR, X. YANG, AND E. ARVAS, *A limited survey of various conjugate gradient methods for solving complex matrix equations arising in electromagnetic wave interactions*, Wave Motion, 10 (1988), pp. 527–546.

[35] D. J. SCHNEIDER AND J. H. FREED, *Calculating slow-motional magnetic resonance spectra: A user's guide*, in Biological Magnetic Resonance, J. J. Berliner and J. Reuben, eds., vol. 8, Plenum Press, 1989, pp. 1–76.

[36] H. D. SIMON AND A. YEREMIN, *A new approach to construction of efficient iterative schemes for massively parallel algorithms: Variable block CG and BiCG methods and variable block Arnoldi procedure*, in Proc. Sixth SIAM Conf. Parallel Processing for Scientific Computing, R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold, and D.A.Reed, eds., Philadelphia, March 1993, SIAM, pp. 57–60.

[37] V. SIMONCINI AND E. GALLOPOULOS, *An iterative method for nonsymmetric systems with multiple right-hand sides*, Tech. Report 1242, Center for Supercomputing Research and Development, July 1992.

[38] ——, *Convergence properties of block GMRES for solving systems for multiple right-hand sides*, Tech. Report 1316, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Oct. 1993.

[39] C. F. SMITH, A. F. PETERSON, AND R. MITTRA, *The biconjugate gradient method for electromagnetic scattering*, IEEE Trans. Ant. Prop., 38 (1990), pp. 938–940.

[40] ——, *A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields*, IEEE Trans. Ant. Prop., 37 (Nov. 1989), pp. 1490–1493.

[41] H. A. VAN DER VORST AND J. B. M. MELISSEN, *A Petrov-Galerkin type method for solving $Ax = b$ where A is symmetric complex*, IEEE Trans. Magn., 26 (1990), pp. 706–708.

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

[43] V. V. VOEVODIN, *On methods of conjugate direction*, USSR Comput. Maths. Math. Phys., 19 (1980), pp. 228–233.

[44] W. YANG AND W. H. MILLER, *Block Lanczos approach combined with matrix continued fraction for the*

S-matrix Kohn variational principle in quantum scattering, J. Chem. Phys., 6 (Sep. 1989), pp. 3504–3508.