Research Institute for Advanced Computer Science
NASA Ames Research Center

Conjugate Gradient Type Methods for Linear Systems With Complex Symmetric Coefficient Matrices

Roland Freund

December 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.54

NASA Cooperative Agreement NCC 2–387

Conjugate Gradient Type Methods for Linear Systems With Complex Symmetric Coefficient Matrices

Roland Freund

December 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.54

NASA Cooperative Agreement NCC 2–387

CONJUGATE GRADIENT TYPE METHODS FOR LINEAR SYSTEMS WITH COMPLEX SYMMETRIC COEFFICIENT MATRICES*

ROLAND FREUND†

Abstract. We consider conjugate gradient type methods for the solution of large sparse linear system $Ax = b$ with complex symmetric coefficient matrices $A = A^T$. Such linear systems arise in important applications, such as the numerical solution of the complex Helmholtz equation. Furthermore, most complex non-Hermitian linear systems which occur in practice are actually complex symmetric. We investigate conjugate gradient type iterations which are based on a variant of the nonsymmetric Lanczos algorithm for complex symmetric matrices. In particular, we propose a new approach with iterates defined by a quasi-minimal residual property. The resulting algorithm presents several advantages over the standard biconjugate gradient method. We also include some remarks on the obvious approach to general complex linear systems by solving equivalent real linear systems for the real and imaginary parts of x. Finally, numerical experiments for linear systems arising from the complex Helmholtz equation are reported.

Key words. complex symmetric matrices, nonsymmetric Lanczos algorithm, biconjugate gradients, minimal residual property

AMS(MOS) subject classifications. 65F10, 65N20

1. Introduction.

Conjugate gradient type methods — used in combination with preconditioning — are among the most effective iterative procedures for solving large sparse nonsingular systems of linear equations

$$Ax = b. \tag{1.1}$$

The archetype of these schemes is the classical conjugate gradient algorithm (CG hereafter) of Hestenes and Stiefel [20] for Hermitian positive definite matrices A.

While most linear systems which arise in practice have real coefficient matrices A and real right-hand sides b, there are some important applications (see [14] and the references therein) which lead to complex linear systems. Partial differential equations which model dissipative processes usually involve complex coefficient functions and/or complex boundary conditions (see e.g. [23]), and discretizing them yields linear systems with complex matrices A. A typical example for this category is the complex Helmholtz equation

$$-\Delta u - \sigma_1 u + i\sigma_2 u = f, \tag{1.2}$$

where σ_1, σ_2 are real coefficient functions, which describes the propagation of damped time-harmonic waves as e.g. electromagnetic waves in conducting media. Further applications, which give rise to complex linear systems, include discretizations of the time-dependent Schrödinger equation using implicit difference schemes, inverse

* This paper is based on a presentation at the Copper Mountain Conference on Iterative Methods, April 1–5, 1990, Copper Mountain, Colorado. This work was supported in part by DARPA via Cooperative Agreement NCC 2-387 between NASA and the Universities Space Research Association (USRA).

† RIACS, Mail Stop T045–1, NASA Ames Research Center, Moffett Field, CA 94035, USA, and Institut für Angewandte Mathematik und Statistik, Universität Würzburg, D–8700 Würzburg, Federal Republic of Germany.

scattering problems, underwater acoustics, eddy current computations [2], numerical computations in quantum chromodynamics, and numerical conformal mapping.

In all these examples, the resulting coefficient matrices A are non-Hermitian. However, they still exhibit special structures. Often, A differs from a Hermitian matrix only by a shift and a rotation:

$$A = e^{i\theta}(T + i\sigma I), \quad T = T^H \quad \text{Hermitian}, \quad \sigma \in \mathbf{C},\ \theta \in \mathbf{R}, \quad i := \sqrt{-1}. \tag{1.3}$$

In almost all other cases, which lead to complex systems, the coefficient matrix is symmetric:

$$A = A^T \quad \text{is complex symmetric.} \tag{1.4}$$

Note that the two families (1.3) and (1.4) overlap. The matrix (1.3) is complex symmetric iff T is real.

Surprisingly, when complex linear systems (1.1) are solved in practice, usually no attempt is made to exploit the special structures (1.3) or (1.4). Indeed, there are two popular approaches. The first one (see e.g. [1]) is to apply preconditioned CG to the Hermitian positive definite normal equations

$$A^H A x = A^H b. \tag{1.5}$$

Of course, complex numbers can always be avoided by rewriting (1.1) as a real linear system for the real and imaginary parts of x. The second popular approach is to solve this real and, in general, nonsymmetric linear system by one of the generalized CG methods such as GMRES [32]. It turns out that in both cases the resulting iterative schemes tend to converge slowly. As a consequence, complex linear systems have the bad reputation of being difficult to solve by CG type methods. On the other hand, for the class of shifted Hermitian matrices (1.3), there are efficient CG type algorithms [9, 10, 14] for complex linear systems in their original form (1.1). We refer the reader to [14] for a detailed study and practical aspects of these schemes. In [14] it is also shown how the special structure (1.3) can be preserved by using polynomial preconditioning.

In this paper, we are mainly concerned with CG type methods for linear systems (1.1) with coefficient matrices of the second class (1.4). In particular, we consider approaches based on a variant of the nonsymmetric Lanczos algorithm which was successfully used for computing eigenvalues of complex symmetric matrices (see [28] and [5, Chapter 6]). This Lanczos recursion generates basis vectors for the Krylov subspace induced by A which are orthogonal with respect to a certain indefinite inner product. The standard way to obtain from this basis iterates, which approximate the exact solution of (1.1), is to enforce a biconjugate gradient (BCG hereafter) condition. Here, we propose a new approach which generates iterates via a quasi-minimal residual property. On typical examples, the resulting algorithm displays better convergence properties than the BCG approach. In particular, it produces residuals whose norms are almost monotonically decreasing, in contrast to the erratic convergence behavior which is typical for BCG. Moreover, the new technique eliminates one of the two sources of possible breakdown in the BCG approach.

The outline of the paper is as follows. In Section 2, we discuss the Lanczos recursion for complex symmetric matrices and state some of its theoretical properties. Section 3 deals with a variant of the BCG algorithm for the special matrices (1.4).

In Section 4, we propose the quasi-minimal residual approach for complex symmetric matrices. Section 5 contains some remarks on the problem of breakdown of the complex symmetric Lanczos recursion. In Section 6, we are concerned with the issue "complex versus equivalent real linear systems". In particular, some results are presented which indicate that for Krylov subspace methods, such as CG type algorithms, it is always preferable to solve the original complex system rather than equivalent real ones. In Section 7, some typical results of numerical experiments for linear systems arising from finite difference approximations to the complex Helmholtz equation (1.2) are given. Finally, in Section 8, we make some concluding remarks.

Throughout the paper, all vectors and matrices, unless stated otherwise, are assumed to be complex. As usual, $\overline{M} = [\overline{m_{jk}}]$, $M^T = [m_{kj}]$, and $M^H = \overline{M}^T$ denote the complex conjugate, transpose, and Hermitian matrix, respectively, corresponding to the matrix $M = [m_{jk}]$. Moreover, we set $\mathrm{Re}\, M = (M + \overline{M})/2$ and $\mathrm{Im}\, M = (M - \overline{M})/(2i)$ for its real and imaginary part, respectively. The notation

$$K_k(c, B) := \mathrm{span}\{c, Bc, \ldots, B^{k-1}c\}$$

is used for the kth Krylov subspace of $\mathbf{C}^n$ generated by $c \in \mathbf{C}^n$ and the $n \times n$ matrix B. Furthermore, $\sigma(B)$ denotes the spectrum of B. The vector norm $\|x\| = \sqrt{x^H x}$ is always the Euclidean norm. The set of all complex polynomials of degree at most k is denoted by

$$\Pi_k := \{p(\lambda) \equiv \gamma_0 + \gamma_1\lambda + \cdots + \gamma_k\lambda^k \mid \gamma_0, \gamma_1, \ldots, \gamma_k \in \mathbf{C}\}.$$

Moreover, the coefficient matrix A of (1.1) is always $n \times n$ and, unless stated otherwise, assumed to be complex symmetric. Generally, $x_k \in \mathbf{C}^n$, $k = 0, 1, \ldots$, denote iterates for (1.1) with corresponding residual vectors $r_k := b - Ax_k$. If necessary, quantities of different algorithms will be distinguished by superscripts, e.g. x_k^{BCG} and x_k^{QMR}. Finally, an iterative scheme for solving (1.1) is called a Krylov subspace method if its iterates are of the form

(1.6) $\quad x_k \in x_0 + K_k(r_0, A)$ or, equivalently, $x_k = x_0 + P(A)r_0,\ P \in \Pi_{k-1}$.

Note that, in particular, CG type algorithms for (1.1) fall into this category.

2. The Lanczos algorithm for complex symmetric matrices. In this section, we are concerned with a complex symmetric variant of the Lanczos algorithm and its theoretical properties.

The Lanczos method [24] for general complex $n \times n$ matrices M is as follows (see e.g. [35, pp. 388–394]):

ALGORITHM 2.1 (NONSYMMETRIC LANCZOS METHOD).

(1) *Start:*
- *Choose* $r_0, s_0 \in \mathbf{C}^n$, $r_0, s_0 \neq 0$;
- *Set* $v = r_0$, $w = s_0$, *and* $v_0 = w_0 = 0$.

(2) *For* $k = 1, 2, \ldots$ *do:*
- *Compute* $\eta = w^T v$;
- *If* $\eta = 0$: *Stop;*
- *Otherwise, choose* $\beta_k, \gamma_k \in \mathbf{C}$ *with* $\beta_k\gamma_k = \eta$;
- *Set* $v_k = v/\gamma_k$ *and* $w_k = w/\beta_k$;
- *Compute* $\alpha_k = w_k^T M v_k$;
- *Set* $v = Mv_k - \alpha_k v_k - \beta_k v_{k-1}$;
- *Set* $w = M^T w_k - \alpha_k w_k - \gamma_k w_{k-1}$.

As Lanczos pointed out [26, p. 176], work and storage of Algorithm 2.1 can be halved if M is Hermitian resp. complex symmetric, by choosing starting vectors $s_0 = \bar{r}_0$ resp. $s_0 = r_0$. The resulting Hermitian Lanczos method has been studied extensively (see [18, Chapter 9] and the references therein). In contrast, the literature on the complex symmetric variant is scarce and restricted to the application of the algorithm to computing eigenvalues of complex symmetric matrices (see Moro and Freed [28] and Cullum and Willoughby [5, Chapter 6]). Here, we hope to convince the reader that the complex symmetric Lanczos algorithm is also very useful for solving linear systems.

The basic method is as follows:

ALGORITHM 2.2 (LANCZOS METHOD FOR $A = A^T$).

(1) *Start:*
- *Choose* $r_0 \in \mathbb{C}^n$, $r_0 \neq 0$;
- *Set* $\tilde{v}_1 = r_0$ *and* $v_0 = 0$.

(2) *For* $k = 1, 2, \ldots$ *do:*
- *Compute* $\beta_k = \left(\tilde{v}_k^T \tilde{v}_k\right)^{1/2}$;
- *If* $\beta_k = 0$: *Set* $m_0 = k - 1$, *and stop;*
- *Otherwise, set* $v_k = \tilde{v}_k / \beta_k$;
- *Compute* $\alpha_k = v_k^T A v_k$;
- *Set* $\tilde{v}_{k+1} = A v_k - \alpha_k v_k - \beta_k v_{k-1}$.

In the next proposition, some elementary properties of Algorithm 2.2. are listed; proofs can be found in [5, Chapter 6]. We set

$$V_k := [v_1 \quad v_2 \quad \ldots \quad v_k] \quad \text{and} \quad T_k := \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \ldots & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \ddots & \vdots \\ 0 & \beta_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_k \\ 0 & \ldots & 0 & \beta_k & \alpha_k \end{bmatrix}. \tag{2.1}$$

Moreover, $m_\star = m_\star(r_0, A) := \dim K_n(r_0, A)$ denotes the *grade* of r_0 with respect to A (cf. [35, p. 37]). We remark that $m_\star \geq 1$ is the smallest integer such that $K_{m_\star}$ is an A−invariant subspace of $\mathbb{C}^n$. Equivalently, if A is nonsingular and $r_0 = b - Ax_0$, $m_\star \geq 1$ is the smallest integer such that

$$A^{-1} b \in x_0 + K_{m_\star}(r_0, A). \tag{2.2}$$

PROPOSITION 2.3.

(a) In exact arithmetic, Algorithm 2.2 stops after a finite number of steps $k = m_0 + 1$ *and* $0 \leq m_0 \leq m_\star$. *Furthermore,* $\tilde{v}_{m_0+1} = 0$ *if* $m_0 = m_\star$ *("regular termination"), and* $\tilde{v}_{m_0+1} \neq 0$ *if* $m_0 < m_\star$ *("breakdown").*

(b) For $k = 1, 2, \ldots, m_0$:

$$v_k^T v_j = \begin{cases} 0, & \text{if } k \neq j \\ 1, & \text{if } k = j \end{cases}, \quad j = 1, 2, \ldots, m_0, \tag{2.3}$$

$$K_k(r_0, A) = \operatorname{span}\{v_1, v_2, \ldots, v_k\}, \tag{2.4}$$

$$A V_k = V_k T_k + [0 \quad 0 \quad \ldots \quad 0 \quad \tilde{v}_{k+1}]. \tag{2.5}$$

Notice that, by (2.3–4), the Lanczos vectors $v_1, \ldots, v_k$ form an orthonormal basis for $K_k(r_0, A)$ with respect to the (non-Hermitian) inner product

$$(x, y) := y^T x, \quad x, y \in \mathbf{C}^n. \tag{2.6}$$

In particular, if Algorithm 2.2 terminates regularly, it generates a basis of the affine space $x_0 + K_{m_*}(r_0, A)$ which, in view of (2.2), contains the exact solution of $Ax = b$.

Next, we remark that (2.6) is the proper (cf. Craven [4]) inner product for complex symmetric matrices. Unfortunately, it has the defect that there exist vectors $v \in \mathbf{C}^n$ which are *quasi-null* [4], i.e. $(v, v) = 0$, but $v \neq 0$. Consequently, it can not be excluded that Algorithm 2.2 actually breaks down. Indeed, in view of part (a) of Proposition 2.3, a breakdown occurs if one encounters a quasi-null vector $\tilde{v}_k$. The phenomenon of breakdown will be discussed further in Section 5.

We conclude this section with a result on the connection of the complex symmetric variant 2.2 with the general Lanczos Algorithm 2.1. Unlike Hermitian matrices, complex symmetric matrices do not have any special spectral properties. Indeed (see e.g. [21, Theorem 4.4.9]), any complex $n \times n$ matrix is similar to a complex symmetric matrix. This result entails that the general nonsymmetric Lanczos Algorithm 2.1 differs from the complex symmetric version 2.2 only in the additional starting vector s_0 which can be chosen independently of r_0 in 2.1. A strict statement of this correspondence is given in the following

THEOREM 2.4. *Let M be a complex $n \times n$ matrix and $r_0 \in \mathbf{C}^n$, $r_0 \neq 0$.*

(a) There exists a complex symmetric $n \times n$ matrix A which is similar to M:

$$M = XAX^{-1} \quad \textit{where } X \textit{ is nonsingular.} \tag{2.7}$$

(b) Set $\hat{r}_0 = X^{-1} r_0$ and $s_0 = X^{-T} \hat{r}_0$. Let $v_k, w_k, \alpha_k, \beta_k, \gamma_k$ resp. $\hat{v}_k, \hat{\alpha}_k, \hat{\beta}_k$ be the quantities generated by Algorithm 2.1 (starting with r_0, s_0) resp. Algorithm 2.2 (starting with $\hat{r}_0$). Let m_0 denote the termination index for Algorithm 2.2. Then, for $k = 1, 2, \ldots, m_0$:

$$\hat{v}_k = \Big(\prod_{j=1}^{k} \frac{\gamma_j}{\hat{\beta}_j}\Big) X^{-1} v_k = \Big(\prod_{j=1}^{k} \frac{\beta_j}{\hat{\beta}_j}\Big) X^T w_k, \quad \hat{\alpha}_k = \alpha_k, \quad (\hat{\beta}_k)^2 = \beta_k \gamma_k. \tag{2.8}$$

Proof. Only part (b) remains to be proved. First, by means of (2.7), we rewrite Algorithm 2.1 in terms of $A, X^{-1} v_k, X^T w_k$. By comparing the resulting iteration with Algorithm 2.2 and using induction on k, one readily verifies (2.8). □

After these preliminaries, we finally turn to linear systems (1.1) now. In the sequel, it is always assumed that A is nonsingular.

3. The biconjugate gradient algorithm for complex symmetric matrices. In his celebrated papers [24, 25], Lanczos also proposed a scheme, closely related to Algorithm 2.1, for solving general non-Hermitian linear systems, namely the biconjugate gradient algorithm (BCG). We refer the reader to [11, 31, 22] for a detailed discussion of the BCG approach.

Like Algorithm 2.1, BCG for general linear systems is started with two vectors: the residual $r_0 = b - Ax_0$ of the initial guess x_0 and a second vector $s_0 \neq 0$. We remark that s_0 is still unspecified. Due to the lack of a criterion for the choice of s_0, one usually sets $s_0 = r_0$ in practice. For the case of complex symmetric matrices A, it is straightforward to show that, in analogy to the complex symmetric variant 2.2

of the general Lanczos Algorithm 2.1, the choice $s_0 = r_0$ results in a scheme which requires only half the work and storage of general BCG. The resulting procedure is as follows:

ALGORITHM 3.1. *(BCG for $A = A^T$)*

(1) *Start:*
- *Choose $x_0 \in \mathbf{C}^n$;*
- *Set $p_0 = r_0 = b - Ax_0$ and compute $r_0^T r_0$.*

(2) *For $k = 1, 2, \ldots$ do:*
- *Compute Ap_{k-1} and $p_{k-1}^T Ap_{k-1}$;*
- *If $p_{k-1}^T Ap_{k-1} = 0$ or $r_{k-1}^T r_{k-1} = 0$: Set $m_1 = k-1$, and stop;*
- *Otherwise, set $\delta_k = r_{k-1}^T r_{k-1} / p_{k-1}^T Ap_{k-1}$;*
- *Compute $x_k = x_{k-1} + \delta_k p_{k-1}$ and $r_k = r_{k-1} - \delta_k Ap_{k-1}$;*
- *Compute $r_k^T r_k$ and set $\rho_k = r_k^T r_k / r_{k-1}^T r_{k-1}$;*
- *Compute $p_k = r_k + \rho_k p_{k-1}$.*

In the sequel, BCG always refers to the complex symmetric Algorithm 3.1. Next, we list some basic properties of BCG which follow readily from results (e.g. Jacobs [22]) for the general biconjugate gradient method.

PROPOSITION 3.2.

(a) In exact arithmetic, Algorithm 3.1 stops after a finite number of steps $k = m_1 + 1$ and $0 \le m_1 \le m_*$. Furthermore, $x_{m_1} = A^{-1}b$ if $m_1 = m_*$ ("regular termination"), and $x_{m_1} \neq A^{-1}b$ if $m_1 < m_*$ ("breakdown").

(b) For $k = 1, 2, \ldots, m_1$:

$$r_{k-1}^T r_{j-1} = 0, k \neq j, \quad j = 1, 2, \ldots, m_1, \tag{3.1}$$

$$K_k(r_0, A) = \operatorname{span}\{r_0, r_1, \ldots, r_{k-1}\}. \tag{3.2}$$

(c) Let $k \in \{1, 2, \ldots, m_1\}$. Then, x_k is uniquely determined by the Galerkin condition

$$(b - Ax_k)^T y = 0 \quad \text{for all} \quad y \in K_k(r_0, A), \quad x_k = x_0 + K_k(r_0, A), \tag{3.3}$$

with respect to the inner product (2.6).

By comparing (3.1–2) with (2.3–4), we conclude that r_{k-1} is parallel to the Lanczos vector v_k generated by Algorithm 2.2. More precisely, one easily verifies that

$$r_{k-1} = (-1)^{k-1} \delta_1 \cdots \delta_{k-1} \beta_1 \cdots \beta_{k-1} \beta_k v_k, \quad k = 1, 2, \ldots, m_1. \tag{3.4}$$

Notice that there are two different causes for breakdown of Algorithm 3.1. The first one, namely the occurrence of a quasi-null residual vector r_{k-1}, is, in view of (3.4), equivalent to the breakdown of the complex symmetric Lanczos Algorithm 2.2. In addition, Algorithm 3.1 breaks down if one encounters a search direction $p_{k-1} \neq 0$ with $p_{k-1}^T Ap_{k-1} = 0$. This second cause of breakdown is more severe than the first one. As we will see in Section 4, it occurs if no Galerkin iterate (3.3) exists.

Closely related to the biconjugate gradient method for general linear systems (1.1) is the conjugate gradients squared algorithm (CGS hereafter) which was recently proposed by Sonneveld [33].

ALGORITHM 3.3. *(CGS for general A)*

(1) *Start:*
- *Choose $x_0 \in \mathbf{C}^n$ and $s_0 \in \mathbf{C}^n$, $s_0 \neq 0$;*
- *Set $p_0 = u_0 = r_0 = b - Ax_0$ and compute $s_0^T r_0$.*

(2) *For* $k = 1, 2, \ldots$ *do:*

- *Compute* Ap_{k-1} *and* ${s_0}^T Ap_{k-1}$;
- *If* ${s_0}^T Ap_{k-1} = 0$ *or* ${s_0}^T r_{k-1} = 0$: *Stop;*
- *Otherwise, set* $\alpha_k = s_0^T r_{k-1} / s_0^T Ap_{k-1}$;
- *Compute* $q_k = u_{k-1} - \alpha_k Ap_{k-1}$;
- *Compute* $x_k = x_{k-1} + \alpha_k(u_{k-1} + q_k)$ *and* $r_k = r_{k-1} - \alpha_k A(u_{k-1} + q_k)$;
- *Compute* $s_0^T r_k$ *and set* $\beta_k = s_0^T r_k / s_0^T r_{k-1}$;
- *Compute* $u_k = r_k + \beta_k q_k$;
- *Compute* $p_k = u_k + \beta_k(q_k + \beta_k p_{k-1})$.

Notice that, like general BCG, CGS has a second unspecified starting vector s_0. However, unlike BCG, even with the special choice $s_0 = r_0$, CGS can not exploit the complex symmetry of A. In particular, for $A = A^T$, Algorithm 3.3 requires per iteration about twice as much work as the BCG Algorithm 3.1.

Finally, as a special case of the general connection [33] between the CGS and BCG approaches, we have the following

PROPOSITION 3.4. *Let* $A = A^T$, $r_0 = r_0^{BCG} = r_0^{CGS}$, *and, in Algorithm 3.3,* $s_0 = r_0$. *Then, for* $k = 0, 1, \ldots, m_1$,

$$r_k^{BCG} = p_k(A) r_0 \quad \textit{and} \quad r_k^{CGS} = \big(p_k(A)\big)^2 r_0$$

for some $p_k \in \Pi_k$ *with* $p_k(0) = 1$.

4. A quasi-minimal residual algorithm for complex symmetric matrices. In this section, we propose a new approach for solving complex symmetric linear systems. The method is based on the complex symmetric Lanczos Algorithm 2.2. For simplicity, we assume throughout this section that, in exact arithmetic, Algorithm 2.2 terminates regularly, i.e.

$$\beta_k \neq 0 \quad \text{for} \quad k = 1, 2, \ldots, m_\star, \quad \beta_{m_\star+1} = 0. \tag{4.1}$$

Moreover, let always be $k \in \{1, 2, \ldots, m_\star\}$ in the following.

4.1 Basic approach. Let x_k be the kth iterate of any Krylov subspace method (1.6). Then, by (2.4) and with V_k as defined in (2.1), we have

$$x_k = x_0 + V_k z_k \quad \text{where} \quad z_k \in \mathbf{C}^k. \tag{4.2}$$

Using (2.5) and $r_0 = \beta_1 v_1$, it follows from (4.2) that

$$r_k = b - Ax_k = r_0 - AV_k z = \beta_1 v_1 - V_{k+1} \tilde{T}_k z_k = V_{k+1}\big(\beta_1 e_1 - \tilde{T}_k z_k\big). \tag{4.3}$$

Here, $e_1 := [1 \;\; 0 \;\; \cdots \;\; 0]^T$ denotes the first unit vector,

$$\tilde{T}_k := \begin{bmatrix} T_k \\ \beta_{k+1} e_k^T \end{bmatrix} \quad \text{with} \quad e_k^T := [0 \;\; \cdots \;\; 0 \;\; 1], \tag{4.4}$$

and, if $k = m_\star$, $v_{m_\star+1} := 0$. Recall that T_k was defined in (2.1).

Clearly, the aim is to choose z_k in (4.2–3) such that $r_k \approx 0$ as good as possible. In the BCG approach, this is attempted by enforcing the Galerkin condition (3.3).

Using (2.3–4) and (4.3), one easily verifies that (3.3) holds iff r_k and v_{k+1} are parallel or, equivalently, z_k is a solution of the linear system

$$T_k z = \beta_1 e_1. \tag{4.5}$$

Note that, by (4.1), (4.5) is inconsistent if T_k is singular. Thus, we have the following

PROPOSITION 4.1. *A BCG iterate x_k^{BCG} satisfying the Galerkin condition (3.3) exists if, and only if, T_k is nonsingular. Moreover, if it exists, it is unique and given by*

$$x_k^{BCG} = x_0 + V_k z_k \quad \textit{and} \quad r_k = -\beta_{k+1}(z_k)_k v_{k+1} \tag{4.6}$$

where z_k is the solution of (4.5) and $(z_k)_k$ denotes its kth component.

Proposition 4.1 demonstrates the defects of the BCG approach. Simple examples show that singular T_k may indeed occur, and then, in view of Proposition 3.2, the BCG Algorithm 3.1 would break down in exact arithmetic. In floating-point arithmetic, such a breakdown is unlikely to happen. However, T_k may still be close to singular and then the Galerkin condition (3.3) defines a poor approximation to the true solution of (1.1). This is the reason for the typical erratic convergence behavior with wildly oscillating residual norms.

Obviously, the question arises whether there is a better strategy than (3.3) for choosing z_k in (4.2–3). Ideally, one would like to have the minimal residual (MR) property

$$\|b - Ax_k\| = \min_{x \in x_0 + K_k(r_0, A)} \|b - Ax\| = \min_{z \in \mathbf{C}^k} \|V_{k+1}(\beta_1 e_1 - \tilde{T}_k z_k)\|. \tag{4.7}$$

However, by (2.3), in general (see Theorem 4.4 for an exception) the columns of V_{k+1} are orthonormal only with respect to (2.6) rather than the Euclidean inner product $(x, y) = y^H x$. Consequently, solving the least-squares problem on the right-hand side of (4.7) results in an algorithm for which work and storage per iteration step k grows linearly with k. Hence, if one insists on a "true" iterative scheme with constant work and storage per iteration, this excludes the MR method.

Here, we propose the *quasi-minimal residual* (QMR) approach as a substitute for (4.7). Let

$$\Omega_{k+1} = \mathrm{diag}(\omega_1, \omega_2, \ldots, \omega_{k+1}) \quad \text{with} \quad \omega_j > 0 \quad \text{for all} \quad j$$

be a given positive diagonal weight matrix and rewrite (4.3) in the form

$$r_k = \left(V_{k+1}\Omega_{k+1}^{-1}\right)\left(\omega_1 \beta_1 e_1 - \Omega_{k+1}\tilde{T}_k z_k\right). \tag{4.8}$$

Instead of $\|r_k\|$ as in (4.7), one may at least minimize the vector of components in the representation (4.8) of r_k:

$$\min_{z \in \mathbf{C}^k} \|\omega_1 \beta_1 e_1 - \Omega_{k+1}\tilde{T}_k z\|. \tag{4.9}$$

Hence, we define the iterates of the QMR method as follows:

$$x_k = x_k^{QMR} = x_0 + V_k z_k \quad \text{where } z_k \in \mathbf{C}^k \text{ is the solution of (4.9).} \tag{4.10}$$

Notice that, $\Omega_{k+1}\tilde{T}_k$ is a $(k+1) \times k$ matrix which, by (4.4) and (4.1), has full rank. Thus, the least squares problem (4.9) always has a unique solution z_k.

Clearly, the QMR approach still depends on the weights ω_j. A natural choice is

$$\omega_j = \|v_j\|, \quad j = 1, 2, \ldots, k+1, \tag{4.11}$$

so that all basis vectors v_j/ω_j in the representation (4.8) of r_k have Euclidean length 1. Our numerical tests (cf. Section 7) also confirmed (4.11) as the best strategy.

4.2 Practical implementation. Next, we present an algorithm for the actual computation of the QMR iterates (4.10). The derivation is similar to that of Paige and Saunders' SYMMLQ and MINRES algorithms [29] for real symmetric matrices.

By (4.4), (2.1), and (4.1), $\Omega_{k+1}\tilde{T}_k$ is a tridiagonal $(k+1) \times k$ matrix with full column rank. Hence, it admits a QR factorization of the type

$$Q_{k+1}\Omega_{k+1}\tilde{T}_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix} \tag{4.12}$$

where Q_{k+1} is a unitary $(k+1) \times (k+1)$ matrix and R_k a nonsingular matrix of the form

$$R_k := \begin{bmatrix} \zeta_1 & \eta_2 & \theta_3 & 0 & \dots & 0 \\ 0 & \zeta_2 & \eta_3 & \ddots & \ddots & \vdots \\ 0 & \ddots & \zeta_3 & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \theta_k \\ \vdots & & & \ddots & \ddots & \eta_k \\ 0 & \dots & \dots & \dots & 0 & \zeta_k \end{bmatrix}. \tag{4.13}$$

The decomposition (4.12) can be generated by means of a series of k complex Givens rotations (e.g. [18, p. 47])

$$Q(c_j, s_j) = \begin{bmatrix} c_j & \overline{s}_j \\ -s_j & c_j \end{bmatrix}, \quad c_j \in \mathbf{R}, s_j \in \mathbf{C}, \quad c_j^2 + |s_j|^2 = 1, \quad j = 1, \dots, k.$$

In particular, (4.12) is easily updated from the factorization $Q_k\Omega_k\tilde{T}_{k-1} = R_{k-1}$ of the previous step by setting

$$Q_{k+1} = \begin{bmatrix} I_{k-1} & 0 \\ 0 & Q(c_k, s_k) \end{bmatrix} \begin{bmatrix} Q_k & 0 \\ 0 & 1 \end{bmatrix} \tag{4.14}$$

and computing c_k, s_k and the new elements θ_k, η_k, ζ_k of R_k as follows:

$$\begin{aligned} &\theta_k = \overline{s_{k-2}}\omega_{k-1}\beta_k, \quad \eta_k = c_{k-1}c_{k-2}\omega_{k-1}\beta_k + \overline{s_{k-1}}\omega_k\alpha_k, \\ &\tilde{\zeta}_k = c_{k-1}\omega_k\alpha_k - s_{k-1}c_{k-2}\omega_{k-1}\beta_k, \quad |\zeta_k| = \left(|\tilde{\zeta}_k|^2 + \omega_{k+1}^2|\beta_{k+1}|^2\right)^{1/2}, \\ &\zeta_k = \begin{cases} |\zeta_k|\tilde{\zeta}_k/|\tilde{\zeta}_k|, & \text{if } \tilde{\zeta}_k \neq 0 \\ |\zeta_k|, & \text{if } \tilde{\zeta}_k = 0, \end{cases} \quad c_k = \tilde{\zeta}_k/\zeta_k, \quad s_k = \omega_{k+1}\beta_{k+1}/\zeta_k. \end{aligned} \tag{4.15}$$

By (4.12) and since Q_{k+1} is unitary, (4.9) is equivalent to

$$\min_{z \in \mathbf{C}^k} \left\| \omega_1\beta_1 Q_{k+1}e_1 - \begin{bmatrix} R_k \\ 0 \end{bmatrix} z \right\|. \tag{4.16}$$

From (4.10) and (4.16), it follows that

$$x_k = x_0 + V_k z_k \quad \text{where} \quad z_k := R_k^{-1}t_k, \quad \begin{bmatrix} t_k \\ \tilde{\tau}_{k+1} \end{bmatrix} = \tilde{t}_{k+1} := \omega_1\beta_1 Q_{k+1}e_1. \tag{4.17}$$

Notice that, in view of (4.14), t_k differs from the previous vector t_{k-1} only by its kth component $\tau_k := (t_k)_k = c_k\tilde{\tau}_k$. Next, we define vectors p_j via

$$[\,p_1 \quad p_2 \quad \cdots \quad p_k\,] := V_k R_k^{-1}. \tag{4.18}$$

Finally, using (4.17–18) and (4.13), one obtains the recursion

$$x_k = x_{k-1} + \tau_k p_k, \quad \text{where} \quad p_k = \frac{1}{\zeta_k}\Big(v_k - \eta_k p_{k-1} - \theta_k p_{k-2}\Big),$$

for the QMR iterates. In combination with Algorithm 2.2, the following implementation results:

ALGORITHM 4.2 (QMR METHOD).

(1) *Start:*
- *Choose* $x_0 \in \mathbb{C}^n$;
- *Set* $\tilde{v}_1 = b - Ax_0$, $v_0 = p_0 = p_{-1} = 0$;
- *Set* $\beta_1 = \big(\tilde{v}_1^T\tilde{v}_1\big)^{1/2}$, $\tilde{\tau}_1 = \omega_1\beta_1$, $c_0 = c_{-1} = 1$, *and* $s_0 = s_{-1} = 0$.

(2) *For* $k = 1, 2, \ldots$ *do:*
- *If* $\beta_k = 0$, *stop:* x_{k-1} *solves* $Ax = b$.
- *Otherwise, compute* $v_k = \tilde{v}_k/\beta_k$ *and* $\alpha_k = v_k^T A v_k$;
- *Set* $\tilde{v}_{k+1} = Av_k - \alpha_k v_k - \beta_k v_{k-1}$, $\beta_{k+1} = \big(\tilde{v}_{k+1}^T\tilde{v}_{k+1}\big)^{1/2}$;
- *Compute* θ_k, η_k, ζ_k, c_k, *and* s_k, *using formulas (4.15);*
- *Set* $p_k = (v_k - \eta_k p_{k-1} - \theta_k p_{k-2})/\zeta_k$;
- *Set* $\tau_k = c_k\tilde{\tau}_k$, $\tilde{\tau}_{k+1} = -s_k\tilde{\tau}_k$;
- *Compute* $x_k = x_{k-1} + \tau_k p_k$.

The assumption (4.1) guarantees that, in exact arithmetic, Algorithm 4.2 stops for $k = m_* + 1$ and, by (2.2), x_{k-1} is indeed the solution of (1.1) then. However, in floating-point arithmetic, this finite termination property of the Lanczos recursion is no longer valid, and the stopping criterion stated in Algorithm 4.2 is not useful in practice. Instead, one should terminate the iteration as soon as $\|r_k\|$ is sufficiently reduced. We remark that r_k is not directly available in Algorithm 4.2. However, in view of (4.19), if one updates one additional auxiliary vector, namely

$$h_k = h_{k-1} + \frac{c_k\tilde{\tau}_{k+1}}{|s_1 s_2 \cdots s_k|^2\omega_{k+1}} v_{k+1}, \quad h_0 := r_0,$$

then $\|r_k\|$ can be computed via

$$\|r_k\| = |s_1 s_2 \cdots s_k|^2 \cdot \|h_k\|.$$

Finally, notice that, for the weighting strategy (4.11),

$$\|v_k\| = \frac{\sqrt{s^T s + t^T t}}{|\beta_k|}, \quad s := \operatorname{Re}\tilde{v}_k,\ t := \operatorname{Im}\tilde{v}_k,$$

can be obtained without extra cost during the computation of $\tilde{v}_k^T\tilde{v}_k = s^T s - t^T t + 2is^T t$.

4.3 Properties. In this subsection, we list some further properties of the QMR approach.

PROPOSITION 4.3. *For $k = 1, 2, \ldots, m_*$:*

(a)

$$r_k^{QMR} = |s_k|^2 r_{k-1}^{QMR} + (c_k \tilde{\tau}_{k+1}/\omega_{k+1}) v_{k+1}; \tag{4.19}$$

(b) The BCG iterate x_k^{BCG} defined by (3.3) exists if, and only if, $c_k \neq 0$. Moreover, if $c_k \neq 0$, then

$$x_k^{BCG} = x_{k-1}^{QMR} + (\tilde{\tau}_k/c_k) p_k, \tag{4.20}$$

$$\|r_k^{BCG}\| = |\omega_1 \beta_1 s_1 s_2 \cdots s_{k-1} s_k| \cdot \|v_{k+1}\|/(\omega_{k+1}|c_k|). \tag{4.21}$$

Proof. (a) By (4.17), (4.12), and (4.8), we have

$$r_k^{QMR} = \tilde{\tau}_{k+1} \tilde{w}_{k+1} \quad \text{where} \quad \tilde{w}_{k+1} := V_{k+1} \Omega_{k+1}^{-1} Q_{k+1}^H \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{4.22}$$

With (4.14), it follows that successive vectors $\tilde{w}_{k+1}$ and $\tilde{w}_k$ are connected by

$$\tilde{w}_{k+1} = -\overline{s_k} \tilde{w}_k + (c_k/\omega_{k+1}) v_{k+1}. \tag{4.23}$$

Finally, by combining (4.23) and (4.22) and using $\tilde{\tau}_{k+1} = -s_k \tilde{\tau}_k$, one obtains (4.19).

(b) First, we note that (4.12), (4.4), and (4.14) imply

$$Q_k \Omega_k T_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & c_k \end{bmatrix} R_k. \tag{4.24}$$

Thus, by Proposition 4.1, x_k^{BCG} exists iff $c_k \neq 0$. Now assume $c_k \neq 0$. Using (4.5–6), (4.24), and (4.17), we get

$$x_k^{BCG} = x_0 + V_k z_k^{BCG} \quad \text{where} \quad z_k^{BCG} = R_k^{-1} \begin{bmatrix} t_{k-1} \\ \tilde{\tau}_k/c_k \end{bmatrix}. \tag{4.25}$$

By comparing (4.25) with (4.17), (4.20) follows. For the proof of (4.21), notice that, by (4.25), (4.13) , and the formula for s_k in (4.15),

$$\beta_{k+1}(z_k^{BCG})_k = \beta_{k+1} \tilde{\tau}_k/(\zeta_k c_k) = \tilde{\tau}_k s_k/(\omega_{k+1} c_k). \tag{4.26}$$

Furthermore, Algorithm 4.2 shows that

$$|\tilde{\tau}_k| = |\omega_1 \beta_1 s_1 s_2 \cdots s_{k-1}|. \tag{4.27}$$

Finally, by inserting (4.26-27) into the formula (4.6) for r_k^{BCG}, we arrive at (4.21). □

In view of part (b) of Proposition 4.3, the QMR method has the additional feature that it also yields *all* existing BCG iterates. This is in contrast to the BCG Algorithm 3.1 which breaks down as soon as the *first* nonexisting BCG iterate is encountered. We

remark that, by (4.21), $\|r_k^{BCG}\|$ can be computed without extra cost from quantities which are generated in Algorithm 4.2 anyway. In particular, one may monitor $\|r_k^{BCG}\|$ during the course of the QMR algorithm, and, whenever the actual BCG iterate is desired, compute x_k^{BCG} via (4.20).

There is an important special case for which the QMR method (with weighting strategy (4.11)) is even equivalent to the MR approach (4.7). Consider the subclass of (1.3) of complex symmetric matrices of the form

$$A = T + i\sigma I, \quad T = T^T \text{ real symmetric}, \quad \sigma > 0. \tag{4.28}$$

Assume that $r_0 \in \mathbf{R}^n$ (this can always be achieved by a proper choice of x_0). Then, it is easily verified that the Lanczos vectors v_k generated by Algorithm 2.2 are all real and therefore, by (2.3), orthonormal with respect to the usual Euclidean inner product. In particular, by (4.11), $\omega_j \equiv 1$, and the least squares problem (4.9) is equivalent to the one on the right-hand side of (4.7). Hence we have the following

THEOREM 4.4. *Let A be of the form (4.28) and $r_0 \in \mathbf{R}^n$. Then, the iterates x_k generated by Algorithm 4.2 (with $\omega_j \equiv 1$) satisfy the minimal residual property (4.7).*

5. On the breakdown of the complex symmetric Lanczos algorithm. Recall that, throughout Section 4, possible breakdowns of the complex symmetric Lanczos recursion were explicitly excluded by assuming (4.1). In this section, we make some remarks about the general case and derive a theoretical result concerning so-called *incurable* breakdowns.

First, let us return to the nonsymmetric Lanczos Algorithm 2.1. It stops as soon as $w^T v = 0$ occurs. If this is caused by $v = 0$ or $w = 0$, then one has found an invariant subspace. Unfortunately, Algorithm 2.1 may also break down, i.e. stop with $w^T v = 0$ and $v, w \neq 0$ (see e.g. [35, p. 389]). Although this happens very rarely in practice, the possibility of such breakdowns has brought the nonsymmetric Lanczos method into discredit and, certainly, kept many people from actually using the algorithm. However, especially due to the efforts of Taylor [34], Draux [6], Parlett, Taylor, and Liu [30], and, most recently, Gutknecht [19], the phenomenon of breakdown is now well understood. Moreover, there are *look-ahead* [34, 30, 19] variants of the Lanczos algorithm which allow to leap — except in the very special case of an incurable breakdown [34] — over those iterations in which the standard algorithm would break down.

Here, we only sketch the basic idea of the look-ahead procedure for the special case of the complex symmetric Lanczos method. For a more detailed description of the look-ahead approach (for the general case) the reader is referred to [19]. Assume that breakdown occurs in Algorithm 2.2. In view of Proposition 2.3 this happens iff there is no complete set of m_* Lanczos vectors $v_k \in K_k(r_0, A)$, $k = 1, \ldots, m_*$, which are orthonormal (cf. (2.3)) with respect to the indefinite inner product (2.6). Clearly, there exists a maximal subset

$$\{k_1, k_2, \ldots, k_J\} \subsetneq \{1, 2, \ldots, m_*\}, \quad 1 \leq k_1 < k_2 < \cdots < k_J \leq m_*, \tag{5.1}$$

such that for each $j = 1, 2, \ldots, J$ there exists a vector $v_{k_j} \in K_{k_j}(r_0, A)$ satisfying the orthonormality relations

$$v_{k_j}^T v = 0 \quad \text{for all } v \in K_{k_j - 1}(r_0, A) \quad \text{and} \quad v_{k_j}^T v_{k_j} = 1. \tag{5.2}$$

By the definition of Krylov subspaces, $K_k(r_0, A) = \{P(A) r_0 \mid P \in \Pi_{k-1}\}$, and especially

$$v_{k_j} = P_{k_j - 1}(A) r_0 \quad \text{with} \quad P_{k_j - 1} \in \Pi_{k_j - 1}. \tag{5.3}$$

Therefore, we can rewrite (5.2) in terms of polynomials:

$$(P_{k_j-1}, P) = 0 \quad \text{for all } P \in \Pi_{k_j-2}, \quad (P_{k_j-1}, P_{k_j-1}) \neq 0, \tag{5.4}$$

with the indefinite inner product

$$(P, Q) := r_0^T P(A) Q(A) r_0. \tag{5.5}$$

A polynomial $P_{k_j-1} \in \Pi_{k_j-1}$ which fulfills (5.4) is called a regular orthogonal (with respect to (5.5)) polynomial of degree $k_j - 1$. It is well known [6, 19] that three successive regular orthogonal polynomials are connected via a three-term recurrence. By (5.3), it follows that there is a corresponding three-term recurrence relating the vectors v_{k_j-1}, v_{k_j}, and v_{k_j+1}. The look-ahead Lanczos procedure is a modification of Algorithm 2.2 which — based on this three-term relation — generates the vectors v_{k_j}, $j = 1, 2, \ldots, J$. These vectors can then be completed to a basis of K_{k_J} by setting, e.g.

$$v_k = A^{k-k_j} v_{k_j} \quad \text{for} \quad k = k_j + 1, k_j + 2, \ldots, k_{j+1} - 1, \; j = 0, 1, \ldots, J - 1.$$

(cf. [17]). Here, for $j = 0$, we set $k_0 := 1$. We remark that the resulting look-ahead Lanczos algorithm produces block tridiagonal matrices T_{k_j}, $j = 1, \ldots, J$, of the type (2.1) with $(k_j - k_{j-1}) \times (k_j - k_{j-1})$ matrices α_{k_j} on the block diagonal.

In exact arithmetic, the outlined algorithm terminates with the block tridiagonal T_{k_J}. Suppose that $k_J = m_\star$ in (5.1). Then T_{k_J} represents the restriction of the matrix A to the $A-$invariant subspace $K_{m_\star}(r_0, A)$. Obviously, in view of (2.2), the solution of (1.1) can then be computed from the quantities generated by the look-ahead Lanczos procedure. On the other hand, if $k_J < m_\star$ in (5.1), it is not possible to obtain the solution of (1.1) by means of the Lanczos process. For this reason, the case $k_J < m_\star$ is called incurable breakdown.

Next, we derive a criterion for the occurrence of incurable breakdown in the complex symmetric Lanczos algorithm. In the following, it is assumed that A is diagonalizable. Then (e.g. [21, Theorem 4.4.13]), A has a complete set of orthonormal (with respect to (2.6)) eigenvectors. In particular, r_0 can be expanded into eigenvectors of A. More precisely, by collecting components corresponding to identical eigenvalues, we get

$$r_0 = \sum_{l=1}^{m_\star} \rho_l u_l$$
$$\text{where } \rho_l \neq 0, \; Au_l = \lambda_l u_l, \text{ and, if } l \neq j, \; \lambda_l \neq \lambda_j, \; u_l^T u_j = 0. \tag{5.6}$$

Notice that, unless all eigenvalues of A are distinct, quasi-null vectors u_l may occur in (5.7). In view of the following theorem, this is equivalent to incurable breakdown.

THEOREM 5.1. *Let $A = A^T$ be a diagonalizable $n \times n$ matrix and $r_0 \in \mathbb{C}^n$. Then, in (5.1), $k_J = m_\star$ if, and only if, the eigenvectors in the expansion (5.6) of r_0 satisfy*

$$u_l^T u_l \neq 0 \quad \textit{for all} \quad l = 1, \ldots, m_\star. \tag{5.7}$$

Proof. We need to show that (5.7) is equivalent to the existence of a regular orthogonal polynomial of degree $m_\star - 1$ with respect to the inner product (5.5). From the general theory of orthogonal polynomials, it is well known (e.g. [3, pp. 11–12])

that regular orthogonal polynomials of degree k exist iff the corresponding moment matrix $M_k := (\mu_{j+l})_{j,l=0,\ldots,k}$ is nonsingular. For the case of (5.5), by (5.6), we have

$$\mu_j := r_0^T A^j r_0 = \sum_{l=1}^{m_*} \rho_l^2 \lambda_l^j u_l^T u_l, \quad j = 0, 1, \ldots. \tag{5.8}$$

Note that moment matrices are in particular Hankel matrices. By applying Kronecker's Theorem on the rank of infinite Hankel matrices [16, pp. 204–207] to $M_\infty := (\mu_{j+l})_{j,l=0,1,\ldots}$, it follows that

$$\operatorname{rank} M_\infty = \operatorname{rank} M_k = \operatorname{rank} M_{m-1} = m \quad \text{for all} \quad k \geq m-1, \tag{5.9}$$

where m is the number of poles of the rational function

$$f(z) := \sum_{j=0}^{\infty} \frac{\mu_j}{z^{j+1}}.$$

Using (5.8) and $\sum_{j=0}^{\infty} \lambda_l^j / z^{j+1} \equiv 1/(z-\lambda_l)$, one obtains the following expansion of f:

$$f(z) = \sum_{l=1}^{m_*} \frac{\rho_l^2 u_l^T u_l}{z - \lambda_l} \quad \text{for all} \quad |z| > \max_{l=1,\ldots,m_*} |\lambda_l|. \tag{5.10}$$

In particular, by (5.10), $m \leq m_*$ with equality holding iff (5.7) holds true. Hence, in view of (5.9), M_{m_*-1} is nonsingular iff (5.7) is fulfilled. This concludes the proof. □

As mentioned, (5.7) is guaranteed if A has only simple eigenvalues. Thus we have the following

COROLLARY 5.2. *If $A = A^T$ is an $n \times n$ matrix with n distinct eigenvalues, then incurable breakdowns can not occur in the complex symmetric Lanczos Algorithm 2.2.*

6. Complex versus equivalent real linear systems. In this section, we study connections between (1.1) and its equivalent real versions. Unless stated otherwise, A is now assumed to be a general complex $n \times n$ matrix.

6.1. Equivalent real linear systems. By taking real and imaginary parts in (1.1), we can rewrite (1.1) as the real linear system

$$A_* \begin{bmatrix} \operatorname{Re} x \\ \operatorname{Im} x \end{bmatrix} = \begin{bmatrix} \operatorname{Re} b \\ \operatorname{Im} b \end{bmatrix}, \quad A_* := \begin{bmatrix} \operatorname{Re} A & -\operatorname{Im} A \\ \operatorname{Im} A & \operatorname{Re} A \end{bmatrix}. \tag{6.1}$$

A second real version of (1.1) is

$$A_{**} \begin{bmatrix} \operatorname{Re} x \\ -\operatorname{Im} x \end{bmatrix} = \begin{bmatrix} \operatorname{Re} b \\ \operatorname{Im} b \end{bmatrix}, \quad A_{**} := \begin{bmatrix} \operatorname{Re} A & \operatorname{Im} A \\ \operatorname{Im} A & -\operatorname{Re} A \end{bmatrix}. \tag{6.2}$$

Obviously, (6.1) and (6.2) are the only essentially different possibilities of rewriting (1.1) as a real $2n \times 2n$ system. Furthermore, note that A_* is nonsymmetric iff $A \neq A^H$ is non-Hermitian, whereas A_{**} is symmetric iff $A = A^T$. Hence, for complex symmetric linear systems the approach (6.2) appears to be especially attractive since it permits the use of simple CG type methods such as SYMMLQ and MINRES [29] for real symmetric matrices.

In the following proposition, we collect some simple spectral properties of $A_\star$ and $A_{\star\star}$.

PROPOSITION 6.1.

(a) Let $J = X^{-1}AX$ be the Jordan normal form of A. Then $A_\star$ has the Jordan normal form

$$\begin{bmatrix} J & 0 \\ 0 & \overline{J} \end{bmatrix} = X_\star^{-1} A_\star X_\star \quad \text{where} \quad X_\star := \frac{1}{\sqrt{2}} \begin{bmatrix} X & -i\overline{X} \\ -iX & \overline{X} \end{bmatrix}. \tag{6.3}$$

In particular,

$$\sigma(A_\star) = \sigma(A) \cup \overline{\sigma(A)}. \tag{6.4}$$

(b) The matrices $A_{\star\star}$ and $-A_{\star\star}$ are similar. In particular,

$$-\lambda, \overline{\lambda}, -\overline{\lambda} \in \sigma(A_{\star\star}) \quad \text{for all} \quad \lambda \in \sigma(A_{\star\star}). \tag{6.5}$$

Moreover,

$$\sigma(A_{\star\star}) = \{\lambda \in \mathbb{C} \mid \lambda^2 \in \sigma(\overline{A}A)\}.$$

(c) Let $A = A^T$ be complex symmetric. Then, there exists a singular value decomposition (the so-called Takagi SVD) of A of the form

$$A = U\Sigma U^T, \quad U \text{ unitary}, \quad \Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) \geq 0. \tag{6.6}$$

Moreover, $A_{\star\star}$ is a real symmetric matrix with spectral decomposition

$$A_{\star\star} = \begin{bmatrix} Y & -Z \\ Z & Y \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix} \begin{bmatrix} Y & -Z \\ Z & Y \end{bmatrix}^T \quad \text{where} \quad Y = \operatorname{Re} U, \; Z = \operatorname{Im} U. \tag{6.7}$$

Proof. (a) First, note that

$$X_\star = S \begin{bmatrix} X & 0 \\ 0 & \overline{X} \end{bmatrix} \quad \text{where} \quad S := \frac{1}{\sqrt{2}} \begin{bmatrix} I_n & -iI_n \\ -iI_n & I_n \end{bmatrix} \quad \text{is unitary.} \tag{6.8}$$

In particular, (6.8) shows that with X also $X_\star$ is nonsingular. One readily verifies that

$$S^H A_\star S = \begin{bmatrix} A & 0 \\ 0 & \overline{A} \end{bmatrix},$$

and, in view of (6.8), this implies (6.3). (6.4) is an obvious consequence of (6.3).

(b) Since

$$\begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}^{-1} A_{\star\star} \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} = -A_{\star\star},$$

the real matrices $A_{\star\star}$ and $-A_{\star\star}$ are similar. Hence, (6.5) holds true. The relation between $\sigma(A_{\star\star})$ and $\sigma(\overline{A}A)$ is known (see [21, p. 214] for a proof).

(c) (6.6) is the well-known Takagi singular value decomposition for symmetric matrices (e.g. [21, Corollary 4.4.4]). By rewriting (6.6) in terms of the real and imaginary parts of A and U, one obtains (6.7) (cf. [21, pp. 212–213]). □

Roughly speaking, Krylov subspace methods are most effective for coefficient matrices A whose spectrum, except for possibly a few isolated eigenvalues, is contained

in a half-plane which excludes the origin of the complex plane. On the other hand, if this half-plane condition is not satisfied and if a large number of eigenvalues of A straddle the origin, usually the convergence of CG type algorithms is prohibitively slow. Typically, in these situations (see [7, 12, 13] for examples), iterations based on Krylov subspaces generated by A offer no advantage over solving the normal equations (1.5) by standard CG. See Theorem 6.4 for a theoretical result along these lines.

For complex linear systems which arise in practice the half-plane condition is usually satisfied. Indeed, mostly

$$\sigma(A) \subset \{\lambda \in \mathbf{C} \mid \operatorname{Im}\lambda \geq 0\}. \tag{6.9}$$

However, by rewriting (1.1) as real linear systems (6.1) resp. (6.2), one deliberately creates coefficient matrices whose spectra are most unfavorable for Krylov subspace methods. The case (6.2) is especially bad since, in view of (6.5), $\sigma(A_{**})$ is symmetric with respect to real and imaginary axis and hence the eigenvalues always embrace the origin. Similarly, by (6.4), the coefficient matrix A_* of (6.1) in general has eigenvalues in the upper as well as in the lower half-plane. In particular, if (6.9) holds and, as in most applications, the Hermitian part $(A + A^H)/2$ of A is indefinite, the spectrum of A_* straddles the origin and the half-plane condition is not satisfied for A_*. The following example illustrates this behavior.

Example 6.2. Consider the class (4.28) of complex symmetric matrices $A = T + i\sigma I$ where $T = T^T$ is real and $\sigma > 0$. Obviously,

$$\begin{aligned} \sigma(A) &= \{\lambda = \mu + i\sigma \mid \mu \in \sigma(T)\} \\ &\subset S := [\mu_m + i\sigma, \mu_M + i\sigma]. \end{aligned} \tag{6.10}$$

Here μ_m and μ_M denote the smallest and largest eigenvalue of T, respectively. Note that the complex line segment S is parallel to the real axis and always contained in the upper half of the complex plane. In view of (6.4), (6.10) implies

$$\sigma(A_*) = \{\lambda = \mu \pm i\sigma \mid \mu \in \sigma(T)\} \subset S \cup \overline{S}.$$

We remark that $S \cup \overline{S}$ is a tandem slit consisting of the two complex intervals S and $\overline{S}$ which are parallel and symmetric to each other with respect to the real axis. Moreover, the eigenvalues of A_* straddle the origin, if the Hermitian part T of A is indefinite. Finally, using (4.28) and part (b) of Proposition 6.1, we obtain

$$\begin{aligned} \sigma(A_{**}) &= \{\lambda = \pm\sqrt{\mu^2 + \sigma^2} \mid \mu \in \sigma(T)\} \\ &\subset \left[-\sqrt{\mu_M^2 + \sigma^2}, -\sigma\right] \cup \left[\sigma, \sqrt{\mu_M^2 + \sigma^2}\right]. \end{aligned}$$

Note that the class (4.28) is closely related to shifted skewsymmetric matrices. Indeed, if, instead of $Ax = b$, we rewrite $-iAx = -ib$ as a real system (6.1), one obtains

$$(-iA)_* = \begin{bmatrix} \sigma I_n & T \\ -T & \sigma I_n \end{bmatrix} = \sigma I_{2n} - N, \quad N := \begin{bmatrix} 0 & -T \\ T & 0 \end{bmatrix} \quad \left(= -N^T\right). \tag{6.11}$$

Then, the eigenvalues are contained in a line segment which is parallel to the imaginary axis and symmetric with respect to the real axis:

$$\sigma\left((-iA)_*\right) = \{\lambda = \sigma \pm i\mu \mid \mu \in \sigma(T)\} \subset [\sigma - i\rho, \sigma + i\rho], \quad \rho = \max\{|\mu_m|, |\mu_M|\}.$$

6.2. Correspondence of Krylov subspace methods. In analogy to (1.6) for complex linear systems (1.1), a Krylov subspace method for the solution of the equivalent real systems (6.1) resp. (6.2) generates iterates

$$\begin{bmatrix} \operatorname{Re} x_k \\ \operatorname{Im} x_k \end{bmatrix} = \begin{bmatrix} \operatorname{Re} x_0 \\ \operatorname{Im} x_0 \end{bmatrix} + P(A_*) \begin{bmatrix} \operatorname{Re} r_0 \\ \operatorname{Im} r_0 \end{bmatrix}, \quad P \in \Pi_{k-1}^{(r)}, \tag{6.12}$$

resp.

$$\begin{bmatrix} \operatorname{Re} x_k \\ -\operatorname{Im} x_k \end{bmatrix} = \begin{bmatrix} \operatorname{Re} x_0 \\ -\operatorname{Im} x_0 \end{bmatrix} + P(A_{**}) \begin{bmatrix} \operatorname{Re} r_0 \\ \operatorname{Im} r_0 \end{bmatrix}, \quad P \in \Pi_{k-1}^{(r)}. \tag{6.13}$$

Here and in the sequel, $\Pi_{k-1}^{(r)}$ denotes the subset of Π_{k-1} of polynomials with real coefficients. Furthermore, the notation

$$K_k^{(r)}(c, B) := \{P(B)c \mid P \in \Pi_{k-1}^{(r)}\} \quad \left(\subset K_k(c, B)\right)$$

will be used.

At first glance, it might appear that Krylov subspace iterations (1.6) resp. (6.12–13) for the original complex systems resp. its equivalent real versions correspond to each other. However, as the following proposition shows this is not the case in general.

PROPOSITION 6.3. *Let $k \in \mathbb{N}$.*

(a) Let $P \in \Pi_{k-1}$. Then, $x_k = x_0 + P(A)r_0$ is equivalent to

$$\begin{bmatrix} \operatorname{Re} x_k \\ \operatorname{Im} x_k \end{bmatrix} = \begin{bmatrix} \operatorname{Re} x_0 \\ \operatorname{Im} x_0 \end{bmatrix} + P_1(A_*) \begin{bmatrix} \operatorname{Re} r_0 \\ \operatorname{Im} r_0 \end{bmatrix} + P_2(A_*) \begin{bmatrix} \operatorname{Im} r_0 \\ -\operatorname{Re} r_0 \end{bmatrix} \tag{6.14}$$

where $P = P_1 + iP_2$, $P_1, P_2 \in \Pi_{k-1}^{(r)}$.

(b) Let $P \in \Pi_{k-1}^{(r)}$. Then, (6.13) is equivalent to

$$x_k = \operatorname{Re} x_k + i \operatorname{Im} x_k = x_0 + R(\overline{A}A)\overline{r_0} + S(\overline{A}A)\overline{A}r_0 \tag{6.15}$$

where $R \in \Pi_{\lfloor (k-1)/2 \rfloor}^{(r)}$ and $S \in \Pi_{\lfloor (k-2)/2 \rfloor}^{(r)}$ are defined by $P(\lambda) \equiv R(\lambda^2) + \lambda S(\lambda^2)$.

Proof. First, we note that, for $j = 0, 1, \ldots$,

$$(A_*)^j = \begin{bmatrix} \operatorname{Re} A^j & -\operatorname{Im} A^j \\ \operatorname{Im} A^j & \operatorname{Re} A^j \end{bmatrix} \quad \text{and} \quad (A_{**})^{2j} = \begin{bmatrix} \operatorname{Re}(\overline{A}A)^j & \operatorname{Im}(\overline{A}A)^j \\ -\operatorname{Im}(\overline{A}A)^j & \operatorname{Re}(\overline{A}A)^j \end{bmatrix}, \tag{6.16}$$

as is easily verified by induction on j.

(a) Let γ_j and δ_j be the coefficients of the real polynomials P_1 and P_2, respectively. Then,

$$\begin{aligned} \operatorname{Re} P(A) &= \sum_{j=0}^{k-1} (\gamma_j \operatorname{Re} A^j - \delta_j \operatorname{Im} A^j) \\ \operatorname{Im} P(A) &= \sum_{j=0}^{k-1} (\gamma_j \operatorname{Im} A^j + \delta_j \operatorname{Re} A^j). \end{aligned} \tag{6.17}$$

By reformulating $x_k = x_0 + P(A)r_0$, by means of (6.17) and the first relation in (6.16), in terms of real and imaginary parts, one immediately obtains (6.14).
(b) A routine calculation, using the second identity in (6.16), shows that (6.13) can be rewritten as

$$\begin{bmatrix} \operatorname{Re} x_k \\ -\operatorname{Im} x_k \end{bmatrix} = \begin{bmatrix} \operatorname{Re} x_0 \\ -\operatorname{Im} x_0 \end{bmatrix} + \begin{bmatrix} \operatorname{Re}\{R(\overline{A}A)\overline{r}_0 + S(\overline{A}A)\overline{A}r_0\} \\ -\operatorname{Im}\{R(\overline{A}A)\overline{r}_0 + S(\overline{A}A)\overline{A}r_0\} \end{bmatrix}.$$

Hence (6.13) and (6.15) are equivalent. □

In view of part (a) of Proposition 6.3, the corresponding real equivalent of complex Krylov schemes (1.6) are iterations of the type (6.14) and not the obvious real Krylov subspace methods (6.12). Clearly, the actual choice of the polynomials in (1.6) resp. (6.12–13) is aimed at obtaining iterates which are — in a certain sense — best possible approximations to the exact solution of the corresponding linear system. By using schemes of the type (6.12), from the first, one gives up k of the $2k$ real parameters which are available for optimizing complex Krylov subspace methods (1.6.). Consequently, it is always preferable to solve the complex system (1.1) rather than the real version (6.1) by Krylov subspace methods. Furthermore, numerical tests reveal that the convergence behavior of the two approaches can be drastically different (see Section 7).

6.3. A connection between MR and CGNR for complex symmetric matrices. Now assume that A is a complex symmetric $n \times n$ matrix. Then, in view of part (c) of Proposition 6.1, A_{**} is a real symmetric indefinite matrix whose spectrum is given by

$$\sigma(A_{**}) = \{\pm\sigma_j \mid j = 1, \ldots, n\}. \tag{6.18}$$

Here $\sigma_j = \sigma_j(A) \geq 0$, $j = 1, \ldots, n$, denote the singular values of A.

Since there are simple extensions [29] of classical CG to real symmetric indefinite matrices, it is especially tempting to solve (6.2) by one of these methods. The iterates of these algorithms are defined either via a Galerkin condition or a minimal residual (MR) property. Here, we consider the MR approach. Applied to (6.2) it generates a sequence of iterates z_k, $k = 1, 2, \ldots$, which are characterized by

$$\|b_{**} - A_{**}z_k\| = \min_{z \in z_0 + K_k^{(r)}(r_0^{**}, A_{**})} \|b_{**} - A_{**}z\|, \quad z_k \in z_0 + K_k^{(r)}(r_0^{**}, A_{**}). \tag{6.19}$$

Here, we have set

$$b_{**} := \begin{bmatrix} \operatorname{Re} b \\ \operatorname{Im} b \end{bmatrix}, \quad z_k := \begin{bmatrix} \operatorname{Re} x_k \\ -\operatorname{Im} x_k \end{bmatrix} \text{ for } k = 0, 1, \ldots, \quad r_0^{**} := b_{**} - A_{**}z_0. \tag{6.20}$$

Roughly speaking, CG type algorithms for real symmetric indefinite systems converge slowly if the coefficient matrix is strongly indefinite, in the sense that it has many positive as well as many negative eigenvalues. Unfortunately, since, by (6.18), $\sigma(A_{**})$ is even symmetric to the origin, A_{**} exhibits this undesirable property. Indeed, numerical tests show that the convergence behavior of the MR method (6.19) is practically identical to that of the tabooed approach to (1.1) via solving the normal equations (1.5) by standard CG [20]. In the sequel, we refer to this latter method as CGNR. Notice that the iterates x_k of CGNR are defined by the minimization property

$$\|b - Ax_l\| = \min_{x \in x_0 + K_l(A^H r_0, A^H A)} \|b - Ax\|, \quad x_l \in x_0 + K_l(A^H r_0, A^H A). \tag{6.21}$$

Next, we prove that MR and CGNR are even equivalent, if the starting residual r_0^{**} satisfies a certain symmetry condition. Note that, corresponding to the spectral decomposition (6.7), r_0^{**} can be expanded into eigenvectors of A_{**} as follows:

$$r_0^{**} = \begin{bmatrix} Y & -Z \\ Z & Y \end{bmatrix} c \quad \text{with} \quad c = \begin{bmatrix} c_1 \\ \vdots \\ c_{2n} \end{bmatrix} \in \mathbf{R}^{2n}. \tag{6.22}$$

THEOREM 6.4. *Let x_k^{MR} resp. x_l^{CGNR} denote the iterates generated by (6.3.19–20) resp. (6.21) starting with the same initial guess $x_0 \in \mathbf{C}^n$. Assume that c in the expansion (6.22) of r_0^{**} satisfies*

$$|c_j| = |c_{n+j}|, \quad j = 1, 2, \ldots, n. \tag{6.23}$$

Then,

$$x_l^{CGNR} = x_{2l}^{MR} = x_{2l+1}^{MR}, \quad l = 0, 1, \ldots . \tag{6.24}$$

Proof. First, note that, in view of (6.7) and (6.22), c_j and c_{n+j} are components corresponding to a pair of symmetric eigenvalues $\pm\sigma_j$ of A_{**}. However, for any real symmetric linear system $A_{**}z = b_{**}$ with "symmetric" eigenvalues and "symmetric" starting residual r_0^{**} in the sense of (6.18) and (6.23), respectively, the MR method generates iterates with $z_k \in z_0 + K_{\lfloor k/2 \rfloor}^{(r)}(A_{**}r_0^{**}, A_{**}^2)$ (see e.g. [13]). Consequently, the iterates defined by (6.19) satisfy

$$z_{2l} = z_{2l+1} \in z_0 + K_l^{(r)}(A_{**}r_0^{**}, A_{**}^2). \tag{6.25}$$

In particular, by (6.20), (6.25) shows that $x_{2l}^{MR} = x_{2l+1}^{MR}$.

It remains to prove the first relation in (6.24). To this end, we remark that

$$\|b_{**} - A_{**}z\| = \|b - Ax\| \quad \text{for all} \quad z = \begin{bmatrix} \operatorname{Re} x \\ -\operatorname{Im} x \end{bmatrix}, \; x \in \mathbf{C}^n. \tag{6.26}$$

Moreover, by using (6.20) and part (b) of Proposition 6.3 (applied to polynomials $P(\lambda) \equiv \lambda S(\lambda^2)$), we deduce

$$z_0 + K_l^{(r)}\left(A_{**}r_0^{**}, (A_{**})^2\right) = \left\{ \begin{bmatrix} \operatorname{Re} x \\ -\operatorname{Im} x \end{bmatrix} \;\middle|\; x \in x_0 + K_l^{(r)}(A^H r_0, A^H A) \right\} \tag{6.27}$$

(notice that $\overline{A} = A^H$ in (6.15)!). In view of (6.25–27), (6.19) (for $k = 2l$) can be rewritten in the form

$$\|b - Ax_{2l}^{MR}\| = \min_{x \in x_0 + K_l^{(r)}(A^H r_0, A^H A)} \|b - Ax\|, \quad x_{2l}^{MR} \in x_0 + K_l^{(r)}(A^H r_0, A^H A). \tag{6.28}$$

Finally, note that the iterates of CGNR always correspond to real polynomials, i.e. $x_l^{CGNR} \in x_0 + K_l^{(r)}(A^H r_0, A^H A)$. Hence, by comparing (6.21) with (6.28), we conclude that $x_l^{CGNR} = x_{2l}^{MR}$. □

Clearly, the special symmetry condition (6.23) will not be satisfied in general. Nevertheless, all our numerical experiments showed (cf. Section 7) that (6.24) is still fulfilled approximately, i.e.

$$x_l^{CGNR} \approx_{2l}^{MR} \approx_{2l+1}^{MR}, \quad l = 0, 1, \ldots . \tag{6.29}$$

7. Numerical Examples. We have performed numerical experiments with all algorithms considered in this paper in numerous cases. In this section, we present a few typical results of these experiments.

Consider (1.2) on the unit square $G := (0,1) \times (0,1)$ with $\sigma_1 \in \mathbf{R}$ a constant and σ_2 a real coefficient function. First, assume that u satisfies Dirichlet boundary conditions. Then, approximating (1.2) by finite differences on a uniform $m \times m$ grid with mesh size $h := 1/(m+1)$ yields a linear system (1.1) with A an $n \times n$, $n := m^2$, matrix of the form

$$A = T + ihD, \quad T := A_0 - \sigma_1 h^2 I, \quad D = \mathrm{diag}(d_1, d_2, \ldots, d_n). \tag{7.1}$$

Here A_0 is the symmetric positive definite matrix arising from the usual five-point discretization of $-\Delta$ and the diagonal elements of D are just the values of σ_2 at the grid points.

Similarly, if we consider the real Helmholtz equation (1.2), i.e. $\sigma_2 \equiv 0$, but now with a typical complex boundary condition such as

$$\frac{\partial u}{\partial n} = i\alpha u \quad \text{on} \quad \{(1,y) \mid -1 < y < 1\}$$

and Dirichlet boundary conditions on the other three sides of the boundary of G, one again arrives at (7.1) where

$$d_j = \begin{cases} \alpha, & \text{if } j = lm,\ l = 1, \ldots, m, \\ 0, & \text{otherwise.} \end{cases} \tag{7.2}$$

The test problems presented in this section are all linear systems $Ax = b$ with complex symmetric coefficient matrices of the type (7.1). For Example 7.1, the mesh size $h = 1/64$ was chosen resulting in a 3969×3969 matrix A. In Examples 7.2–4, $h = 1/32$ and thus A is a 961×961 matrix. The right-hand side b was chosen to be a vector with random components in $[-1,1] + i[-1,1]$, with the exception of Example 7.2 where b had constant components $1 + i$. As starting vector $x_0 = 0$ was chosen.

As stopping criterion, we used

$$R_k := \frac{\|b - Ax_k\|}{\|b - Ax_0\|} \leq 10^{-6}. \tag{7.3}$$

In Figures 7.1–4, the relative residual norm (7.3), R_k, is plotted versus the iteration number k, at least for those methods for which work and storage per iteration is roughly the same. In the case of CGS resp. CGNR which both require about twice the work of the other algorithms and especially two matrix-vector products $A \cdot v$ resp. $A \cdot v$, $\overline{A} \cdot v$ per iteration, we have plotted R_k versus $2k$.

In a first series of experiments, QMR (with different weighting strategies) and BCG were compared. The natural choice (4.11) turned out to be the best strategy in all cases. In the following, QMR always refers to Algorithm 4.2 with weights (4.11). Then

QMR produces residual vectors whose norms are almost monotonically decreasing and generally smaller than those of the BCG residuals. However, convergence of QMR and BCG typically occurred after a comparable number of iterations. The following example is typical.

Example 7.1. Here, (7.1) is a 3969×3969 matrix with $\sigma_1 = 200$, and the diagonal elements of D are given by (7.2) with $\alpha = 10$. In Figure 7.1, the convergence behavior of BCG, QMR, and of the unweighted version ($\omega_j \equiv 1$) of the QMR Algorithm 4.2 is displayed.

Figure 7.1

Next, we compared the CGS Algorithm 3.3 with QMR and BCG. Typically, CGS needed slightly fewer iterations than QMR and BCG to reach (7.3). However, per iteration, QMR and BCG require only about half as much work and storage, CGS is not competitive for complex symmetric matrices.

Example 7.2. In (7.1), we set $n = 961$, $\sigma_1 = 100$ and d_j, $j = 1, \ldots, n$, are chosen as random numbers in $[0, 10]$. Figure 7.2 shows the convergence behavior of QMR, BCG, and two runs of CGS with different starting vectors s_0, namely $s_0 = r_0$ resp. s_0 with random components in $[-1, 1] + i[-1, 1]$.

Figure 7.2

Notice the extremely large residual norms in the early stage of the CGS iteration.

In the following two examples, we compared CG type methods for $Ax = b$ with real schemes for the equivalent real systems (6.1) resp. (6.2). For GMRES [32], work and storage per iteration step k grows linearly with k and in practice it is necessary to use restarts. In the sequel, GMRES(k_0) refers to GMRES applied to (6.1) and restarted after every k_0 iterations. Finally, MR($A_{\star\star}$) denotes the minimal residual method (6.19) applied to the real symmetric system (6.2).

Example 7.3. Here, in (7.1), $n = 961$, $\sigma_1 = 100$, and d_j are given by (7.2) with $\alpha = 100$. In Figure 7.3, for QMR, MR($A_{\star\star}$), GMRES(5) resp. CGNR, the relative residual norm (7.3) is plotted versus iteration number k resp. $2k$.

Figure 7.3

Notice that, although the symmetry condition (6.23) is not fulfilled, the curves for *CGNR* and MR($A_{\star\star}$) are almost identical. This confirms (6.29). Finally, we tried GMRES(k_0) also with other restart parameters k_0. For this example, the method did never converge.

Example 7.4. Let A be the 961×961 matrix (7.1) with $\sigma_1 = 1000$, $D = \sigma_2 I$, $\sigma_2 = 100$ and set $\sigma := \sigma_2 h^2$. Note that A is a shifted Hermitian matrix of the form (4.28) (cf. Example 6.2). In particular, A belongs to the class of matrices (1.3) for which efficient true minimal residual algorithms for solving $Ax = b$ exist. Here we used the particular implementation, MR(A), derived in [14, Algorithm 2]. Recall that, by rewriting $-iAx = -ib$ as a real system (6.1), one obtains a shifted skewsymmetric matrix (6.11), $(-iA)_\star$. Again, for such matrices an efficient true minimal residual

algorithm, denoted by MR($(-iA)_\star$), exists [8, 12]. Figure 7.4 shows the convergence behavior of MR(A), MR($A_{\star\star}$), MR($(-iA)_\star$),CGNR, and GMRES(20).

Figure 7.4

Notice that MR($(-iA)_\star$) and CGNR are nearly identical. This is typical for the case that σ is small compared to the spectral radius of T. Furthermore, if $\sigma = 0$, i.e. $(-iA)_\star$ in (6.11) is skewsymmetric, CGNR and MR($(-iA)_\star$) are even equivalent [12].

8. Concluding remarks. Complex linear systems $Ax = b$ which arise in practice often have complex symmetric coefficient matrices A. In this paper, we have explored the use of a variant of the nonsymmetric Lanczos process for complex symmetric matrices for the solution of such linear systems. In particular, we have proposed a new method of defining approximate solutions of $Ax = b$ via a quasi-minimal residual (QMR) property. In contrast to the biconjugate gradient (BCG) approach, the QMR iterates are well-defined as long as the basic Lanczos recursion does not break down. Moreover, unlike the wildly oscillating BCG residuals, the QMR residuals converge almost monotonically. Also, existing BCG iterates can be easily computed from the quantities generated during the QMR iteration. Finally, possible breakdowns — except incurable ones — of the complex symmetric Lanczos recursion can be overcome by using a look-ahead version of the Lanczos process. Incurable breakdowns only occur in very special situations. For example, they can not occur if all eigenvalues of A are distinct.

It is very tempting (and often done in practice!) to avoid complex linear system by solving equivalent real systems instead. We have presented some theoretical and numerical results which show that this — at least for Krylov subspace methods — is a fatal approach. Typically, the resulting real systems are unequally harder to solve by conjugate gradient type algorithms than the original complex ones.

In this paper, we have not addressed the question of how to choose preconditioners M for complex symmetric linear systems. This will be the subject of a forthcoming report. Here, we only remark that complex symmetry is preserved under preconditioning as long as M is complex symmetric. In particular, all algorithms for $A = A^T$ which we have considered can be used in conjunction with a complex symmetric preconditioner M. Note that the standard techniques, such as incomplete factorization [27], applied to $A = A^T$ generate complex symmetric preconditioners M.

Finally, we would like to mention that the quasi-minimal residual approach can also be used to stabilize the general nonsymmetric biconjugate gradient algorithm [15].

REFERENCES

[1] A. Bayliss and C.I. Goldstein, *An iterative method for the Helmholtz equation*, J. Comput. Phys., 49 (1983), pp. 443–457.

[2] C.S. Biddlecombe, E.A. Heighway, J. Simkin and C.W. Trowbridge, *Methods for eddy current computation in three dimensions*, IEEE Trans. Magnetics, MAG-18 (1982), pp. 492–497.

[3] T.S. Chihara, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.

[4] B.D. Craven, *Complex symmetric matrices*, J. Austral. Math. Soc., 10 (1969), pp. 341–354.

[5] J.K. Cullum and R.A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 1, Theory*, Birkhäuser, Basel, 1985.

[6] A. DRAUX, *Polynômes Orthogonaux Formels - Applications*, Lecture Notes in Mathematics, Volume 974, Springer-Verlag, Berlin, 1983.

[7] S.C. EISENSTAT, *Some observations on the generalized conjugate gradient method*, in Numerical Methods, Proceedings, Caracas 1982, V. Pereyra and A. Reinoza, eds., Lecture Notes in Mathematics, Volume 1005, Springer-Verlag, Berlin, 1983, pp. 99–107.

[8] 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.

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

[10] ———, *Orthogonal error methods*, SIAM J. Numer. Anal., 24 (1987), pp. 170–187.

[11] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Proceedings of the Dundee Conference on Numerical Analysis, 1975, G.A. Watson, ed., Lecture Notes in Mathematics, Volume 506, Springer-Verlag, Berlin, 1976, pp. 73–89.

[12] R. FREUND, *Über einige cg-ähnliche Verfahren zur Lösung linearer Gleichungssysteme*, Doctoral Thesis, Universität Würzburg, F.R. of Germany, May 1983.

[13] ———, *Pseudo Ritz values for indefinite Hermitian matrices*, Tech. Report 89.33, RIACS, NASA Ames Research Center, August 1989.

[14] ———, *On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices*, Numer. Math., 57 (1990), pp. 285–312.

[15] ———, *Towards stable implementations of biconjugate gradient type methods*, Manuscript in preparation.

[16] F.R. GANTMACHER, *The Theory of Matrices, Volume 2*, Chelsea Publishing Company, New York, 1959.

[17] G.H. GOLUB AND M.H. GUTKNECHT, *Modified moments for indefinite weight functions*, Numer. Math. (1990) (to appear).

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

[19] M.H. GUTKNECHT, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I*, Tech. Report, December 1989.

[20] M.R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[21] R.A. HORN AND C.R. JOHNSON, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.

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

[23] J.B. KELLER AND D. GIVOLI, *Exact non-reflecting boundary conditions*, J. Comput. Phys., 82 (1989), pp. 172–192.

[24] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45 (1950), pp. 255–282.

[25] ———, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.

[26] ———, *Applied Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1956.

[27] J.A. MEIJERINK AND H.A. VAN DER VORST, *An iterative solution for linear systems of which the coefficient matrix is a symmetric $M-$matrix*, Math. Comp., 31 (1977), pp. 148–162.

[28] G. MORO AND J.H. FREED, *Calculation of ESR spectra and related Fokker-Planck forms by the use of the Lanczos algorithm*, J. Chem. Phys., 74 (1981), pp. 3757–3773.

[29] C.C. PAIGE AND M.A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[30] B.N. PARLETT, D.R. TAYLOR AND Z.A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.

[31] Y. SAAD, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems*, SIAM J. Numer. Anal., 19 (1982), pp. 485–506.

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

[33] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36–52.

[34] D.R. TAYLOR, *Analysis of the look ahead Lanczos algorithm*, Ph.D. Dissertation, University of California, Berkeley, November 1982.

[35] J.H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

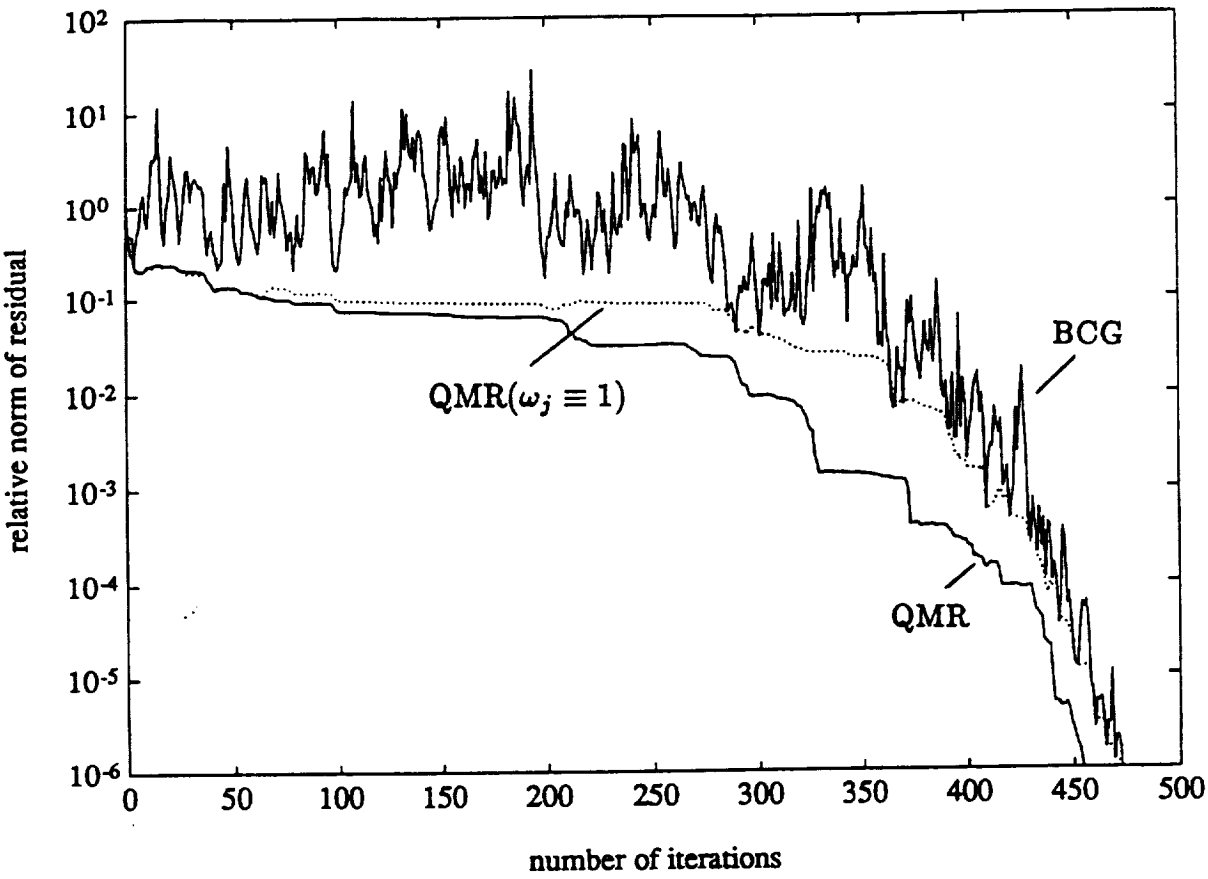
relative norm of residual
number of iterations
QMR($\omega_j \equiv 1$)
BCG
QMR
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
50
100
150
200
250
300
350
400
450
500

Figure 7.1

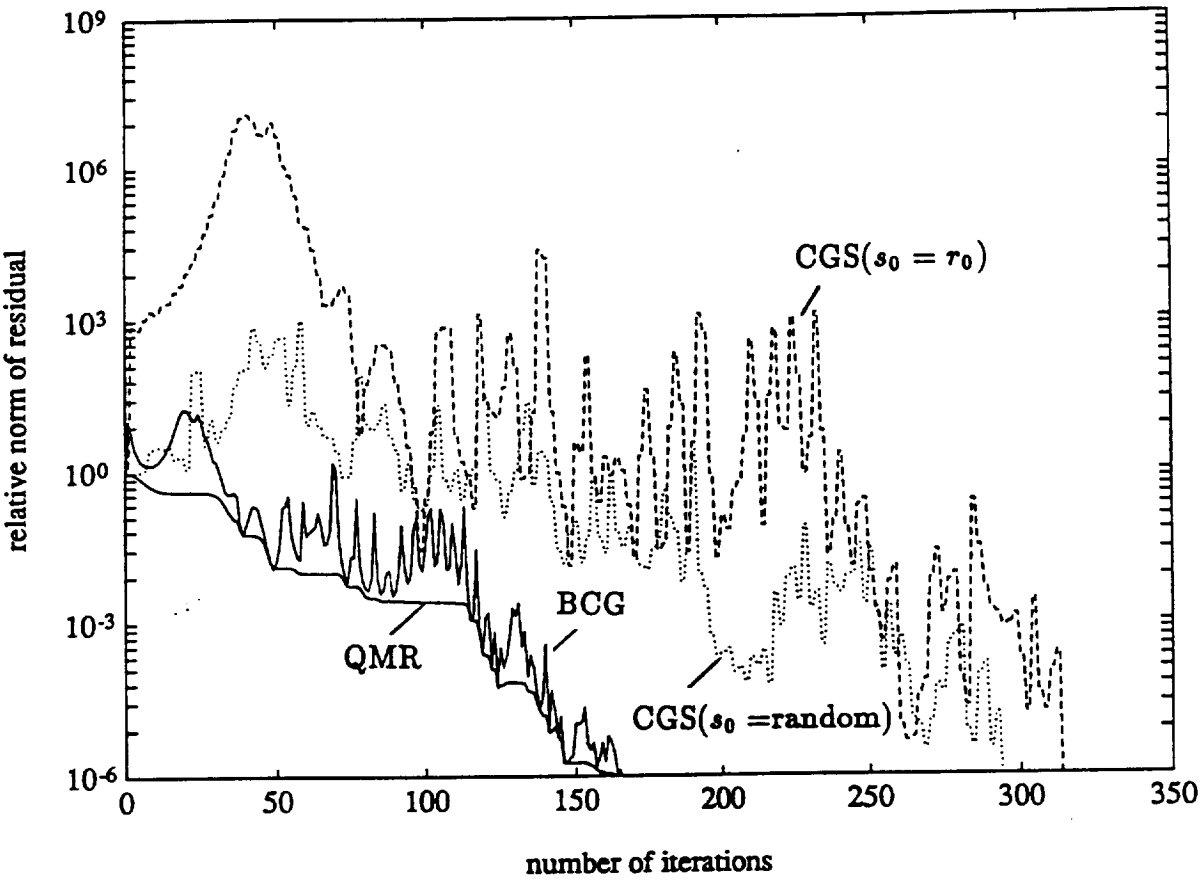
relative norm of residual
number of iterations
CGS($s_0 = r_0$)
BCG
QMR
CGS(s_0 =random)
10^9
10^6
10^3
10^0
10^{-3}
10^{-6}
0
50
100
150
200
250
300
350

Figure 7.2

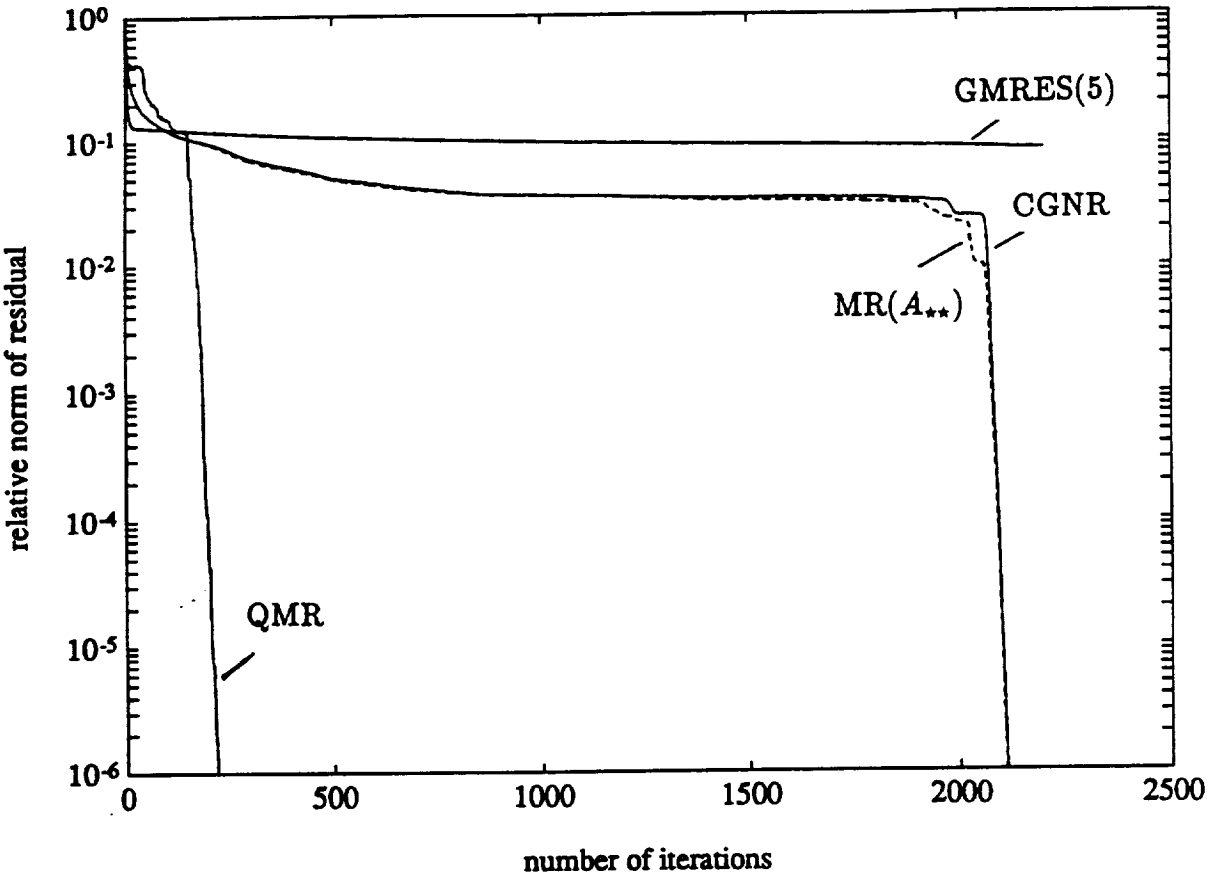
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
relative norm of residual
GMRES(5)
CGNR
MR(A_{**})
QMR
0
500
1000
1500
2000
2500
number of iterations

Figure 7.3

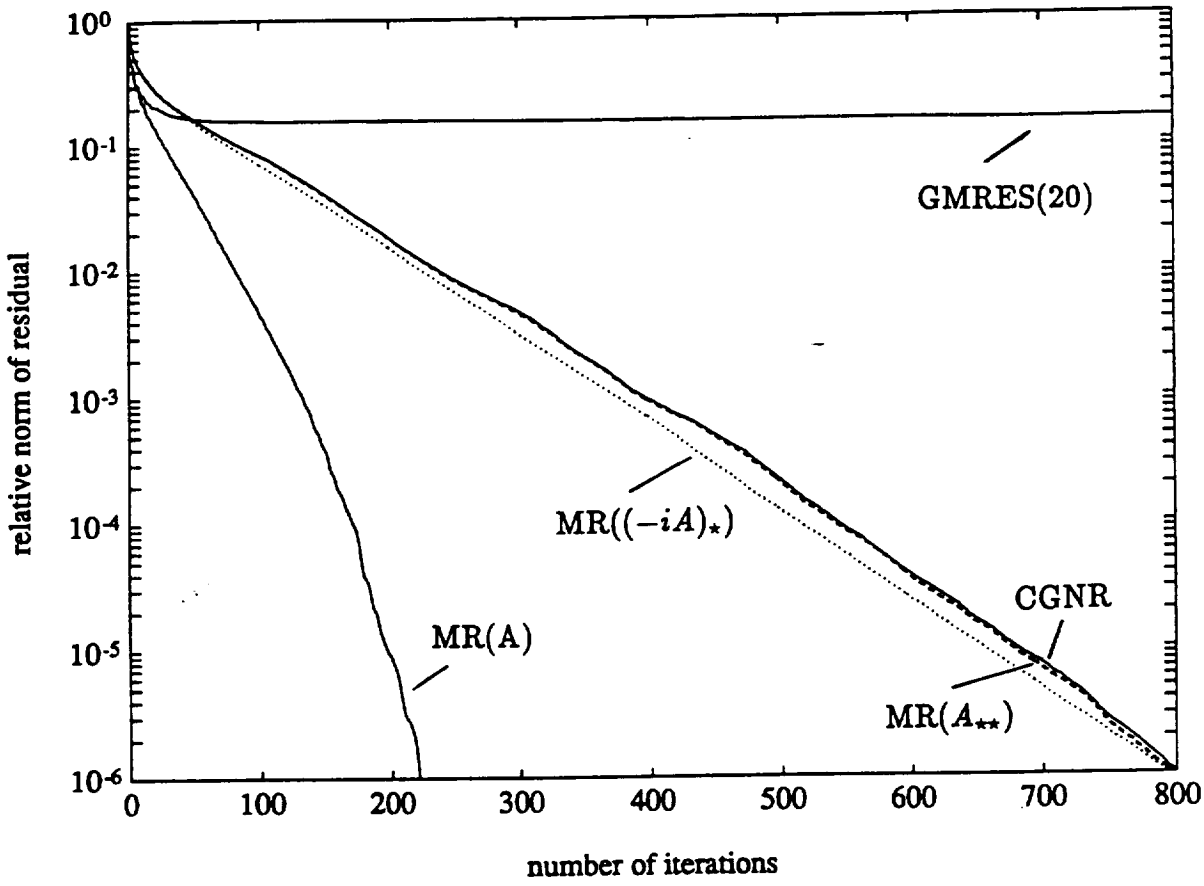
relative norm of residual
number of iterations
GMRES(20)
MR((−iA)*)
CGNR
MR(A)
MR(A**)
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
10^-6
0
100
200
300
400
500
600
700
800

Figure 7.4

RIACS
Mail Stop 230-5
NASA Ames Research Center
Moffett Field, CA 94035