RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

Krylov Subspace Methods for Complex Non-Hermitian Linear Systems

Roland W. Freund

RIACS Technical Report 91.11

May 1991

Krylov Subspace Methods for Complex Non-Hermitian Linear Systems

Roland W. Freund

RIACS Technical Report 91.11

May 1991

Krylov Subspace Methods for Complex Non-Hermitian Linear Systems

Roland W. Freund

The Research Institute for Advanced Computer Science is operated by
Universities Space Research Association (USRA),
The American City Building, Suite 311, Columbia, MD 21044, (301)730–2656.

Work reported herein was supported in part by DARPA via Cooperative Agreement NCC 2–387 between NASA and USRA.

Krylov Subspace Methods for Complex Non-Hermitian Linear Systems

Roland W. Freund

RIACS, Mail Stop Ellis Street
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

Abstract. We consider Krylov subspace methods for the solution of large sparse linear systems $Ax = b$ with complex non-Hermitian coefficient matrices. Such linear systems arise in important applications, such as inverse scattering, numerical solution of time-dependent Schrödinger equations, underwater acoustics, eddy current computations, numerical computations in quantum chromodynamics, and numerical conformal mapping. Typically the resulting coefficient matrices A exhibit special structures, such as complex symmetry, or they are shifted Hermitian matrices. In this paper, we first describe a Krylov subspace approach with iterates defined by a quasi-minimal residual property, the QMR method, for solving general complex non-Hermitian linear systems. Then, we study special Krylov subspace methods designed for the two families of complex symmetric respectively shifted Hermitian linear systems. We also include some results concerning 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.

This work was supported in part by DARPA via Cooperative Agreement NCC 2-387 between NASA and the Universities Space Research Association (USRA).

Contents

1. Introduction

1.1. Krylov subspace methods 1
1.2. Ideal Krylov subspace methods for non-Hermitian matrices 2
1.3. Complex linear systems 3
1.4. Overview of the thesis 4
1.5. Notation 6

2. An implementation of the look-ahead Lanczos process for non-Hermitian matrices

2.1. The classical nonsymmetric Lanczos algorithm 8
2.2. Orthogonal polynomials 10
2.3. The look-ahead Lanczos algorithm 12
2.4. The look-ahead strategy 17
2.5. Implementation details 23

3. A quasi-minimal residual method for general non-Hermitian matrices

3.1. The quasi-minimal residual approach 25
3.2. Implementation details 27
3.3. The connection between QMR and BCG 30
3.4. A convergence theorem 33
3.5. QMR for shifted matrices 36
3.6. Preconditioned QMR 38

4. Lanczos methods for complex symmetric matrices

4.1. The Lanczos recursion for complex symmetric matrices 40
4.2. A theorem on incurable breakdowns 42
4.3. QMR and related algorithms for complex symmetric matrices 44

5. CG-type algorithms and polynomial preconditioning for shifted Hermitian matrices

5.1. Three CG-type approaches 46
5.2. Practical implementations 49
5.3. Comparisons with other implementations. Operation counts 53

5.4. Error bounds 55

5.5. Polynomial preconditioning 62

6. Complex versus equivalent real linear systems

6.1. Equivalent real linear systems 67

6.2. Correspondence of Krylov subspace methods 70

6.3. A connection between MR and CGNR for complex symmetric matrices 71

7. Numerical experiments

7.1. The test problems 74

7.2. Complex symmetric linear systems 75

7.3. Shifted Hermitian linear systems 78

8. Concluding remarks 84

Acknowledgments 85

References 86

1. Introduction

In this chapter, we make some introductory remarks about Krylov subspace methods and list applications where complex linear systems arise. Furthermore, we give an outline of the thesis and introduce some notation.

1.1. Krylov subspace methods

One of the most frequently encountered tasks in numerical computations is the solution of nonsingular systems of linear equations

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

Often, as for linear systems resulting from finite difference or finite element approximations to partial differential equations (PDE's), the coefficient matrix A of (1.1) is very large, but sparse. A natural way to exploit the sparsity of A in the solution process of (1.1) is to use iterative techniques which involve the coefficient matrix A only in the form of matrix-vector products. Most iterative schemes of this type fall into the category of *Krylov subspace methods*: they produce approximations x_n, $n = 1, 2, \ldots$, to $A^{-1}b$ of the form

$$x_n \in x_0 + K_n(r_0, A). \tag{1.2}$$

Here x_0 is any initial guess for (1.1), $r_0 = b - Ax_0$ the corresponding residual vector, and

$$K_n(r_0, A) = \text{span}\{r_0, Ar_0, \ldots, A^{n-1}r_0\}$$

is the nth *Krylov subspace* generated by r_0 and A. Two classical examples of Krylov subspace methods are the conjugate algorithm (CG hereafter) due to Hestenes and Stiefel [HS] and Chebyshev iteration [GV], which are both methods for the solution of linear systems (1.1) with Hermitian positive definite coefficient matrices A. Especially CG is one of the most powerful techniques for solving Hermitian positive definite linear systems. Its success has prompted extensive research into generalizations of the method to indefinite and non-Hermitian matrices and a number of CG-like Krylov subspace methods have been proposed (see, *e.g.*, [Sto, SS1, Saa2] for surveys). Besides CG-like schemes, the second important subclass of Krylov subspace methods are semi-iterative algorithms modeled after Chebyshev iteration. Eiermann, Niethammer, and Varga [ENV] have established a theory for methods of this type for non-Hermitian linear systems.

In this thesis, we are mainly concerned with CG-like Krylov subspace methods.

1.2. Ideal Krylov subspace methods for non-Hermitian matrices

Classical CG has two outstanding features. First, its iterates (1.2) are characterized by a minimization property. Secondly, x_n can be generated cheaply, by means of simple three-term recurrences. For general non-Hermitian matrices, the situation is less satisfactory. An *ideal* CG-like Krylov subspace method for solving non-Hermitian linear systems (1.1) would have features similar to the classical CG algorithm. It would produce iterates x_n in (1.2) which:

(i) are characterized by a minimization property over $K_n(r_0, A)$, such as the minimal residual property

$$\|b - Ax_n\| = \min_{x \in x_0 + K_n(r_0,A)} \|b - Ax\|, \quad x_n \in x_0 + K_n(r_0, A); \tag{1.3}$$

(ii) can be computed with little work and low storage requirements per iteration.

Unfortunately, it turns out that, for general non-Hermitian matrices, one cannot fulfill (i) and (ii) simultaneously. This result is due to Faber and Manteuffel [FM1, FM2] who have shown that CG-type algorithms with (i) and (ii) exist essentially only for matrices of the special form

$$A = e^{i\theta}(T + i\sigma I) \quad \text{where} \quad T = T^H \quad \text{is Hermitian}, \quad \sigma, \theta \in \mathbf{R}, \tag{1.4}$$

(see also Voevodin [Voe] and Joubert and Young [JY]). Instead, most CG-type methods for non-Hermitian linear systems satisfy either (i) or (ii).

In the first category, the most successful scheme is the generalized minimal residual algorithm (GMRES hereafter) due to Saad and Schultz [SS2]. It produces the iterates defined by (1.3) and thus fulfills (i). However, it violates (ii), since work and storage per iteration grow linearly with the iteration number. Consequently, in practice, one cannot afford to run the full algorithm and it is necessary to use restarts. For difficult problems, this often results in very slow convergence.

In the second category, the archetype is the biconjugate gradient algorithm (BCG hereafter) which goes back to Lanczos [Lan2] and, later on, was revived by Fletcher [Fle]. BCG is based on simple three-term recurrences, which keep work and storage requirements constant at each iteration. However, the BCG iterates are defined by a Galerkin condition rather than a minimization property (i), which means that the algorithm can exhibit — and typically does — a rather irregular convergence behavior with wild oscillations in the residual norm. Furthermore, in the BCG algorithm, breakdowns — more precisely, division by 0 — may occur. In finite precision arithmetic, such exact breakdowns are very unlikely; however, near-breakdowns may occur, leading to numerical instabilities in subsequent iterations. Recently, two modifications of BCG, namely CGS [Son] and Bi-CGSTAB [Van], have been proposed. However, while these methods seem to work well in many cases, they do not address the problem of breakdowns, and thus they too, like BCG. are susceptible to instabilities. In exact arithmetic, both CGS and Bi-CGSTAB break down every time BCG does.

1.3. Complex linear systems

While most linear systems which arise in practice have real coefficient matrices A and real right-hand sides b, there are some important applications which lead to complex linear systems. PDE's which model dissipative processes (see, *e.g.*, [Pie, Chapter 10], [Mar]) usually involve complex coefficient functions and/or complex boundary conditions [BGuT, KG], and discretizing them yields linear systems with complex matrices A. A typical example for this category is the complex Helmholtz equation

$$-\Delta u - \sigma_1 u + i\sigma_2 u = f, \tag{1.5}$$

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 [EH, Chapter 8]. Equations of type (1.5) also arise in situations where damping is usually negligible, as in long-range wave propagation problems in underwater acoustics [BGoT, Gol, SLJ], where, by means of parabolic approximation techniques [Tap] and discretization in range direction, the computation of three-dimensional wave propagation is reduced to the solution of a two-dimensional complex Helmholtz equation at each range step. Further applications, which give rise to complex linear systems, include discretizations of the time-dependent Schrödinger equation

$$i\frac{\partial u}{\partial t} = -\Delta u + V(u) \tag{1.6}$$

using implicit difference schemes [DFP], electromagnetic inverse scattering problems [PM. SPM], eddy current computations [BHST], numerical computations in quantum chromodynamics [BBGRM], and numerical conformal mapping [Tru].

In all these examples, the resulting coefficient matrices A are non-Hermitian. However, they still exhibit special structures. Often, as for the linear systems resulting from (1.6), A is a shifted Hermitian matrix, *i.e.*, a matrix of the form (1.4). In most other cases, which lead to complex systems, as for the linear systems resulting from the complex Helmholtz equation (1.5) with first-order boundary conditions, the coefficient matrix is complex symmetric:

$$A = A^T. \tag{1.7}$$

Note that the two families (1.4) and (1.7) overlap. The matrix (1.4) is complex symmetric if, and only if, T is real.

Surprisingly, when the resulting linear systems (1.1) are solved in practice, usually no attempt is made to exploit the special structures (1.4) or (1.7). Indeed, there are two popular approaches. The first one (see, *e.g.*, [BG]) is to apply preconditioned CG to the Hermitian positive definite normal equations

$$A^H A x = A^H b. \tag{1.8}$$

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 CG-like methods, for example GMRES. 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.

Finally, we mention two applications for which shifted linear systems

$$\begin{gathered} Ax = b, \quad A = M + \sigma I, \\ \text{where} \quad M \text{ and } b \text{ are real and fixed}, \quad \sigma \in \mathbf{C}, \end{gathered} \tag{1.9}$$

need to be solved repeatedly for different shifts σ. This situation arises when real parabolic equations are solved using high-order implicit methods (see, *e.g.*, [GS1, GS2]). Furthermore, linear systems (1.9) also come up in the context of frequency response computation in control theory [Lau].

1.4. Overview of the thesis

The purpose of this thesis is twofold. First, we present a novel BCG-like approach for general nonsingular non-Hermitian linear systems (1.1), the quasi-minimal residual algorithm (QMR hereafter), which overcomes the problems of BCG. The QMR method was first proposed by Freund [Fre4] for the special case of complex symmetric linear systems and recently extended to general non-Hermitian matrices by Freund and Nachtigal [FN1, FN2]. The QMR approach uses a look-ahead variant of the nonsymmetric Lanczos process to generate basis vectors for the Krylov subspaces $K_n(r_0, A)$. The look-ahead Lanczos approach was first proposed by Taylor [Tay] and Parlett, Taylor, and Liu [PTL]. For the QMR method, we use the implementation of the look-ahead Lanczos process which was recently developed by Freund. Gutknecht, and Nachtigal [FGN, FN1]. Using the Lanczos basis, the actual QMR iterates are then defined by a relaxed version of (1.3), namely a quasi-minimal residual property. The QMR approach can be implemented using only short recurrences and hence it still satisfies the requirement (ii) for an ideal Krylov subspace method. The quasi-minimal residual property ensures that QMR, unlike BCG, converges smoothly; moreover, existing BCG iterates can also be easily and stably recovered from the QMR process. Finally, for the QMR method, it is possible to obtain error bounds which are essentially the same as the standard bounds for GMRES. To the best of our knowledge, this is the first convergence result for a BCG-like algorithm.

Second, we present CG-type methods which exploit the special structures (1.4) respectively (1.7). In particular, we show that, for complex symmetric matrices, work and storage for the QMR approach can be halved. For shifted Hermitian matrices (1.4), we propose and analyze three different CG-type methods based on the minimal residual property (1.3), a Galerkin condition, and an Euclidean error minimization property. For the practical use of CG-type methods it is crucial that they can be combined with efficient preconditioners. Unfortunately, the more classical techniques, such as incomplete factorization, lead to preconditioned matrices which in general are no longer in the class (1.4). We show that this problem can be resolved and the special structure of the matrices (1.4) preserved by using polynomial preconditioning, and results on the optimal choice of the preconditioner are given. Note that polynomial preconditioning is an attractive approach for vector and parallel computers and, because of that, has become very popular in recent years (see [Saa2] for a survey).

Finally, we also present some results which indicate that for Krylov subspace methods it is always preferable to solve the original complex linear system rather than equivalent real ones.

The outline of this thesis is then as follows. In Section 2, we are concerned with the nonsymmetric Lanczos process. In particular, we sketch the implementation of the look-ahead Lanczos algorithm proposed in [FGN, FN1]. In Section 3, we present the QMR method for general nonsingular non-Hermitian matrices. In Section 4, we consider CG-type algorithms for complex symmetric matrices. In Section 5, we study CG-like methods for shifted Hermitian matrices. In Section 6, we are concerned with the issue "complex versus equivalent real linear systems". In Section 7, we present some numerical examples for complex symmetric and shifted Hermitian linear systems. Finally, in Section 8, we make some concluding remarks.

1.5. Notation

Throughout this thesis, all vectors and matrices are assumed to be complex in general. As usual, $i = \sqrt{-1}$. For any matrix $M = [\,m_{jk}\,]$, we use the following notation:

$$\begin{aligned}
\overline{M} &= [\,\overline{m_{jk}}\,] = \text{the complex conjugate of } M,\\
M^T &= [\,m_{kj}\,] = \text{the transpose of } M,\\
M^H &= \overline{M}^T = \text{the Hermitian of } M,\\
\operatorname{Re} M &= (M + \overline{M})/2 = \text{the real part of } M,\\
\operatorname{Im} M &= (M - \overline{M})/(2i) = \text{the imaginary part of } M,\\
\sigma_{\max}(M) &= \text{the largest singular value of } M,\\
\sigma_{\min}(M) &= \text{the smallest singular value of } M,\\
\|M\| &= \sigma_{\max}(M) = \text{the 2−norm of } M.
\end{aligned}$$

For any vector $c \in \mathbf{C}^m$ and any matrix $B \in \mathbf{C}^{m\times m}$, we use the following notations:

$$\begin{aligned}
\|c\| &= \sqrt{c^H c} = \text{Euclidean norm of } c,\\
\|c\|_B &= \sqrt{c^H B c} = B\text{−norm of } c, \text{ if } B \text{ is Hermitian positive definite,}\\
\lambda(B) &= \text{the set of eigenvalues of } B,\\
\lambda_{\max}(B) &= \text{the largest eigenvalue of } B, \text{ if } B \text{ is Hermitian,}\\
\lambda_{\min}(B) &= \text{the smallest eigenvalue of } B, \text{ if } B \text{ is Hermitian,}\\
K_n(c,B) &= \operatorname{span}\{c, Bc, \ldots, B^{n-1}c\}\\
&= \text{the } n\text{th Krylov subspace of } \mathbf{C}^m \text{ generated by } c \text{ and } B.
\end{aligned}$$

Furthermore, we denote by

$$e_j^{(n)} = [0 \quad \cdots \quad 0 \quad \underset{\substack{\uparrow \\ j}}{1} \quad 0 \quad \cdots \quad 0]^T \in \mathbf{R}^n$$

the jth unit vector of length n and by I_n the $n \times n$ identity matrix. If the dimension n is evident from the context, we will simply write e_j and I. We denote by

$$\begin{aligned}
\Pi_n &= \{\Phi(\lambda) \equiv \sigma_0 + \sigma_1\lambda + \cdots + \sigma_n\lambda^n \mid \sigma_0, \sigma_1, \ldots, \sigma_n \in \mathbf{C}\}\\
\text{and}\quad \Pi_n^{(r)} &= \{\Phi(\lambda) \equiv \sigma_0 + \sigma_1\lambda + \cdots + \sigma_n\lambda^n \mid \sigma_0, \sigma_1, \ldots, \sigma_n \in \mathbf{R}\}
\end{aligned}$$

the set of complex and real polynomials of degree at most n, respectively. Frequently, we will make use of the relation

$$K_n(c,B) = \{\Phi(B)c \mid \Phi \in \Pi_{n-1}\}, \quad n = 1, 2, \ldots\ . \tag{1.10}$$

Throughout this thesis, N denotes the dimension of the coefficient matrix A of (1.1) and $A \in \mathbf{C}^{N \times N}$ is in general non-Hermitian. Moreover, we use the following notation:

$$
\begin{aligned}
x_0 &= \text{initial guess for (1.1)},\\
x_n &= n\text{th iterate},\\
r_n &= b - Ax_n = n\text{th residual vector},\\
v_n &= n\text{th right Lanczos vector},\\
w_n &= n\text{th left Lanczos vector}.
\end{aligned}
$$

If it is not evident from the context which iterative method we are considering, quantities of different algorithms will be distinguished by superscripts, *e.g.*, x_n^{QMR} and x_n^{BCG}.

Finally, one more note. In our formulations of the nonsymmetric Lanczos algorithm and of BCG, we use A^T rather than A^H. This was a deliberate choice in order to avoid complex conjugation of the scalars in the recurrences; the algorithms can be formulated equally well in either terms.

2. An implementation of the look-ahead Lanczos process for non-Hermitian matrices

In this chapter, we first recall the classical nonsymmetric Lanczos method and its close relationship with *formally orthogonal polynomials* (FOP's hereafter). Next, we describe the basic idea of look-ahead Lanczos procedures, and finally, we present an actual implementation of a Lanczos algorithm with look-ahead.

2.1. The classical nonsymmetric Lanczos algorithm

In 1950, Lanczos [Lan1] proposed the following algorithm for successive reduction of a general matrix $A \in \mathbf{C}^{N \times N}$ to tridiagonal form.

Algorithm 2.1. (Classical Lanczos method.)

```
0) Choose $r_0$, $s_0 \in \mathbf{C}^N$ with $r_0$, $s_0 \neq 0$;
   Set $\tilde{v}_1 = r_0$, $\tilde{w}_1 = s_0$, $v_0 = w_0 = 0$;
For $n = 1, 2, \ldots$ :
1) Compute $\eta = \tilde{w}_n^T \tilde{v}_n$;
   If $\eta = 0$: set $L = n - 1$, and stop;
2) Otherwise, choose $\beta_n, \gamma_n \in \mathbf{C}$ with $\beta_n \gamma_n = \eta$;
   Set $v_n = \tilde{v}_n / \gamma_n$ and $w_n = \tilde{w}_n / \beta_n$;
3) Compute $\alpha_n = w_n^T A v_n$;
   Set $\tilde{v}_{n+1} = A v_n - \alpha_n v_n - \beta_n v_{n-1}$;
   Set $\tilde{w}_{n+1} = A^T w_n - \alpha_n w_n - \gamma_n w_{n-1}$.
```

We refer to [Wil, pp. 388–394] for a detailed discussion of the Lanczos algorithm; in particular, proofs of the properties collected in Proposition 2.2 below can be found there. In the sequel, the notations

$$V_n = [\, v_1 \quad v_2 \quad \cdots \quad v_n \,], \quad W_n = [\, w_1 \quad w_2 \quad \cdots \quad w_n \,], \tag{2.1}$$

$$\text{and} \quad H_n = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \gamma_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & \gamma_n & \alpha_n \end{bmatrix} \tag{2.2}$$

will be used. Moreover, let

$$L_r = \dim K_N(r_0, A) \quad \text{and} \quad L_l = \dim K_N(s_0, A^T) \tag{2.3}$$

denote the *grade* of r_0 with respect to A and the *grade* of s_0 with respect to A^T, respectively, (cf. [Wil, p. 37]) and set

$$L_* = \min\{L_r, L_l\}. \tag{2.4}$$

We remark that $L_r \geq 1$ ($L_l \geq 1$) is just the smallest integer such that the subspace $K_{L_r}(r_0, A)$ ($K_{L_l}(s_0, A^T)$) is A-invariant (A^T-invariant).

Proposition 2.2.

a) In exact arithmetic, Algorithm 2.1 stops after a finite number of steps $n = L + 1$ and $0 \leq L \leq L_*$.

b) For k, $n = 1, 2, \ldots, L$:

$$w_k^T v_n = \begin{cases} 0, & \text{if } k \neq n, \\ 1, & \text{if } k = n. \end{cases} \tag{2.5}$$

c) For $n = 1, 2, \ldots, L$:

$$\begin{aligned} K_n(r_0, A) &= \operatorname{span}\{v_1, v_2, \ldots, v_n\}, \\ K_n(s_0, A^T) &= \operatorname{span}\{w_1, w_2, \ldots, w_n\}, \end{aligned} \tag{2.6}$$

$$\begin{aligned} AV_n &= V_n H_n + [0 \quad 0 \quad \cdots \quad 0 \quad \tilde{v}_{n+1}], \\ A^T W_n &= W_n H_n^T + [0 \quad 0 \quad \cdots \quad 0 \quad \tilde{w}_{n+1}]. \end{aligned} \tag{2.7}$$

Note that the *termination index* L of Algorithm 2.1 is the smallest integer such that

$$\tilde{w}_{L+1}^T \tilde{v}_{L+1} = 0. \tag{2.8}$$

There are two essentially different cases for fulfilling the termination condition (2.8). The first case, referred to as *regular termination*, occurs when $\tilde{v}_{L+1} = 0$ or $\tilde{w}_{L+1} = 0$. If $\tilde{v}_{L+1} = 0$, then $L = L_r$ and the *right* Lanczos vectors $v_1, \ldots, v_{L_r}$ span the A-invariant subspace $K_{L_r}(r_0, A)$. Similarly, if $\tilde{w}_{L+1} = 0$, then $L = L_l$ and the *left* Lanczos vectors $w_1, \ldots, w_{L_l}$ span the A^T-invariant subspace $K_{L_l}(s_0, A^T)$. Unfortunately, it can also happen that the termination condition (2.8) is satisfied with $\tilde{v}_{L+1} \neq 0$ and $\tilde{w}_{L+1} \neq 0$. This second case is referred to as *serious breakdown* [Wil, p. 389]. Note that, in this case, $L < L_*$ and the Lanczos vectors span neither an A-invariant nor an A^T-invariant subspace of $\mathbf{C}^N$.

It is the possibility of serious breakdowns, or, in finite precision arithmetic, of *near-breakdowns*, *i.e.*,

$$\tilde{w}_n^T \tilde{v}_n \approx 0, \quad \text{but} \quad \tilde{w}_n \not\approx 0 \quad \text{and} \quad \tilde{v}_n \not\approx 0, \tag{2.9}$$

that has brought the classical nonsymmetric Lanczos algorithm into discredit. However, by means of a look-ahead procedure, it is possible to leap (except in the very special case of an *incurable breakdown* [Tay]) over those iterations in which the standard algorithm would break down. In the next section, using the intimate connection between the Lanczos process and FOP's, we describe the basic idea of the Lanczos method with look-ahead.

2.2. Orthogonal polynomials

One readily verifies that the Lanczos vectors generated by Algorithm 2.1 are of the form

$$v_n = \frac{1}{\gamma_1\gamma_2\cdots\gamma_n}\Phi_{n-1}(A)r_0 \quad \text{and} \quad w_n = \frac{1}{\beta_1\beta_2\cdots\beta_n}\Phi_{n-1}(A^T)s_0, \tag{2.10}$$

where $\Phi_{n-1} \in \Pi_{n-1}$ is a uniquely defined monic polynomial. Then, introducing the formal inner product

$$(\Phi, \Psi) := \left(\Psi(A^T)s_0\right)^T \left(\Phi(A)r_0\right) = s_0^T \Psi(A)\Phi(A)r_0 \tag{2.11}$$

and using (2.6), (1.10), and (2.10), we can rewrite the biorthogonality condition (2.5) in terms of polynomials:

$$(\Phi_{n-1}, \Phi) = 0 \quad \text{for all} \quad \Phi \in \Pi_{n-2} \tag{2.12}$$

and

$$(\Phi_{n-1}, \Phi_{n-1}) \neq 0. \tag{2.13}$$

Note that, except for the case of Hermitian $A = A^H$ (cf. Chapter 5), the formal inner product (2.11) is indefinite. Therefore, in the general case, there exist polynomials $\Phi \neq 0$ with "length" $(\Phi, \Phi) = 0$ or even $(\Phi, \Phi) < 0$.

A polynomial $\Phi_{n-1} \in \Pi_{n-1}$, $\Phi_{n-1} \neq 0$, that fulfills (2.12) is called a FOP (with respect to the formal inner product (2.11)) of degree $n-1$ (see, *e.g.*, [Bre], [Dra], [Gut]). Note that the condition (2.12) is empty for $n = 1$, and hence any $\Phi_0 \equiv \sigma_0 \neq 0$ is a FOP of degree 0. From (2.12),

$$\Phi_{n-1}(\lambda) \equiv \sigma_0 + \sigma_1\lambda + \cdots + \sigma_{n-1}\lambda^{n-1}$$

is a FOP of degree $n-1$ if, and only if, its coefficients $\sigma_0, \ldots, \sigma_{n-1}$ are a nontrivial solution of the linear system

$$\begin{bmatrix} \mu_0 & \mu_1 & \mu_2 & \cdots & \mu_{n-2} \\ \mu_1 & \iddots & & \iddots & \vdots \\ \mu_2 & & \iddots & & \vdots \\ \vdots & \iddots & & & \mu_{2n-5} \\ \mu_{n-2} & \cdots & \cdots & \mu_{2n-5} & \mu_{2n-4} \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_{n-2} \end{bmatrix} = -\sigma_{n-1} \begin{bmatrix} \mu_{n-1} \\ \mu_n \\ \mu_{n+1} \\ \vdots \\ \mu_{2n-3} \end{bmatrix}. \tag{2.14}$$

Here

$$\mu_j = s_0^T A^j r_0 = (\lambda^j, 1), \quad j = 0, 1, \ldots,$$

are the moments associated with (2.11). A FOP Φ_{n-1} is called *regular* if it is uniquely determined by (2.12) up to a scalar, and it is said to be ***singular*** otherwise. We remark that a FOP of degree 0 is always regular. With (2.14), one easily verifies the statements in the following

Proposition 2.3.
a) A regular FOP Φ_{n-1} has exactly degree $n-1$. In particular, a regular FOP is unique if it is required to be monic.
b) A regular FOP of degree $n-1$ exists if, and only if, the coefficient matrix of (2.14) is nonsingular.
c) Let Φ_{n-1} be a regular FOP (with respect to the formal inner product (2.11)) of degree $n-1$. Then, a regular FOP of degree n exists if, and only if, (2.13) is satisfied.

We remark that, by part b) of Proposition 2.3, singular FOP's occur if, and only if, the corresponding linear system (2.14) has a singular coefficient matrix, but is consistent. If (2.14) is inconsistent, then no FOP Φ_{n-1} exists. This case is referred to as ***deficient***, and by relaxing (2.12) slightly, one can define so-called ***deficient*** FOP's (see [Gut] for details). Simple examples (see, *e.g.*, [FN1, Section 13]) show that the singular and deficient cases do indeed occur.

Now let us return to the classical nonsymmetric Lanczos process 2.1. Using (2.8), (2.10), (2.11), and part c) of Proposition 2.3, we conclude that a serious breakdown occurs if, and only if, no regular FOP exists for some $L < L_*$. In this case, the termination index L is the smallest integer L for which there exists no regular FOP of degree L.

On the other hand, there is a maximal subset of indices

$$\{n_1, n_2, \ldots, n_J\} \subseteq \{1, 2, \ldots, L_*\}, \quad n_1 := 1 < n_2 < \cdots < n_J \leq L_*, \tag{2.15}$$

such that, for each $j = 1, 2, \ldots, J$, there exists a monic regular FOP $\Phi_{n_j-1} \in \Pi_{n_j-1}$. Note that $n_1 = 1$ since $\Phi_0(\lambda) \equiv 1$ is a monic regular FOP of degree 0. Furthermore, three successive regular FOP's $\Psi_{n_{j-1}-1}$, Ψ_{n_j-1}, and $\Psi_{n_{j+1}-1}$ are connected via a relation of the form

$$\begin{gathered}\Phi_{n_{j+1}-1}(\lambda) \equiv \Psi_{n_j-1}(\lambda)\Phi_{n_j-1}(\lambda) - \delta_{n_j-1}\Phi_{n_{j-1}-1}(\lambda),\\ \text{where } \Psi_{n_j-1} \in \Pi_{n_{j+1}-n_j},\ \delta_{n_j-1} \in \mathbb{C}.\end{gathered} \tag{2.16}$$

The recurrences (2.16) for FOP's were mentioned by Gragg [Gra, pp. 222–223] and by Draux [Dra]; also, in the context of the partial realization problem, by Kung [Kun, Chapter IV] and Gragg and Lindquist [GL]. For a proof of (2.16), we refer the reader to [Gut].

Now, setting, in analogy to (2.10),

$$v_{n_j} = \phi_{n_j}\Phi_{n_j-1}(A)r_0 \quad \text{and} \quad w_{n_j} = \psi_{n_j}\Phi_{n_j-1}(A^T)s_0,$$

where $\phi_{n_j}, \psi_{n_j} \neq 0$ are scaling factors, we obtain two sequences of vectors $\{v_{n_j}\}_{j=1}^J$ and $\{w_{n_j}\}_{j=1}^J$ which, in view of (2.16), can be computed by means of short recurrences. These vectors will be called ***regular*** vectors, since they correspond to regular FOP's. Note that v_1 and w_1 are always regular. The look-ahead Lanczos procedure is an extension of the classical nonsymmetric Lanczos algorithm; in exact arithmetic, it generates the vectors v_{n_j} and w_{n_j}, $j = 1, \dots, J$. If $n_J = L_*$ in (2.15), then these vectors can be complemented to a basis for an A-invariant or A^T-invariant subspace of $\mathbf{C}^N$. An incurable breakdown occurs if, and only if, $n_J < L_*$ in (2.15). Finally, note that the regular vectors v_{n_j} and w_{n_j} are uniquely defined (up to a nonzero scalar) by the biorthogonality relations

$$w_{n_j}^T v = w^T v_{n_j} = 0 \quad \text{for all} \quad v \in K_{n_j-1}(r_0, A), \; w \in K_{n_j-1}(s_0, A^T), \qquad j = 1, \dots, J. \tag{2.17}$$

The look-ahead procedure we have sketched so far only skips over exact breakdowns. It yields what is called the *nongeneric* Lanczos algorithm in [Gut]. Of course, in finite precision arithmetic, a viable look-ahead Lanczos algorithm also needs to leap over near-breakdowns (2.9). Roughly speaking, a robust implementation should attempt to generate only the "well-defined" regular vectors. In practice, then, one aims at generating two sequences of vectors $\{v_{n_{j_k}}\}_{k=1}^K$ and $\{w_{n_{j_k}}\}_{k=1}^K$, where

$$\{n_{j_k}\}_{k=1}^K \subseteq \{n_j\}_{j=1}^J, \quad j_1 := 1, \tag{2.18}$$

is a suitable subset of (2.15). We set $j_1 = 1$, since v_1 and w_1 are always regular.

Taylor [Tay] and Parlett, Taylor, and Liu [PTL] were the first to propose such a practical procedure. However, in [Tay, PTL], the details of an actual implementation are worked out only for look-ahead steps of length 2.

In [FGN, FN1], Freund, Gutknecht, and Nachtigal have proposed an implementation of the look-ahead Lanczos method for general complex non-Hermitian matrices. The algorithm can handle look-ahead steps of any length and is not restricted to steps of length 2. On many modern computer architectures, the computation of inner products of long vectors is a bottleneck. The algorithm described in [FGN, FN1] has the additional feature that it requires the same number of inner products as the classical Lanczos process, as opposed to the look-ahead algorithm described in [Tay, PTL], which always requires additional inner products. In particular, our implementation differs from the one in [Tay, PTL] even for look-ahead steps of length 2.

In the next section, we present a sketch of the look-ahead Lanczos algorithm proposed in [FGN, FN1] and list some of its basic properties.

2.3. The look-ahead Lanczos algorithm

First, we introduce some notation. As in the last section, $n = 1, 2, \dots$ denote the indices of the Lanczos vectors v_n and w_n. From now on, we will always normalize the Lanczos vectors so that

$$\|v_n\| = \|w_n\| = 1, \quad n = 1, 2, \dots . \tag{2.19}$$

For simplicity, we set $n_k := n_{j_k}$ for the indices of the "well-defined" regular vectors, cf. (2.18). However, notice that there is no guarantee that the indices n_k generated by the look-ahead Lanczos algorithm in finite precision arithmetic actually satisfy (2.18). The index $k = 1, 2, \ldots$ is used as a counter for the computed regular Lanczos vectors v_{n_k} and w_{n_k}.

In order to obtain complete bases for the subspaces $K_n(r_0, A)$ and $K_n(s_0, A^T)$, we need to add vectors

$$v_n \in K_n(r_0, A) \setminus K_{n-1}(r_0, A) \quad \text{and} \quad w_n \in K_n(s_0, A^T) \setminus K_{n-1}(s_0, A^T), \qquad n = n_{k-1}+1, \ldots, n_k - 1, \ k = 2, 3, \ldots, \tag{2.20}$$

to the two sequences of computed regular vectors v_{n_k} and w_{n_k}, $k = 1, 2, \ldots$, respectively. The vectors in (2.20) are called *inner* vectors. We will refer to both the regular and the inner vectors v_n and w_n generated by the look-ahead variant as right and left Lanczos vectors, in analogy to the terminology for the classical nonsymmetric Lanczos Algorithm 2.1.

For each fixed $n = 1, 2, \ldots$, we denote by $l = l(n)$ the number of the last computed regular vector with index $\leq n$. Then, the first n Lanczos vectors $v_1, \ldots, v_n$ and $w_1, \ldots, w_n$ generated by the look-ahead Lanczos process can be grouped into l blocks

$$\begin{aligned} V^{(k)} &= [\, v_{n_k} \quad v_{n_k+1} \quad \cdots \quad v_{n_{k+1}-1} \,], \quad W^{(k)} = [\, w_{n_k} \quad w_{n_k+1} \quad \cdots \quad w_{n_{k+1}-1} \,], \\ &\qquad k = 1, 2, \ldots, l-1, \\ V^{(l)} &= [\, v_{n_l} \quad v_{n_l+1} \quad \cdots \quad v_n \,], \quad W^{(l)} = [\, w_{n_l} \quad w_{n_l+1} \quad \cdots \quad w_n \,]. \end{aligned} \tag{2.21}$$

In the sequel, we denote by

$$h_k = n_{k+1} - n_k, \ k = 1, 2, \ldots, l-1, \quad \tilde{h}_l = n - n_l$$

the number of vectors in each block. Note that the first vectors v_{n_k} and w_{n_k} in each block are just the regular vectors. The lth block is called *complete* if $n = n_{l+1} - 1$; in this case, at the next step $n+1$, a new block is started with the regular vectors $v_{n_{l+1}}$ and $w_{n_{l+1}}$. Otherwise, if $n < n_{l+1} - 1$, the lth block is *incomplete* and at the next step, the Lanczos vectors v_{n+1} and w_{n+1} are added to the lth block as inner vectors.

So far, we have not specified how to actually construct the inner vectors. The point is that the inner vectors can be chosen such that the v_n's and w_n's from blocks corresponding to different indices k are still biorthogonal to each other. More precisely, in analogy to the biorthogonality relation (2.5) for the classical Lanczos algorithm, we have

$$(W^{(j)})^T V^{(k)} = \begin{cases} 0 & \text{if } j \neq k, \\ D^{(k)} & \text{if } j = k, \end{cases} \qquad j, k = 1, 2, \ldots, l. \tag{2.22}$$

We remark that the inner vectors constructed because of an exact breakdown correspond to singular or deficient FOP's, while the inner vectors constructed because of a near-breakdown correspond to polynomials which in general are combinations of regular, singular, and deficient FOP's.

Next, we show that the matrices $D^{(k)}$ in (2.22) are necessarily nonsingular, except for possibly the lth block, *i.e.*,

$$D^{(k)} \text{ is nonsingular, } \; k = 1,2,\dots,l-1, \; \text{ and } D^{(l)} \text{ is nonsingular if } n = n_{l+1}-1. \tag{2.23}$$

Indeed, assume that $D^{(k)} = (W^{(k)})^T V^{(k)}$ is singular for some $k \leq l$, where, in the case $k = l$, the lth block is complete. Then, there exists a vector z such that

$$(W^{(k)})^T V^{(k)} z = 0 \quad \text{and} \quad V^{(k)} z \neq 0. \tag{2.24}$$

With (2.22) and (2.24), it follows that $\tilde{v} = v_{n_{k+1}} + V^{(k)} z$ fulfills

$$w^T \tilde{v} = 0 \quad \text{for all} \quad w \in K_{n_{k+1}-1}(s_0, A^T). \tag{2.25}$$

Using (2.17) and (2.25), we conclude that $\tilde{v} = \phi v_{n_{k+1}}$ for some scalar $\phi \neq 0$, which is impossible.

With these preliminaries, the basic structure of the look-ahead Lanczos algorithm is as follows.

Algorithm 2.4. (Sketch of the look-ahead Lanczos process.)

```
0) Choose $r_0$, $s_0 \in \mathrm{C}^N$ with $r_0$, $s_0 \neq 0$;
     Set $v_1 = r_0/\|r_0\|$, $w_1 = s_0/\|s_0\|$;
     Set $V^{(1)} = v_1$, $W^{(1)} = w_1$, $D^{(1)} = (W^{(1)})^T V^{(1)}$;
     Set $n_1 = 1$, $l = 1$, $v_0 = w_0 = 0$, $V_0 = W_0 = \emptyset$, $\rho_1 = \xi_1 = 1$;
For $n = 1, 2, \dots$ :
1) Decide whether to construct $v_{n+1}$ and $w_{n+1}$ as regular or inner vectors
     and go to 2) or 3), respectively;
2) (Regular step.) Compute
     $\tilde{v}_{n+1} = Av_n - V^{(l)}(D^{(l)})^{-1}(W^{(l)})^T Av_n$
                 $- V^{(l-1)}(D^{(l-1)})^{-1}(W^{(l-1)})^T Av_n,$
                                                                        (2.26)
     $\tilde{w}_{n+1} = A^T w_n - W^{(l)}(D^{(l)})^{-T}(V^{(l)})^T A^T w_n$
                 $- W^{(l-1)}(D^{(l-1)})^{-T}(V^{(l-1)})^T A^T w_n,$
     set $n_{l+1} = n + 1$, $l = l + 1$, $V^{(l)} = W^{(l)} = \emptyset$, and go to 4);
3) (Inner step.) Compute
     $\tilde{v}_{n+1} = Av_n - \zeta_n v_n - (\eta_n/\rho_n) v_{n-1}$
                 $- V^{(l-1)}(D^{(l-1)})^{-1}(W^{(l-1)})^T Av_n,$
                                                                        (2.27)
     $\tilde{w}_{n+1} = A^T w_n - \zeta_n w_n - (\eta_n/\xi_n) w_{n-1}$
                 $- W^{(l-1)}(D^{(l-1)})^{-T}(V^{(l-1)})^T A^T w_n;$
```

4) Compute $\rho_{n+1} = \|\tilde{v}_{n+1}\|$ *and* $\xi_{n+1} = \|\tilde{w}_{n+1}\|$*;*
If $\rho_{n+1} = 0$ *or* $\xi_{n+1} = 0$*: set* $L = n$*, and stop;*
Otherwise, set

$$\begin{aligned} v_{n+1} &= \tilde{v}_{n+1}/\rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}, \\ V^{(l)} &= [\, V^{(l)} \quad v_{n+1} \,], \quad W^{(l)} = [\, W^{(l)} \quad w_{n+1} \,], \\ D^{(l)} &= (W^{(l)})^T V^{(l)}. \end{aligned} \tag{2.28}$$

If only regular steps 2) are performed, all blocks have size $h_l = 1$ and Algorithm 2.4 reduces to the classical Lanczos process. Therefore, the strategy for the decision in step 1) should be such that regular steps are performed whenever possible and blocks of size $h_k > 1$ are built only to avoid exact or near-breakdowns. A practical procedure for the decision in step 1) will be discussed in Section 2.4.

In (2.27), ζ_n and η_n, $n = 0, 1, \ldots$, are recurrence coefficients with $\eta_{n_k} = 0$, $k = 1, 2, \ldots$. One may choose these coefficients so that they remain the same from one block to the next and change only with respect to their index inside the block, $n - n_k$, or one may choose these coefficients so that they change from one block to the next. For instance, one practical choice for the basic three-term recursions

$$v = Av_n - \zeta_n v_n - \eta_n(v_{n-1}/\rho_n) \quad \text{and} \quad w = A^T w_n - \zeta_n w_n - \eta_n(w_{n-1}/\xi_n)$$

for generating the inner vectors in (2.27) is Chebyshev iteration [Man], where the recurrence coefficients are derived from suitably scaled and translated Chebyshev polynomials. In this case, the translation parameters could be adjusted using spectral information obtained from previous Lanczos steps. We do not necessarily advocate the use of fancy recursions in (2.27). From our experience, the algorithm we propose builds very small blocks, typically of size 2 or 3. Except for artificially constructed examples, the largest block we observed in test runs with "real-life" matrices was of size 4. It occurred for the SHERMAN5 matrix from the Harwell-Boeing set of sparse test matrices [DGL] where out of 1500 steps, the algorithm built 2 × 2 blocks 49 times, 3 × 3 blocks 7 times, and one 4 × 4 block (see [FN2, Example 2]). Hence, the recursion in (2.27) is not overly important, and in our experiments, we have used the recursion coefficients $\zeta_n = 1$ and, if $n \neq n_k$, $\eta_n = 1$. Finally, one could consider orthogonalizing (in the Euclidean sense) the right respectively left Lanczos vectors within each block. However, for the blocks we have seen built, such an orthogonalization process did not lead to better numerical properties of the algorithm. Therefore, in view of the additional inner products which need to be computed, orthogonalizing within each block is not justified.

Next, we list some basic properties of Algorithm 2.4 which will be used in the sequel. First, note that the Lanczos vectors generated by Algorithm 2.4 indeed satisfy the block biorthogonality relations (2.22). The proof is standard, using induction on n, and is omitted here. Setting, in analogy to (2.1),

$$\begin{aligned} V_n &= [\, v_1 \quad v_2 \quad \dots \quad v_n \,] &= [\, V^{(1)} \quad V^{(2)} \quad \dots \quad V^{(l)} \,], \\ W_n &= [\, w_1 \quad w_2 \quad \dots \quad w_n \,] &= [\, W^{(1)} \quad W^{(2)} \quad \dots \quad W^{(l)} \,], \end{aligned} \tag{2.29}$$

one clearly has

$$\begin{aligned} K_n(r_0, A) &= \left\{ V^{(n)} z \mid z \in \mathbf{C}^n \right\}, \\ K_n(s_0, A^T) &= \left\{ W^{(n)} z \mid z \in \mathbf{C}^n \right\}. \end{aligned} \tag{2.30}$$

Moreover, the recursions for the v's in (2.26) and (2.27) can be rewritten in matrix formulation as follows:

$$AV_n = V_{n+1} H_n^{(e)}. \tag{2.31}$$

Here,

$$H_n^{(e)} = \begin{bmatrix} H_n \\ \rho_{n+1} e_n^T \end{bmatrix}, \tag{2.32}$$

where

$$H_n = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \gamma_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_l \\ 0 & \cdots & 0 & \gamma_l & \alpha_l \end{bmatrix} \tag{2.33}$$

is an $n \times n$ block tridiagonal matrix with blocks of the form

$$\alpha_k = \begin{bmatrix} * & * & 0 & \ddots & 0 & * \\ \rho_{n_k+1} & * & \ddots & \ddots & \vdots & \vdots \\ 0 & \rho_{n_k+2} & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & * & * \\ \vdots & & \ddots & \ddots & * & * \\ 0 & \cdots & \cdots & 0 & \rho_{n_{k+1}-1} & * \end{bmatrix}, \qquad \gamma_k = \begin{bmatrix} 0 & \cdots & 0 & \rho_{n_k} \\ \vdots & \ddots & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}. \tag{2.34}$$

The blocks β_k are in general full matrices. Furthermore, for $k = 1, \dots, l-1$, the matrices α_k, β_k, and γ_k are of size $h_k \times h_k$, $h_{k-1} \times h_k$, and $h_k \times h_{k-1}$, respectively. The matrices α_l, β_l, and γ_l corresponding to the current block l are of size $\tilde{h}_l \times \tilde{h}_l$, $h_{l-1} \times \tilde{h}_l$, and $\tilde{h}_l \times h_{l-1}$, respectively. Here $\tilde{h}_l = h_l$ if the lth block is complete.

In view of (2.33) and (2.34), $H_n^{(e)}$ is an upper Hessenberg matrix with positive subdiagonal elements, and hence

$$\operatorname{rank} H_n^{(e)} = n. \tag{2.35}$$

In exact arithmetic, the stopping criterion in step 4) of Algorithm 2.4 will be satisfied after L_* steps, where L_* is given by (2.3) and (2.4), except in the very special situation of an incurable breakdown. Recall from Section 2.2 that an incurable breakdown occurs if, and only if, $n_J < L_*$ in (2.15). One can show (cf. [Gut]) that, if $n_J < L_*$, Algorithm 2.4 will produce, starting with the regular vectors v_{n_l} and w_{n_l} where $n_l = n_J$, infinite blocks $V^{(l)}$ and $W^{(l)}$ of nonzero Lanczos vectors such that $(W^{(l)})^T V^{(l)}$ is the infinite zero matrix.

We would like to stress that incurable breakdowns are very rare and do not present a problem in practice. Furthermore, even in the case of an incurable breakdown, the look-ahead Lanczos process still yields information on the spectrum of A, as Taylor [Tay] showed in his Mismatch Theorem (see also [Gut, Par]). For later use, we summarize the termination properties of the look-ahead Lanczos process in the following

Proposition 2.5. *There is a termination index $L \leq N$ such that, in exact arithmetic, Algorithm 2.4 will either stop in step $n = L$ with $\rho_{L+1} = 0$ or $\xi_{L+1} = 0$, or, starting with the regular vectors v_{L+1} and w_{L+1}, an incurable breakdown will occur. If $\rho_{L+1} = 0$ or $\xi_{L+1} = 0$, then $v_1, \dots, v_L$ or $w_1, \dots, w_L$ span the A-invariant subspace $K_L(v_1, A)$ or the A^T-invariant subspace $K_L(s_0, A^T)$, respectively. Moreover, in all cases,*

$$\lambda(H_L) \subseteq \lambda(A). \tag{2.36}$$

2.4. The look-ahead strategy

In this section, we discuss the criteria used to decide in step 1) of Algorithm 2.4 whether a pair of Lanczos vectors v_{n+1} and w_{n+1} is built as inner vectors or as regular vectors. We propose three criteria, namely (2.40)-(2.42) below. If all three checks (2.40)-(2.42) are satisfied, then v_{n+1} and w_{n+1} are constructed as regular vectors, otherwise, they are constructed as inner vectors. Let us motivate these three criteria.

First, recall (cf. (2.23)) that for v_{n+1} and w_{n+1} to be built as regular vectors it is necessary that $D^{(l)}$ is nonsingular. Therefore, it is tempting to base the decision "regular versus inner step" solely on checking whether $D^{(l)}$ is close to singular, and to perform a regular step if, and only if,

$$\sigma_{\min}(D^{(l)}) \geq tol, \tag{2.37}$$

for some suitably chosen tolerance tol. For example, Parlett [Par] suggests $tol = \epsilon^{1/4}$ or $tol = \epsilon^{1/3}$, where ϵ denotes the roundoff unit. Then (2.37) would guarantee that complete blocks of computed Lanczos vectors satisfy

$$\sigma_{\min}(D^{(k)}) \geq tol, \quad k = 1, 2, \dots .$$

This, together with (2.22), would imply by [Par, Theorem 10.1] that

$$\sigma_{\min}(V_n) \geq \frac{tol}{\sqrt{n}} \quad \text{and} \quad \sigma_{\min}(W_n) \geq \frac{tol}{\sqrt{n}}, \quad n = n_k - 1, \ k = 1, 2, \dots . \tag{2.38}$$

Since the columns of V_n and W_n are unit vectors, $\sigma_{\min}(V_n)$ and $\sigma_{\min}(W_n)$ are a measure of the linear independence of these vectors; in particular, (2.38) would ensure that the Lanczos vectors remain linearly independent. However, in the outlined algorithm, the block orthogonality (2.22) is enforced only among two or three successive blocks, and in finite precision arithmetic, biorthogonality of blocks whose indices are far apart is typically lost. The theorem assumes that (2.22) holds for all indices, and without this, the theorem fails in finite arithmetic. We illustrate this with a simple example.

Example 2.1. In Figure 2.1, we plot $\sigma_{\min}(D^{(l(n))})$ (dots), $\min_{1\leq k<l(n)}(\sigma_{\min}(D^{(k)}))$ solid line), and $\sqrt{n}\,\sigma_{\min}(V_n)$ (dotted line), as functions of the iteration index $n = 1, 2, \ldots$, for a random 50×50 dense matrix. The theorem predicts that

$$\sqrt{n}\,\sigma_{\min}(V_n) \geq \min_{1\leq k<l(n)}(\sigma_{\min}(D^{(k)})),$$

which is clearly not the case.

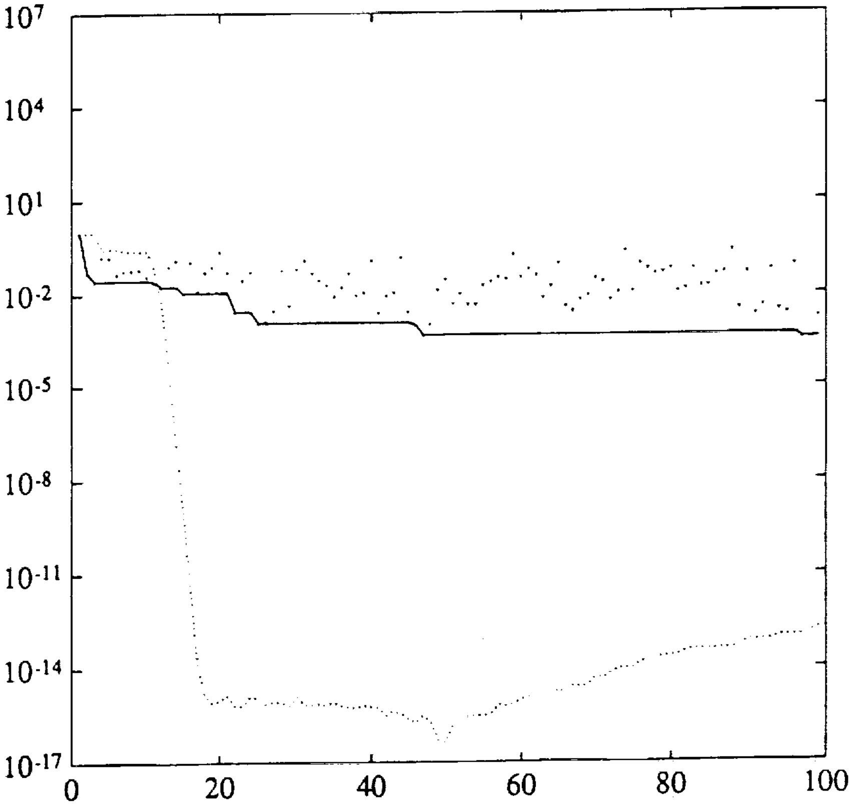

Figure 2.1. $\sigma_{\min}(D^{(l(n))})$ (dots), $\min_{1\leq k<l(n)}(\sigma_{\min}(D^{(k)}))$ (solid line), and $\sqrt{n}\,\sigma_{\min}(V_n)$ (dotted line), plotted versus the iteration index n.

As this simple example shows, the check (2.37) alone does not ensure that the computed Lanczos vectors are sufficiently linearly independent. In particular, if the look-ahead strategy is based only on criterion (2.37), the algorithm may produce, within a block, Lanczos vectors which are almost linearly dependent. When this happens, the check (2.37) usually fails in all subsequent iterations and thus the algorithm never completes the current block, *i.e.*, it has generated an *artificial* incurable breakdown.

In addition, numerical experience indicates another problem with (2.37): for values of *tol* which are "reasonably" larger than machine epsilon, the behavior of the algorithm is very sensitive with respect to the actual value of *tol*. We also illustrate this with an example.

Example 2.2. Here we consider the 3-D PDE

$$Lu = f \quad \text{on} \quad (0,1) \times (0,1) \times (0,1), \tag{2.39}$$

where

$$\begin{aligned} Lu = &- \frac{\partial}{\partial x}\left(e^{xy}\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(e^{xy}\frac{\partial u}{\partial y}\right) - \frac{\partial}{\partial z}\left(e^{xy}\frac{\partial u}{\partial z}\right) \\ &+ 30(x+y+z)\frac{\partial u}{\partial x} + \left(\frac{1}{1+x+y+z} - 250\right)u, \end{aligned}$$

with Dirichlet boundary conditions $u = 0$. The right-hand side f is chosen such that

$$u = (1-x)(1-y)(1-z)\left(1-e^{-x}\right)\left(1-e^{-y}\right)\left(1-e^{-z}\right)$$

is the exact solution of (2.39). We discretize (2.39) using centered differences on a uniform $15 \times 15 \times 15$ grid with mesh size $h = 1/16$. This leads to a linear system (1.1) with real nonsymmetric coefficient matrix A of order $N = 3375$ and 22275 nonzero elements. We applied the QMR Algorithm 3.1 based on the look-ahead Lanczos Algorithm 2.4 to this linear system. As initial guess, we used $x_0 = 0$, and, in Algorithm 2.4, $s_0 = r_0$ was chosen. This example was run on a machine with $\epsilon \approx$ 1.3E−29. In the first case, we set $tol = \epsilon^{1/4} \approx$ 6.0E−08, while in the second case, we set $tol = \epsilon^{1/3} \approx$ 2.3E−10. In Figure 2.2, we plot $\sigma_{\min}(D^{(l(n))})$ versus the iteration index n for the two runs, the dotted line for $\epsilon^{1/4}$ and the solid line for $\epsilon^{1/3}$. In the first case, the algorithm starts building a block which it never closes, and the singular values clearly become smaller and smaller. Yet if *tol* is only slightly smaller, as in the second case, the algorithm runs to completion, in this case solving the linear system to the desired accuracy, and thus indicating that the block built in the first case was not a true, but an artificial incurable breakdown. Furthermore, in the second case, the QMR approach takes $n = 149$ steps to reduce the norm of the initial residual by a factor of 10^{-6}; see Figure 2.3, where the relative residual norm $\|r_n\| \,/\, \|r_0\|$ is plotted versus n (solid line). For the run with $tol = \epsilon^{1/4} \approx$ 6.0E−08, the resulting convergence curve is shown as the dotted line in Figure 2.3. Notice that, due to the artificial incurable breakdown, QMR does not converge in this case.

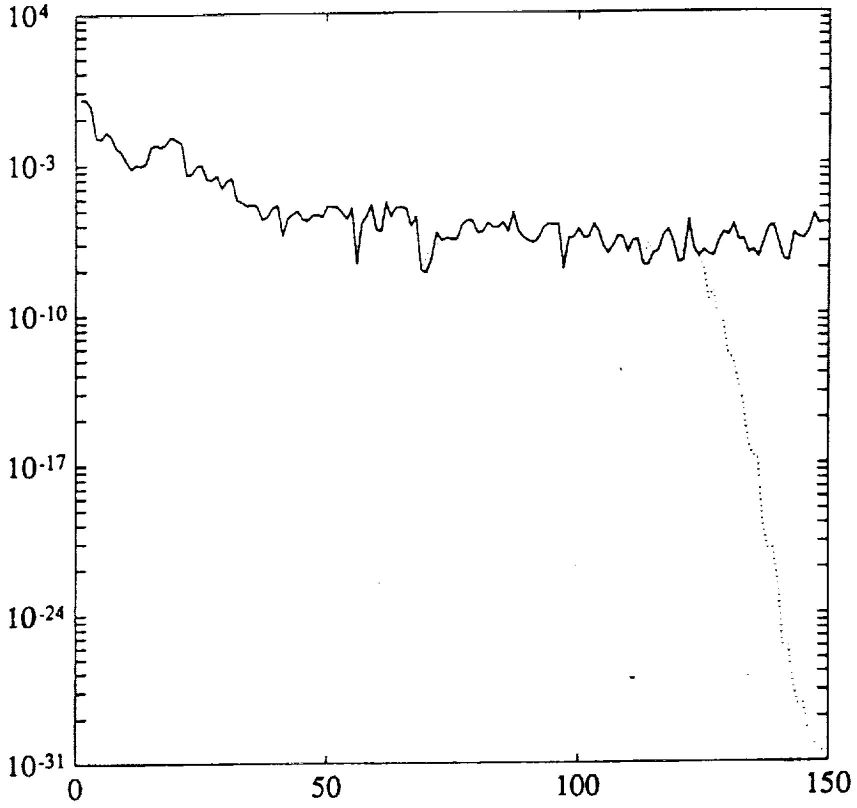

Figure 2.2. $\epsilon^{1/4}$ (dotted line) and $\epsilon^{1/3}$ (solid line), plotted versus the iteration index n.

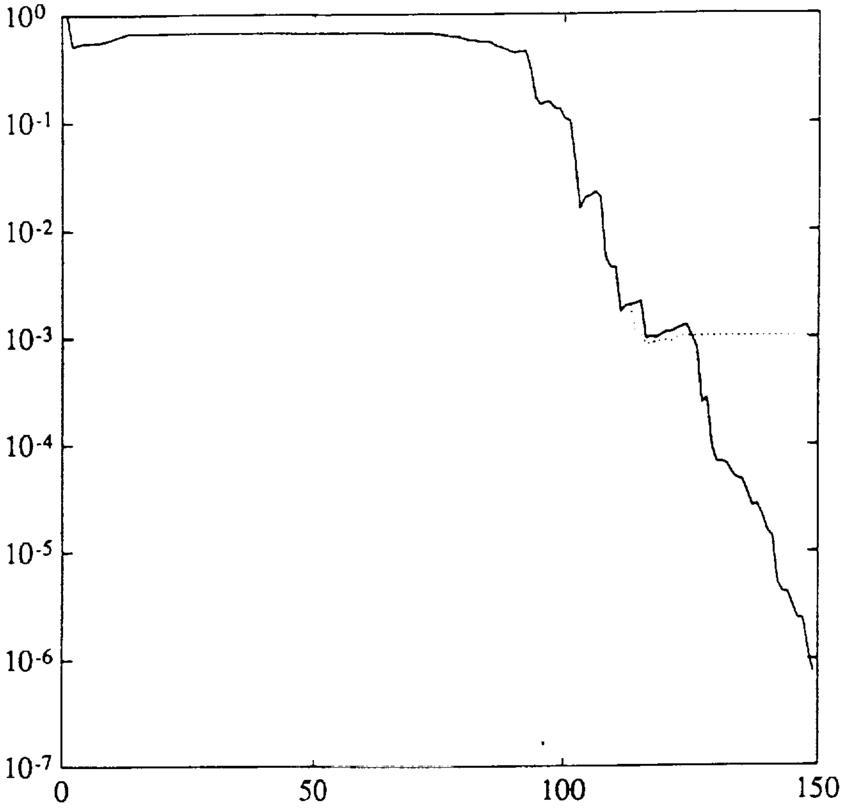

Figure 2.3. Relative residual norm $\|r_n\| \,/\, \|r_0\|$ plotted versus n.

These numerical examples clearly show that the decision "regular versus inner step" cannot be based on (2.37) alone. Instead, we propose to relax the check (2.37), so that it merely ensures that $D^{(l(n))}$ is numerically nonsingular, and to add the checks (2.41) and (2.42) below which guarantee that the computed Lanczos vectors remain sufficiently linearly independent. Hence, instead of (2.37), we check for

$$\sigma_{\min}(D^{(l(n))}) \geq \epsilon, \tag{2.40}$$

where ϵ denotes the roundoff unit.

Our numerical experiments have shown that typically the algorithm starts to generate Lanczos vectors which are almost linearly dependent, once a regular vector v_{n+1} was computed whose component $Av_n \in K_{n+1}(r_0, A)$ is dominated by its component in the previous Krylov space $K_n(r_0, A)$ (and similarly for w_{n+1}).

In order to avoid the construction of such regular vectors, we check the l_1-norm of the coefficients for $V^{(l-1)}$ and $V^{(l)}$ in (2.26); v_{n+1} can be computed as a regular vector only if

$$\begin{aligned} &\sum_{j=n_{l-1}}^{n_l-1} \left|((D^{(l-1)})^{-1}(W^{(l-1)})^T A v_n)_j\right| \leq n(A) \\ \text{and} \quad &\sum_{j=n_l}^{n} \left|((D^{(l)})^{-1}(W^{(l)})^T A v_n)_j\right| \leq n(A). \end{aligned} \tag{2.41}$$

Here $n(A)$ is a factor depending on the norm of A; we will indicate later how this factor is computed. Similarly, we check the l_1-norm of the coefficients for $W^{(l-1)}$ and $W^{(l)}$ in (2.26); w_{n+1} can be computed as a regular vector only if

$$\begin{aligned} &\sum_{j=n_{l-1}}^{n_l-1} \left|((D^{(l-1)})^{-T}(V^{(l-1)})^T A^T w_n)_j\right| \leq n(A) \\ \text{and} \quad &\sum_{j=n_l}^{n} \left|((D^{(l)})^{-T}(V^{(l)})^T A^T w_n)_j\right| \leq n(A). \end{aligned} \tag{2.42}$$

The pair v_{n+1} and w_{n+1} is built as regular vectors only if all the checks in (2.40)-(2.42) hold true.

We need to indicate how $n(A)$ is chosen in (2.41) and (2.42). Numerical experience with matrices whose norm is known indicates that setting $n(A) = \|A\|$ is too strict and can result in artificial incurable breakdowns. A better setting seems to be $n(A) = 10 \cdot \|A\|$, but even this is dependent on the matrix. In any case, in practice one does not know $\|A\|$, and there is also the issue of a maximal block size, determined by limits on available storage. To solve the problems of estimating the norms and a suitable factor $n(A)$, as well as cope with limited storage and yet allow the algorithm to proceed as far as possible, we propose the following procedure. Suppose we are given an initial value for $n(A)$, based either on an estimate from the user (for example, $n(A)$ from a previous run with the matrix A), or by setting

$$n(A) = \max\left\{\|Av_1\|, \left\|A^T w_1\right\|\right\}.$$

Note that here A denotes the matrix actually used in generating the Lanczos vectors, thus including the case when we are solving a preconditioned linear system (cf. Section 3.6). We then update $n(A)$ dynamically, as follows. In each block, whenever an inner vector is built because one of the checks (2.41) or (2.42) is not satisfied, the algorithm keeps track of the size of the terms that have caused one or more of (2.41)-(2.42) to be false. If the block closes naturally, then this information is not needed. If, however, the algorithm is about to run out of storage, then $n(A)$ is replaced with the smallest value which has caused an inner vector to be built. The updated value of $n(A)$ is guaranteed to pass all the checks in

(2.41) and (2.42) at least once, and hence the block is guaranteed to close. This also frees up the storage that was used by the previous block, thus ensuring that the algorithm can proceed.

2.5. Implementation details

We now turn to a few implementation details for Algorithm 2.4. In particular, we show that our implementation of the look-ahead Lanczos process requires the same number of inner products per step as the classical Lanczos Algorithm 2.1. For a regular step, one needs to compute $D^{(l)}$, $(W^{(l)})^T Av_n$, and $(W^{(l-1)})^T Av_n$ in (2.26). For an inner step, one needs to compute $(W^{(l-1)})^T Av_n$ in (2.27) and to update $D^{(l)}$ in (2.28). We will show that for a block of size h_l, only $2h_l$ inner products are required: $2h_l - 1$ will be required to compute $D^{(l)}$, and one inner product will be required to compute $(W^{(l)})^T Av_n$. We will obtain $(W^{(l-1)})^T Av_n$ without performing any inner products. Note that a block of size h_l in Algorithm 2.4 corresponds to h_l steps in Algorithm 2.1, which each require 2 inner products. In addition, in step 4) of the look-ahead Lanczos algorithm. Euclidean norms of 2 vectors of length N need to be computed. However, for a robust implementation of the classical Lanczos process it is also advisable to scale the Lanczos vectors v_n and w_n in Algorithm 2.1 to have unit length. cf. [Tay, PTL].

To simplify the derivations, we will use the "monic" versions

$$\hat{v}_n = \frac{1}{\phi_n} v_n = \Phi_{n-1}(A) r_0 \quad \text{and} \quad \hat{w}_n = \frac{1}{\psi_n} w_n = \Phi_{n-1}(A^T) s_0 \tag{2.43}$$

of the Lanczos vectors v_n and w_n, where $\Phi_{n-1} \in \Pi_{n-1}$ is monic and $\phi_n, \psi_n \in \mathbb{C}$. By $\hat{V}^{(l)}, \hat{D}^{(l)}, \ldots$, we denote the matrices defined as in (2.21) and (2.22), with the monic vectors instead of the original Lanczos vectors. Clearly, all quantities involving the original vectors v_n and w_n can be obtained from the corresponding quantities involving $\hat{v}_n$ and $\hat{w}_n$ simply by scaling. Finally, we remark that, using a similar argument as in (2.44) below, one easily verifies that

$$(\hat{W}^{(l)})^T A \hat{v}_n = (\hat{V}^{(l)})^T A^T \hat{w}_n \quad \text{and} \quad (\hat{W}^{(l-1)})^T A \hat{v}_n = (\hat{V}^{(l-1)})^T A^T \hat{w}_n.$$

Therefore, the coefficients $(D^{(l)})^{-T}(V^{(l)})^T A^T w_n$ and $(D^{(l-1)})^{-T}(V^{(l-1)})^T A^T w_n$, which occur in the recursions for the left Lanczos vectors in (2.26) or (2.27), can be generated from $(D^{(l)})^{-1}(W^{(l)})^T Av_n$ and $(D^{(l-1)})^{-1}(W^{(l-1)})^T Av_n$, without computing any additional inner products.

Consider first $\hat{D}^{(l)}$. Using (2.43) and the fact that polynomials in A commute, we deduce that

$$\hat{w}_j^T \hat{v}_m = s_0^T \Phi_{j-1}(A) \Phi_{m-1}(A) r_0 = s_0^T \Phi_{m-1}(A) \Phi_{j-1}(A) r_0 = \hat{w}_m^T \hat{v}_j. \tag{2.44}$$

This shows that the matrix $\hat{D}^{(l)}$ is symmetric, and hence we only need to compute its upper triangle.

We will now show that once the diagonal and first superdiagonal of $\hat{D}^{(l)}$ have been computed by inner products, the remaining upper triangle can be computed by recurrences. Let $\hat{w}_j$ and $\hat{v}_m$ be two vectors from the current block. Using (2.27) and the fact that the inner vectors from block l are biorthogonal to the vectors from the previous block, we have

$$\begin{aligned}
\hat{w}_j^T \hat{v}_m &= \hat{w}_j^T (A\hat{v}_{m-1} - \zeta_{m-1}\hat{v}_{m-1} - \eta_{m-1}\hat{v}_{m-2}) \\
&= (A^T \hat{w}_j)^T \hat{v}_{m-1} - \zeta_{m-1}\hat{w}_j^T \hat{v}_{m-1} - \eta_{m-1}\hat{w}_j^T \hat{v}_{m-2} \\
&= (\hat{w}_{j+1} + \zeta_j \hat{w}_j + \eta_j \hat{w}_{j-1})^T \hat{v}_{m-1} - \zeta_{m-1}\hat{w}_j^T \hat{v}_{m-1} - \eta_{m-1}\hat{w}_j^T \hat{v}_{m-2} \\
&= \hat{w}_{j+1}^T \hat{v}_{m-1} + \zeta_j \hat{w}_j^T \hat{v}_{m-1} + \eta_j \hat{w}_{j-1}^T \hat{v}_{m-1} - \zeta_{m-1}\hat{w}_j^T \hat{v}_{m-1} - \eta_{m-1}\hat{w}_j^T \hat{v}_{m-2}.
\end{aligned}$$

Thus, $\hat{w}_j^T \hat{v}_m$ depends only on elements of $\hat{D}^{(l)}$ from the previous two columns, and hence, with the exception of the diagonal and the first superdiagonal, can be computed without any additional inner products. Note that the recurrences and the biorthogonality used in the above derivation are enforced numerically, and so computing $\hat{w}_j^T \hat{v}_m$ by the above recurrence should give the same results – up to roundoff – as computing the inner product directly.

We will now show how to compute $(\hat{W}^{(l)})^T A\hat{v}_n$ with only one additional inner product, while $(\hat{W}^{(l-1)})^T A\hat{v}_n$ can be obtained with no additional inner products. Consider $\hat{w}_j^T A\hat{v}_n$, for $\hat{w}_j$ a vector from either the current or the previous block. We have

$$\begin{aligned}
\hat{w}_j^T A\hat{v}_n &= (A^T \hat{w}_j)^T \hat{v}_n = (\hat{w}_{j+1} + \zeta_j \hat{w}_j + \eta_j \hat{w}_{j-1})^T \hat{v}_n \\
&= \hat{w}_{j+1}^T \hat{v}_n + \zeta_j \hat{w}_j^T \hat{v}_n + \eta_j \hat{w}_{j-1}^T \hat{v}_n.
\end{aligned}$$

For $j < n_l - 1$, $(\hat{W}^{(l-1)})^T \hat{v}_n = 0$, and hence $\hat{w}_j^T A\hat{v}_n = 0$. For $j = n_l - 1$, the above reduces to $\hat{w}_{n_l-1}^T A\hat{v}_n = \hat{w}_{n_l}^T \hat{v}_n$, which is computed as part of the first row of $\hat{D}^{(l)}$. For $n_l \leq j < n_{l+1}$, all of the terms needed are available from $\hat{D}^{(l)}$. Finally, for the last vector in the current block, $j = n_{l+1} - 1$, we do not have $\hat{w}_{n_{l+1}}^T \hat{v}_n$, and hence have to compute it directly, thus requiring another inner product.

3. A quasi-minimal residual method for general non-Hermitian matrices

We now turn to linear systems (1.1). From now on, it is always assumed that A is nonsingular. Furthermore, all iterative algorithms considered in the sequel are Krylov subspace methods, *i.e.*, their iterates x_n, $n = 1, 2, \ldots$, satisfy (1.2), where $x_0 \in \mathbf{C}^N$ is any given initial guess for the exact solution $A^{-1}b$ of (1.1). Finally, $r_n = b - Ax_n$ always denotes the residual vector corresponding to the nth iterate x_n.

3.1. The quasi-minimal residual approach

In this section, we describe the basic idea of the QMR approach for solving general non-Hermitian linear systems (1.1).

We set

$$\rho_0 = \|r_0\|, \quad v_1 = r_0/\rho_0. \tag{3.1}$$

Let $v_1, v_2, \ldots, v_n$ be the right Lanczos vectors generated by Algorithm 2.4, with the normalized initial residual v_1 as one of the two starting vectors. By the first relation in (2.30), we have the parametrization

$$x_n = x_0 + V_n z, \quad z \in \mathbf{C}^n, \tag{3.2}$$

for all possible iterates (1.2). Note that the second starting vector, $w_1 \in \mathbf{C}^N$, is still unspecified. Due to the lack of a criterion for the choice of w_1, one usually sets $w_1 = v_1$ in practice.

From (3.1) and (2.31), the residual vectors corresponding to (3.2) satisfy

$$r_n = r_0 - AV_n z = r_0 - V_{n+1} H_n^{(e)} z = V_{n+1} \left(\rho_0 e_1^{(n+1)} - H_n^{(e)} z \right). \tag{3.3}$$

Next, we introduce an $(n+1) \times (n+1)$ diagonal weight matrix

$$\Omega_n = \operatorname{diag}(\omega_1, \omega_2, \ldots, \omega_{n+1}), \quad \omega_j > 0, \ j = 1, \ldots, n+1, \tag{3.4}$$

to serve as a free parameter that can be used to modify the scaling of the problem. With it, (3.3) reads

$$\begin{aligned} r_n &= V_{n+1} \Omega_n^{-1} \Omega_n \left(\rho_0 e_1^{(n+1)} - H_n^{(e)} z \right) \\ &= V_{n+1} \Omega_n^{-1} \left(d_n - \Omega_n H_n^{(e)} z \right), \quad \text{with } d_n = \omega_1 \rho_0 e_1^{(n+1)}. \end{aligned} \tag{3.5}$$

Ideally, we would like to choose $z \in \mathbf{C}^n$ in (3.5) such that $\|r_n\|$ is minimal. However, since in general V_{n+1} is not unitary, this would require $\mathcal{O}(Nn^2)$ work, which is too expensive.

We will instead minimize just the Euclidean norm of the bracketed terms in (3.5), *i.e.*, we will choose $z = z_n \in \mathbf{C}^n$ as the solution of the least squares problem

$$\left\| d_n - \Omega_n H_n^{(e)} z_n \right\| = \min_{z \in \mathbf{C}^n} \left\| d_n - \Omega_n H_n^{(e)} z \right\| . \tag{3.6}$$

By (2.35) and (3.4), $H_n^{(e)}$ and $\Omega_n H_n^{(e)}$ are $(n+1) \times n$ matrices with full column rank n. This guarantees that the solution z_n of (3.6) is unique and hence, via (3.2), defines a unique nth iterate x_n. In view of the minimization property (3.6), we refer to this iteration scheme as the *quasi-minimal residual* (QMR) method. Clearly, the QMR iterates still depend on the choice of the weights ω_j in (3.4). In our numerical experiments, the simplest scaling

$$\omega_j = 1, \quad j = 1, 2, \ldots , \tag{3.7}$$

gave satisfactory results. Recall from (2.19) that all the columns of V_{n+1} are unit vectors. Hence, the scaling (3.7) ensures that all basis vectors v_j/ω_j, $j = 1, \ldots, n+1$, in the representation (3.5) of r_n have the same Euclidean length; this is a "natural" requirement. However, better strategies for choosing Ω_n might be possible, and therefore we have formulated the QMR approach with a general scaling matrix Ω_n.

For the solution of the least squares problem (3.6), we use the standard approach (see, *e.g.*, [GVL, Chapter 6]) based on a QR decomposition of $\Omega_n H_n^{(e)}$:

$$\Omega_n H_n^{(e)} = Q_n^H \begin{bmatrix} R_n \\ 0 \end{bmatrix} . \tag{3.8}$$

Here, Q_n is a unitary $(n+1) \times (n+1)$ matrix, and R_n is a nonsingular upper triangular $n \times n$ matrix. Inserting (3.8) in (3.6) yields

$$\begin{aligned} \min_{z \in \mathbf{C}^n} \left\| d_n - \Omega_n H_n^{(e)} z \right\| &= \min_{z \in \mathbf{C}^n} \left\| Q_n^H \left(Q_n d_n - \begin{bmatrix} R_n \\ 0 \end{bmatrix} z \right) \right\| \\ &= \min_{z \in \mathbf{C}^n} \left\| Q_n d_n - \begin{bmatrix} R_n \\ 0 \end{bmatrix} z \right\| . \end{aligned}$$

Hence, z_n is given by

$$z_n = R_n^{-1} t_n, \quad \text{where} \quad t_n = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix}, \quad \begin{bmatrix} t_n \\ \tilde{\tau}_{n+1} \end{bmatrix} = Q_n d_n. \tag{3.9}$$

Furthermore, we have

$$\left\| d_n - \Omega_n H_n^{(e)} z_n \right\| = |\tilde{\tau}_{n+1}|. \tag{3.10}$$

We conclude this section by summarizing the basic structure of the QMR algorithm.

Algorithm 3.1. (QMR algorithm.)

```
0) Choose $x_0 \in \mathbf{C}^N$ and set $r_0 = b - Ax_0$. $\rho_0 = \|r_0\|$, $v_1 = r_0/\rho_0$;
   Choose $w_1 \in \mathbf{C}^N$ with $\|w_1\| = 1$;
For $n = 1, 2, \ldots$ :
1) Perform the $n$th iteration of the look-ahead Lanczos Algorithm 2.4;
   This yields matrices $V_n$, $V_{n+1}$, $H_n^{(e)}$ which satisfy (2.31);
2) Update the QR factorization (3.8) of $\Omega_n H_n^{(e)}$ and the vector $t_n$ in (3.9);
3) Compute
       $x_n = x_0 + V_n R_n^{-1} t_n;$        (3.11)
4) If $x_n$ has converged: stop.
```

3.2. Implementation details

In this section, we give some of the details for the actual implementation of steps 2), 3), and 4) of the QMR Algorithm 3.1. In particular, it is shown that the QMR iterates x_n can be computed with short recurrences. This approach for updating the iterates x_n is based on a technique which was first used by Paige and Saunders [PS] in connection with their SYMMLQ and MINRES algorithms for real symmetric matrices.

First, note that the QR decomposition (3.8) of $\Omega_n H_n^{(e)}$ can be computed by means of n Givens rotations, taking advantage of the fact that $\Omega_n H_n^{(e)}$ is an upper Hessenberg matrix. Hence, the unitary factor in (3.8) is of the form

$$Q_n = G_n \begin{bmatrix} G_{n-1} & 0 \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} G_1 & 0 \\ 0 & I_{n-1} \end{bmatrix}, \tag{3.12}$$

where, for $j = 1, 2, \ldots, n$,

$$G_j = \begin{bmatrix} I_{j-1} & 0 & 0 \\ 0 & c_j & s_j \\ 0 & -\overline{s_j} & c_j \end{bmatrix}, \quad \text{with} \quad c_j \in \mathbf{R},\ s_j \in \mathbf{C},\ c_j^2 + |s_j|^2 = 1. \tag{3.13}$$

Recall that, in view of (2.33) and (2.32), $\Omega_n H_n^{(e)}$ is block tridiagonal. Therefore, the upper triangular factor in (3.8) is of the form

$$R_n = \begin{bmatrix} \delta_1 & \epsilon_2 & \theta_3 & 0 & \cdots & 0 \\ 0 & \delta_2 & \epsilon_3 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \delta_3 & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \theta_l \\ \vdots & & & \ddots & \ddots & \epsilon_l \\ 0 & \cdots & \cdots & \cdots & 0 & \delta_l \end{bmatrix}, \tag{3.14}$$

where the blocks δ_k and ϵ_k are of the same size as the blocks α_k and β_k, respectively, in (2.33). Moreover, the diagonal blocks δ_k are nonsingular upper triangular matrices. Clearly, a QR decomposition based on unitary matrices (3.12) limits fill-in to the row above each block β_k in (2.33). Hence each of the blocks θ_k in (3.14) has possible nonzero entries only in its last row.

Next, we note that the decomposition (3.8) is easily updated from the factorization of $\Omega_{n-1}H_{n-1}^{(e)}$ of the previous step $n-1$. Indeed, to obtain R_n, one only needs to compute its last column,

$$[\mu_1 \quad \cdots \quad \mu_n]^T = R_n e_n^{(n)}, \tag{3.15}$$

and append it to R_{n-1}. This is done by first multiplying the last column of $\Omega_n H_n^{(e)}$ by the previous Givens rotations; by (2.33), this last column has zero entries in positions $1, 2, \ldots, n_G$, where

$$n_G = \begin{cases} \max(n_{l-1} - 1, 1) & \text{if } v_n \text{ is an inner vector,} \\ \max(n_{l-2} - 1, 1) & \text{if } v_n \text{ is a regular vector.} \end{cases}$$

Therefore, only the Givens rotations with indices $n_G, n_G + 1, \ldots, n-1$ have to be applied, and, by setting

$$\begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{n-1} \\ \mu \\ \nu \end{bmatrix} = \begin{bmatrix} G_{n-1} & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} G_{n-2} & 0 \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} G_{n_G} & 0 \\ 0 & I_{n_G - 1} \end{bmatrix} \Omega_n H_n^{(e)} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \tag{3.16}$$

we obtain the desired vector (3.15) up to its last component μ_n. It remains to multiply (3.16) by a suitably chosen Givens rotation G_n which zeros out the last element $\nu = \omega_{n+1}\rho_{n+1}$. To achieve this, set

$$\begin{aligned} c_n &= \frac{|\mu|}{\sqrt{|\mu|^2 + |\nu|^2}}, & \overline{s_n} &= c_n \frac{\nu}{\mu}, & &\text{if } \mu \neq 0, \\ c_n &= 0, & \overline{s_n} &= 1, & &\text{if } \mu = 0. \end{aligned} \tag{3.17}$$

and finally one gets $\mu_n = c_n\mu + s_n\nu$. For later use, we notice that

$$|s_n\mu_n| = \omega_{n+1}\rho_{n+1}, \tag{3.18}$$

which is readily verified using (3.17). The vector t_n in (3.9) is updated by setting

$$t_n = G_n \begin{bmatrix} t_{n-1} \\ 0 \end{bmatrix}.$$

Clearly, t_n differs from t_{n-1} only in its last two entries which are given by

$$\tau_n = c_n \tilde{\tau}_n \quad \text{and} \quad \tilde{\tau}_{n+1} = -\overline{s_n}\,\tilde{\tau}_n. \tag{3.19}$$

Next, we turn to the computation of the QMR iterates x_n in (3.11). We define vectors p_j via

$$P_n = [\, p_1 \quad p_2 \quad \dots \quad p_n \,] = V_n R_n^{-1}. \tag{3.20}$$

Then, with (3.11) and (3.9), it follows that

$$x_n = x_{n-1} + p_n \tau_n.$$

It remains to show how to compute p_n. In analogy to the partitioning of V_n in (2.21) and (2.29), we group the columns of P_n into blocks

$$P_n = [\, P^{(1)} \quad P^{(2)} \quad \dots \quad P^{(l)} \,]. \tag{3.21}$$

With (3.20), (3.14), and (3.21), one obtains the relation

$$P^{(l)} = \left(V^{(l)} - P^{(l-1)}\epsilon_l - P^{(l-2)}\theta_l \right) \delta_l^{-1}, \tag{3.22}$$

and thus p_n can be updated via short recurrences.

Finally, for step 4) of Algorithm 3.1, a convergence criterion is needed. We stop the QMR iteration as soon as

$$\|r_n\| \leq tol \cdot \|r_0\|; \tag{3.23}$$

here tol is a suitable tolerance, *e.g.*, $tol = 10^{-6}$. In the QMR algorithm described so far, neither the residual vectors r_n nor their norms $\|r_n\|$ are generated explicitly. However, in part a) of the next proposition, we derive an upper bound for $\|r_n\|$ which is available at no extra cost. In our implementation, the convergence criterion is checked for this upper bound, (3.24), rather than $\|r_n\|$. Once this test is satisfied, we switch over to checking (3.23) for the true residual norm $\|r_n\|$. Typically, this is necessary only in the last one or two iterations, since (3.24) is a good upper bound for $\|r_n\|$.

The residual vector itself can be easily updated at the expense of one additional SAXPY per iteration, based on the recursion given in part b) of the following

Proposition 3.2. *For $n = 1, 2, \dots$:*

a)

$$\|r_n\| \leq \|r_0\| \sqrt{n+1}\; |s_1 s_2 \cdots s_{n-1} s_n| \max_{j=1,\dots,n+1} (\omega_1/\omega_j)\,; \tag{3.24}$$

b)

$$r_n = |s_n|^2 r_{n-1} + \frac{c_n \tilde{\tau}_{n+1}}{\omega_{n+1}} v_{n+1}. \tag{3.25}$$

Proof. By taking norms in (3.5) and with (3.10), we obtain

$$\|r_n\| \le \|V_{n+1}\| \cdot \|\Omega_n^{-1}\| \cdot |\tilde{\tau}_{n+1}| \,. \tag{3.26}$$

Now, from (2.19) and (2.29), V_{n+1} has $n+1$ columns of Euclidean norm 1, and this implies

$$\|V_{n+1}\| \le \sqrt{n+1}. \tag{3.27}$$

Furthermore, by (3.4),

$$\|\Omega_n^{-1}\| \le \max_{j=1,\ldots,n+1} (1/\omega_j)\,. \tag{3.28}$$

Finally, by (3.19),

$$|\tilde{\tau}_{n+1}| = |\tilde{\tau}_1| \cdot |s_1 s_2 \cdots s_{n-1} s_n|\,, \tag{3.29}$$

where, in view of (3.9), (3.5), and (3.1),

$$\tilde{\tau}_1 = \|r_0\|\,\omega_1, \tag{3.30}$$

and by combining (3.26–3.30), one obtains the inequality (3.24).

Now we turn to part b). By inserting $z = z_n$ from (3.9) in (3.5) and using (3.8), one obtains

$$r_n = \tilde{\tau}_{n+1} y_{n+1}, \tag{3.31}$$

where

$$y_{n+1} = V_{n+1}\Omega_n^{-1} Q_n^H \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

From (3.12), one readily verifies that two successive vectors y_{n+1} and y_n in (3.31) are connected by

$$y_{n+1} = -s_n y_n + \frac{c_n}{\omega_{n+1}} v_{n+1}. \tag{3.32}$$

Finally, by inserting (3.32) in (3.31) and using the second relation in (3.19), we arrive at (3.25). □

3.3. The connection between QMR and BCG

In this section, we are concerned with the connection between QMR and BCG. In particular, it is shown that BCG iterates can be easily recovered from the QMR process.

In the BCG approach, one aims at computing iterates x_n which are characterized by the Galerkin type condition

$$w^T(b - Ax_n) = 0 \quad \text{for all} \quad w \in K_n(s_0, A^T), \quad x_n \in x_0 + K_n(r_0, A). \tag{3.33}$$

(see, *e.g.*, [Saal]). Here, $s_0 \in \mathbf{C}^N$ is any nonzero vector. Usually, one sets $s_0 = r_0$. In the classical BCG algorithm [Lan2, Fle, Jac], the iterates (3.33) are generated as follows.

Algorithm 3.3. (BCG algorithm.)

```
0) Choose $x_0 \in \mathbf{C}^N$ and set $q_0 = r_0 = b - Ax_0$;
   Choose $s_0 \in \mathbf{C}^N$, $s_0 \neq 0$, and set $\tilde{q}_0 = \tilde{r}_0 = s_0$;
For $n = 1, 2, \ldots$ :
1) Compute $\delta_n = \tilde{r}_{n-1}^T r_{n-1} / \tilde{q}_{n-1}^T A q_{n-1}$ and set $x_n = x_{n-1} + \delta_n q_{n-1}$;
   Set $r_n = r_{n-1} - \delta_n A q_{n-1}$ and $\tilde{r}_n = \tilde{r}_{n-1} - \delta_n A^T \tilde{q}_{n-1}$;
2) Compute $\rho_n = \tilde{r}_n^T r_n / \tilde{r}_{n-1}^T r_{n-1}$;
   Set $q_n = r_n + \rho_n q_{n-1}$ and $\tilde{q}_n = \tilde{r}_n + \rho_n \tilde{q}_{n-1}$;
3) If $r_n = 0$ or $\tilde{r}_n = 0$ , stop.
```

BCG is closely related to the classical nonsymmetric Lanczos algorithm. Indeed (see, *e.g.*, [Saa1]), for $n = 1, 2, \ldots$,

$$r_{n-1} = \phi_n v_n, \ \phi_n \in \mathbf{C}, \ \phi \neq 0, \quad \text{and} \quad \tilde{r}_{n-1} = \psi_n w_n, \ \psi_n \in \mathbf{C}, \ \psi \neq 0, \tag{3.34}$$

where v_n and w_n denote the vectors generated by the classical Lanczos Algorithm 2.1 with starting vectors

$$r_0 \quad \text{and} \quad s_0. \tag{3.35}$$

Unfortunately, like the Lanczos algorithm, BCG is also susceptible to breakdowns and numerical instabilities. Obviously, Algorithm 3.3 breaks down prematurely, if

$$\tilde{q}_{n-1}^T A q_{n-1} = 0, \quad \tilde{r}_{n-1} \neq 0, \ r_{n-1} \neq 0, \tag{3.36}$$

or

$$\tilde{r}_{n-1}^T r_{n-1} = 0, \quad \tilde{r}_{n-1} \neq 0, \ r_{n-1} \neq 0, \tag{3.37}$$

occurs. We will refer to (3.36) and (3.37) as *breakdown of the first* and *second kind*, respectively. In general, Galerkin iterates (3.33) need not exist for every n. This is the cause of the breakdown of the first kind. Indeed, one can show that (3.36) occurs if no BCG iterate x_n exists. Breakdowns of the second kind have a different cause: by (3.34), (3.37) is equivalent to a serious breakdown in the classical nonsymmetric Lanczos process.

Next, we rewrite the Galerkin condition (3.33) in terms of the look-ahead Lanczos Algorithm 2.4, started with the initial vectors (3.35). This yields a formulation of the BCG approach for which breakdowns of the second kind, except for ones caused by an incurable breakdown in the look-ahead Lanczos process, cannot occur. In analogy to (3.2), we use the parametrization

$$x_n = x_0 + V_n u_n, \quad u_n \in \mathbf{C}^n, \tag{3.38}$$

for the BCG iterates. Then, by (2.31), the corresponding residual vector satisfies

$$r_n = b - Ax_n = V_n (f_n - H_n u_n) - (u_n)_n \tilde{v}_{n+1}, \quad \text{with} \quad f_n = \rho_0 e_1^{(n)}. \tag{3.39}$$

By inserting (3.39) in (3.33) and using (2.30), it follows that the iterate (3.38) satisfies (3.33) if, and only if,

$$W_n^T V_n \ (f_n - H_n u_n) = (u_n)_n W_n^T \tilde{v}_{n+1}. \tag{3.40}$$

To simplify the discussion of (3.40), we will attempt to recover the BCG iterate only when the current block $l = l(n)$ in Algorithm 2.4 is complete. Therefore, in the sequel, it is always assumed that $n = n_{l+1} - 1$. This ensures that, in view of (2.22) and (2.23), the linear system (3.40) reduces to

$$H_n u_n = f_n, \tag{3.41}$$

from which we can now derive a simple criterion for the existence of the nth BCG iterate.

Proposition 3.4. *Let $n = n_{l+1} - 1$, $l = 0, 1, \ldots$. Then, the following three conditions are equivalent:*

(i) the BCG iterate x_n^{BCG} defined by (3.33) exists;

(ii) H_n is nonsingular;

(iii) $c_n \neq 0$.

Moreover, if x_n^{BCG} exists, then

$$x_n^{BCG} = x_n^{QMR} + \frac{\tau_n |s_n|^2}{c_n^2} p_n, \tag{3.42}$$

$$\left\| r_n^{BCG} \right\| = \|r_0\| \cdot |s_1 s_2 \cdots s_{n-1} s_n| \, \frac{\omega_1}{\omega_{n+1} c_n}. \tag{3.43}$$

Proof. Clearly, an nth BCG iterate exists iff the linear system (3.41) has a solution. From (3.39), (2.33), and (2.34), the extended coefficient matrix $[\, f_n \quad H_n \,]$ of (3.41) is an upper triangular matrix whose diagonal elements are all nonzero, and thus it has full row rank n. Consequently, (3.41) has'a solution iff H_n is nonsingular. This shows the equivalence of (i) and (ii).

Next, using (3.8), (2.32), and (3.12), one readily verifies that

$$Q_{n-1} \Omega_{n-1} H_n = \begin{bmatrix} I_{n-1} & 0 \\ 0 & c_n \end{bmatrix} R_n. \tag{3.44}$$

This relation implies that (ii) and (iii) are equivalent.

Now assume $c_n \neq 0$. From (3.41) and (3.44) it follows that

$$u_n = R_n^{-1} \begin{bmatrix} I_{n-1} & 0 \\ 0 & 1/c_n \end{bmatrix} Q_{n-1} \Omega_{n-1} f_n. \tag{3.45}$$

Recalling the definitions of d_n and f_n in (3.5) and (3.39), and using (3.9), we can rewrite (3.45) as follows:

$$u_n = z_n + R_n^{-1} \begin{bmatrix} 0 \\ \tilde{\tau}_n / c_n - \tau_n \end{bmatrix}. \tag{3.46}$$

By comparing (3.38) and (3.46) with (3.2) and (3.9), and by using (3.20), we obtain the relation

$$x_n^{BCG} = x_n^{QMR} + \left(\frac{\tilde{\tau}_n}{c_n} - \tau_n \right) p_n$$

which, by (3.19), is just (3.42). By inserting (3.41) in (3.39), it follows that

$$r_n^{BCG} = -(u_n)_n \tilde{v}_{n+1}. \tag{3.47}$$

From (3.47), (3.46), and (3.9), we obtain

$$\|r_n^{BCG}\| = \frac{\|\tilde{v}_{n+1}\|}{c_n} \left| \frac{\tilde{\tau}_n}{\mu_n} \right| \quad \text{where} \quad \mu_n = (R_n)_{n,n}. \tag{3.48}$$

In view of (3.18),

$$\|\tilde{v}_{n+1}\| = \rho_{n+1} = \frac{|s_n \mu_n|}{\omega_{n+1}}. \tag{3.49}$$

Then, by inserting (3.49), (3.29), and (3.30) in (3.48), we get (3.43), and this concludes the proof. □

Proposition 3.4 shows that existing BCG iterates can be recovered easily from the QMR process. By (3.43), $\|r_n^{BCG}\|$ can be computed at no extra cost from quantities which are generated in the QMR Algorithm 3.1 anyway. In particular, one may monitor $\|r_n^{BCG}\|$ during the course of the QMR iteration, and compute x_n^{BCG} via (3.42) whenever the actual BCG iterate is desired.

Finally, we remark that CGS [Son] and Bi-CGSTAB [Van] are modifications of the BCG Algorithm 3.3. In many cases, these algorithms have better convergence properties than BCG. However, neither CGS nor Bi-CGSTAB addresses the problem of breakdowns. Indeed, one can show that, in exact arithmetic, CGS as well as Bi-CGSTAB break down every time BCG does.

3.4. A convergence theorem

In this section, we derive bounds for the QMR residuals which are essentially the same as the standard bounds for GMRES. To the best of the author's knowledge, this is the first convergence result for a BCG-like algorithm for general non-Hermitian matrices.

Let L denote the termination index of the look-ahead Lanczos Algorithm 2.4, as introduced in Proposition 2.5. We remark that, in exact arithmetic, the QMR Algorithm 3.1 will also terminate in step $n = L$. For a diagonalizable matrix M, we denote by

$$\kappa(M) = \min_{X:\ X^{-1}MX \text{ diagonal}} \|X\| \cdot \|X^{-1}\|$$

the condition number for the eigenvalue problem of M (see, *e.g.*, [BBG, p.46]).

The main result of this section can then be formulated as follows.

Theorem 3.5. *Suppose that the $L \times L$ matrix H_L generated by L steps of the look-ahead Lanczos Algorithm 2.4 is diagonalizable, and set*

$$H = \Omega_{L-1} H_L \Omega_{L-1}^{-1}. \tag{3.50}$$

Then, for $n = 1, 2, \ldots, L-1$, the residual vectors of the QMR Algorithm 3.1 satisfy

$$\|r_n\| \leq \|r_0\| \kappa(H) \sqrt{n+1}\, \varepsilon_n \max_{j=1,\ldots,n+1} (\omega_1/\omega_j), \tag{3.51}$$

where

$$\varepsilon_n = \min_{\Phi \in \Pi_n :\, \Phi(0)=1} \max_{\lambda \in \lambda(A)} |\Phi(\lambda)|. \tag{3.52}$$

Moreover, if Algorithm 2.4 terminates with $\rho_{L+1} = 0$, then $x_L = A^{-1}b$ is the exact solution of $Ax = b$.

Proof. Using (3.26–3.28), (3.10), (3.5–3.6), and (3.1), one readily verifies that

$$\|r_n\| \leq \|r_0\| \sqrt{n+1}\, \vartheta_n \max_{j=1,\ldots,n+1} (\omega_1/\omega_j),$$

where ϑ_n is given by

$$\vartheta_n = \min_{z \in \mathbf{C}^n} \left\| e_1^{(n+1)} - \Omega_n H_n^{(e)} z \right\|. \tag{3.53}$$

Therefore, for the proof of (3.51), it remains to show that

$$\vartheta_n \leq \kappa(H)\, \varepsilon_n. \tag{3.54}$$

In the following, let $n \in \{1, 2, \ldots, L-1\}$ be arbitrary, but fixed. By

$$H_L = \begin{bmatrix} H_n^{(e)} & * \\ 0 & * \end{bmatrix}$$

and (3.50), we have

$$H \begin{bmatrix} z \\ 0 \end{bmatrix} = \begin{bmatrix} \Omega_n H_n^{(e)} \Omega_{n-1}^{-1} z \\ 0 \end{bmatrix} \quad \text{for all} \quad z \in \mathbf{C}^n. \tag{3.55}$$

Recall that H_L, and therefore also H, is an upper Hessenberg matrix with nonzero subdiagonal elements. This implies that

$$\left\{ \begin{bmatrix} z \\ 0 \end{bmatrix} \;\middle|\; z \in \mathbf{C}^n \right\} = \{\Phi(H) e_1^{(L)} \mid \Phi \in \Pi_{n-1}\}. \tag{3.56}$$

Using (3.55–3.56), we can rewrite (3.53) as follows:

$$\vartheta_n = \min_{z \in \mathbf{C}^n} \left\| e_1^{(L)} - H \begin{bmatrix} z \\ 0 \end{bmatrix} \right\| = \min_{\Phi \in \Pi_n : \Phi(0)=1} \left\| \Phi(H) e_1^{(L)} \right\|. \qquad (3.57)$$

H_L is assumed diagonalizable, so, by (3.50), H is also diagonalizable, and by expanding $e_1^{(L)}$ into any set of eigenvectors of H, we deduce from (3.57) that

$$\vartheta_n \leq \kappa(H) \min_{\Phi \in \Pi_n : \Phi(0)=1} \max_{\lambda \in \lambda(H)} |\Phi(\lambda)|. \qquad (3.58)$$

By (3.50) and (2.36), we have $\lambda(H) = \lambda(H_L) \subseteq \lambda(A)$, and thus (3.58) is equivalent to the desired inequality (3.54).

Finally, we need to show that $x_L = A^{-1}b$, if Algorithm 2.4 terminates with $\rho_{L+1} = 0$. For $n = L$ and $\rho_{L+1} = 0$, the least squares problem (3.6) reduces to a linear system with coefficient matrix $\Omega_{L-1} H_L$. Since A is nonsingular, by (2.36), this linear system is nonsingular, and hence it can be solved exactly. Therefore, $r_L = 0$ and this concludes the proof. □

Recall (cf. Proposition 2.5) that, in exact arithmetic, it can also happen that the QMR algorithm terminates with $\rho_{L+1} \neq 0$. In this case, one restarts the QMR method, using the last available QMR iterate as the new initial guess. Theorem 3.5 shows that x_{L-1} is a good choice. However, the finite termination property of the look-ahead Lanczos Algorithm 2.4 is usually lost in finite precision arithmetic. In particular, situations where the QMR algorithm needs to be restarted are very rare in practice.

We remark that for the "natural" scaling $\omega_j \equiv 1$, the bound (3.51) simplifies somewhat.

Next, we contrast the bounds (3.51) for QMR with the standard bounds [SS2] for GMRES. Assume that A is a diagonalizable matrix. Then, the residuals r_n^{GMRES} generated by the GMRES algorithm (without restarts) satisfy

$$\|r_n^{GMRES}\| \leq \|r_0\| \kappa(A) \varepsilon_n, \quad n = 1, 2, \ldots ,$$

where, as before, ε_n is given by (3.52). Hence, up to the slow growing factor $\sqrt{n+1}$ in (3.51) and different constants, the error bounds for QMR and GMRES are essentially the same.

In general, simple upper bounds for (3.52) are known only for special cases. For example, assume that the eigenvalues are contained in an ellipse in the complex plane which does not contain the origin:

$$\lambda(A) \subset \mathcal{E}, \quad 0 \notin \mathcal{E}.$$

Let $f_1 \neq f_2$ denote the two foci of $\mathcal{E}$. The ellipse can be represented in the form

$$\mathcal{E} = \left\{ \lambda \in \mathbf{C} \;\middle|\; |\lambda - f_1| + |\lambda - f_2| \leq \frac{|f_1 - f_2|}{2} \left(r + \frac{1}{r} \right) \right\} \quad \text{with} \quad r > 1.$$

Moreover, let R be the unique solution of

$$\frac{1}{2}\left(R + \frac{1}{R} \right) = \frac{|f_1| + |f_2|}{|f_1 - f_2|}, \quad R > 1.$$

The linear transformation

$$z = z(\lambda) = \frac{2\lambda - f_1 - f_2}{f_1 - f_2}$$

maps $\mathcal{E}$ onto the ellipse

$$\mathcal{E}_r = \left\{ z \in \mathbf{C} \;\middle|\; |z - 1| + |z + 1| \leq r + \frac{1}{r} \right\} \tag{3.59}$$

and the origin 0 in the λ-plane onto a point $a \in \partial\mathcal{E}_R$ on the boundary of $\mathcal{E}_R$ in the z-plane. Here, $\mathcal{E}_R$ is the ellipse defined as in (3.59), with r replaced by R. Clearly, $0 \notin \mathcal{E}$ implies $R > r$. Then, by applying Theorem 3.6 below, we obtain the following upper bound for (3.52):

$$\varepsilon_n \leq \frac{r^n + 1/r^n}{R^n + 1/R^n}, \quad n = 1, 2, \ldots .$$

Theorem 3.6. *Let $r \geq 1$, $a \in \partial\mathcal{E}_R$, $R > r$. Then,*

$$\min_{\Phi \in \Pi_n : \Phi(0) = 1} \max_{z \in \mathcal{E}_r} |\Phi(z)| \leq \frac{r^n + 1/r^n}{R^n + 1/R^n}, \quad n = 1, 2, \ldots . \tag{3.60}$$

The upper bound (3.60) is due to Fischer and Freund [FF, Theorem 2]. Furthermore, in [FF] it is shown that equality holds in (3.60), if $r > 1$ and R is not "too close" to r.

3.5. QMR for shifted matrices

In this section, we are concerned with situations where A is given as a shifted matrix of the form

$$A = M + \sigma I, \quad M \in \mathbf{C}^{N \times N}, \quad \sigma \in \mathbf{C}. \tag{3.61}$$

Obviously, one has

$$K_n(r_0, A) = K_n(r_0, M) \quad \text{and} \quad K_n(s_0, A^T) = K_n(s_0, M^T), \quad n = 1, 2, \ldots, \tag{3.62}$$

and it is easily verified that the look-ahead Lanczos Algorithm 2.4 applied to A or M indeed generates identical basis vectors for the Krylov subspaces (3.62), provided the recurrence coefficients ζ_n in (2.27) are shifted correspondingly. More precisely, we have

Proposition 3.7. *Let v_n and w_n (respectively $\mathring{v}_n$ and $\mathring{w}_n$) be the Lanczos vectors generated by Algorithm 2.4 applied to M (respectively $A = M + \sigma I$) with recurrence coefficients ζ_n and η_n (respectively $\mathring{\zeta}_n = \zeta_n + \sigma$ and $\mathring{\eta}_n = \eta_n$). Then, the termination index L (cf. Proposition 2.5) is the same in both cases, and*

$$\mathring{v}_n = v_n \quad \text{and} \quad \mathring{w}_n = w_n, \quad n = 1, 2, \ldots, L.$$

Furthermore, for $n = 1, 2, \ldots, L$,

$$AV_n = V_{n+1} H_n^{(e)}(\sigma), \quad H_n^{(e)}(\sigma) := H_n^{(e)} + \sigma \begin{bmatrix} I_n \\ 0 \end{bmatrix}, \tag{3.63}$$

where $H_n^{(e)}$ denotes the upper Hessenberg matrix (2.32) generated by Algorithm 2.4 applied to M.

Now suppose we want to solve, using the QMR method, m shifted linear systems

$$(M + \sigma_j I) x^{(j)} = b, \quad j = 1, 2, \ldots, m, \tag{3.64}$$

which differ only in the shifts σ_j. In view of Proposition 3.7, all m runs of the QMR Algorithm 3.1 can be based on only one run of the look-ahead Lanczos Algorithm 2.4 (applied to M).

A sketch of the resulting QMR process for solving (3.64) is as follows.

Algorithm 3.8. (QMR algorithm for solving m shifted systems (3.64).)

```
0) For $j = 1, 2, \ldots, m$, set $x_0^{(j)} = 0$ and $r_0^{(j)} = b$;
   Set $\rho_0 = \|b\|$, $v_1 = b/\rho_0$;
   Choose $w_1 \in \mathbf{C}^N$ with $\|w_1\| = 1$;
For $n = 1, 2, \ldots$ :
1) Perform the $n$th iteration of the look-ahead Lanczos Algorithm 2.4 applied to $M$;
   This yields matrices $V_n$, $V_{n+1}$, $H_n^{(e)}$ which satisfy $MV_n = V_{n+1} H_n^{(e)}$;
2) For all $j = 1, 2, \ldots, m$ for which $x_n^{(j)}$ has not converged yet :
         Update the QR factorization
               $\Omega_n H_n^{(e)}(\sigma_j) = (Q_n^{(j)})^H \begin{bmatrix} R_n^{(j)} \\ 0 \end{bmatrix}$
         of $\Omega_n H_n^{(e)}(\sigma_j)$ and the vector $t_n^{(j)}$ (cf. (3.9));
         Compute
               $x_n^{(j)} = x_0^{(j)} + V_n (R_n^{(j)})^{-1} t_n^{(j)}$;
3) If all $x_n^{(j)}$ have converged: stop.
```

Finally, we recall (cf. (1.9)) that, for typical application which lead to shifted systems (3.64), the matrix M and the right-hand side b are real, and only the shifts σ_j in (3.64) are in general complex. Obviously, the Lanczos vectors generated within Algorithm 3.S are all real then, as long as one chooses $w_1 \in \mathbf{R}^N$ and. in (2.27) real recurrence coefficients ζ_n and η_n. Therefore, even in the case of complex shifts, no complex quantities occur in step 1) of Algorithm 3.8.

3.6. Preconditioned QMR

As for other conjugate gradient type methods, for solving realistic problems, it is crucial to combine the QMR algorithm with an efficient preconditioning technique. In this section. we show how to incorporate preconditioners into the QMR algorithm.

Let M be a given nonsingular $N \times N$ matrix which approximates in some sense the coefficient matrix A of the linear system (1.1), $Ax = b$. Moreover, assume that M is decomposed in the form

$$M = M_1 M_2. \tag{3.65}$$

Instead of solving the original system (1.1), we apply the QMR algorithm to the equivalent linear system

$$A'y = b', \quad \text{where} \quad A' = M_1^{-1} A M_2^{-1}, \; b' = M_1^{-1}(b - Ax_0), \; y = M_2(x - x_0). \tag{3.66}$$

Here x_0 denotes some initial guess for the solution of $Ax = b$. The iterates y_n and residual vectors $r'_n = b' - A'y_n$ for the preconditioned system (3.66) are transformed back into the corresponding quantities for the original system by setting

$$x_n = x_0 + M_2^{-1} y_n \quad \text{and} \quad r_n = M_1 r'_n. \tag{3.67}$$

For the special cases $M_1 = I$ or $M_2 = I$ in (3.65) one obtains right or left preconditioning. respectively.

Using (3.67), the QMR Algorithm 3.1 combined with preconditioning can be sketched as follows.

Algorithm 3.9. (QMR approach with preconditioning.)

```
0) Choose $x_0 \in \mathbf{C}^N$ and set $r'_0 = M_1^{-1}(b - Ax_0)$, $\rho_0 = \|r'_0\|$, $v_1 = r'_0/\rho_0$, $y_0 = 0$;
   Choose $w_1 \in \mathbf{C}^N$ with $\|w_1\| = 1$;
For $n = 1, 2, \ldots$ :
1) Perform the nth iteration of the look-ahead Lanczos Algorithm 2.4 (applied
   to $A'$);
   This yields matrices $V_n$, $V_{n+1}$, $H_n^{(e)}$ which satisfy $A'V_n = V_{n+1} H_n^{(e)}$;
2) Update the QR factorization (3.8) of $\Omega_n H_n^{(e)}$ and the vector $t_n$ in (3.9);
3) Compute $y_n = V_n R_n^{-1} t_n$;
4) If $y_n$ has converged: compute $x_n = x_0 + M_2^{-1} y_n$, and stop.
```

In the case of right or left preconditioning, Algorithm 3.9 simplifies somewhat. In general, however, for the QMR algorithm applied to a preconditioned system, one has to be able to compute $M_1^{-1}z$, $M_1^{-T}z$, $M_2^{-1}z$, and $M_2^{-T}z$, for arbitrary vectors z.

4. Lanczos methods for complex symmetric matrices

In this chapter, we consider the QMR method and related algorithms for the special case of complex symmetric matrices. Throughout this chapter, it is assumed that $A = A^T$.

4.1. The Lanczos recursion for complex symmetric matrices

As already pointed out by Lanczos [Lan4, p. 176], work and storage of the classical Lanczos Algorithm 2.1 can be halved if A is Hermitian respectively complex symmetric, by choosing starting vectors $s_0 = \overline{r}_0$ respectively $s_0 = r_0$. The resulting Hermitian Lanczos method has been studied extensively (see [GVL, 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 [MF] and Cullum and Willoughby [CW, Chapter 6]). Here, we hope to convince the reader that the complex symmetric Lanczos algorithm, especially combined with look-ahead, is also very useful for solving linear systems.

Obviously, if one chooses $s_0 = r_0$ and sets $\gamma_n = \beta_n$ in Algorithm 2.1, then all left and right Lanczos vectors coincide, *i.e.*, $v_n = w_n$. Hence, Algorithm 2.1 reduces to the following procedure.

Algorithm 4.1. (Classical Lanczos method for $A = A^T$.)

```
0) Choose $r_0 \in \mathbf{C}^N$ with $r_0 \neq 0$;
   Set $\tilde{v}_1 = r_0$, $v_0 = 0$;
For $n = 1, 2, \ldots$ :
1) Compute $\beta_n = \left(\tilde{v}_n^T \tilde{v}_n\right)^{1/2}$;
   If $\beta_n = 0$: set $L = n - 1$ and stop;
2) Otherwise, set $v_n = \tilde{v}_n / \beta_n$;
3) Compute $\alpha_n = v_n^T A v_n$;
   Set $\tilde{v}_{n+1} = A v_n - \alpha_n v_n - \beta_n v_{n-1}$.
```

For the special case of Algorithm 4.1, the properties (2.5) and (2.6) in Proposition 2.2 reduce to:

$$v_k^T v_n = \begin{cases} 0, & \text{if } k \neq n, \\ 1, & \text{if } k = n. \end{cases} \qquad k, n = 1, 2, \ldots, L, \tag{4.1}$$

and

$$K_n(r_0, A) = \operatorname{span}\{v_1, v_2, \ldots, v_n\}, \quad n = 1, 2, \ldots, L. \tag{4.2}$$

Notice that (4.1) and (4.2) just state that the Lanczos vectors $v_1, \ldots, v_n$ form an orthonormal basis for $K_n(r_0, A)$ with respect to the (non-Hermitian) inner product

$$(x, y) := y^T x, \quad x, y \in \mathbf{C}^N. \tag{4.3}$$

We remark that (4.3) is the proper (cf. Craven [Cra]) "inner product" for complex symmetric matrices. Unfortunately, it has the defect that there exist vectors $v \in \mathbb{C}^N$ which are *quasi-null* [Cra], *i.e.*, $(v,v) = 0$, but $v \neq 0$. Consequently, as in the case of the general classical Lanczos Algorithm 2.1, exact and near-breakdowns in the complex symmetric Lanczos Algorithm 4.1 cannot be excluded. Indeed, in view of (2.8), an exact breakdown occurs if, and only if, one encounters a quasi-null vector $\tilde{v}_n$.

Therefore, as in the case of general non-Hermitian matrices, in order to obtain a stable implementation of the complex symmetric Lanczos process, one needs to use a look-ahead variant of the method. Clearly, for complex symmetric A and with identical starting vectors $r_0 = s_0$, the left and right Lanczos vectors generated by the look-ahead Lanczos algorithm coincide. In particular, as in the case of Algorithm 4.1, work and storage for the complex symmetric variant is only half of that of the look-ahead Lanczos Algorithm 2.4 for general non-Hermitian matrices.

A sketch of the resulting complex symmetric look-ahead Lanczos process is then as follows.

Algorithm 4.2. (Sketch of the look-ahead Lanczos process for $A = A^T$.)

```
0) Choose $r_0 \in \mathbb{C}^N$ with $r_0 \neq 0$;
   Set $v_1 = r_0/\|r_0\|$;
   Set $V^{(1)} = v_1$, $D^{(1)} = (V^{(1)})^T V^{(1)}$;
   Set $n_1 = 1$, $l = 1$, $v_0 = 0$, $V_0 = \emptyset$, $\rho_1 = 1$;
For $n = 1, 2, \ldots$ :
1) Decide whether to construct $v_{n+1}$ as a regular or an inner vector
   and go to 2) or 3), respectively;
2) (Regular step.) Compute
      $\tilde{v}_{n+1} = Av_n - V^{(l)}(D^{(l)})^{-1}(V^{(l)})^T Av_n - V^{(l-1)}(D^{(l-1)})^{-1}(V^{(l-1)})^T Av_n$,
   set $n_{l+1} = n+1$, $l = l+1$, $V^{(l)} = \emptyset$, and go to 4);
3) (Inner step.) Compute
      $\tilde{v}_{n+1} = Av_n - \zeta_n v_n - (\eta_n/\rho_n) v_{n-1} - V^{(l-1)}(D^{(l-1)})^{-1}(V^{(l-1)})^T Av_n$,
4) Compute $\rho_{n+1} = \|\tilde{v}_{n+1}\|$;
   If $\rho_{n+1} = 0$: stop;
   Otherwise, set
      $v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}$,   $V^{(l)} = [\, V^{(l)} \quad v_{n+1} \,]$,   $D^{(l)} = (V^{(l)})^T V^{(l)}$.
```

We conclude this section with a result which further clarifies the connection of the complex symmetric Algorithm 4.1 with the general classical Lanczos Algorithm 2.1. First, recall that, unlike Hermitian matrices, complex symmetric matrices do not have any special spectral properties. Indeed (see, *e.g.*, [HJ, Theorem 4.4.9]), any complex $N \times N$ matrix is similar to a complex symmetric matrix. This result entails that the classical nonsymmetric Lanczos Algorithm 2.1 differs from the complex symmetric Algorithm 4.1 only in the additional starting vector s_0 which can be chosen independently of r_0 in Algorithm 2.1. A strict statement of this correspondence is given in the following

Theorem 4.3. *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:*

$$M = XAX^{-1} \quad \textit{where } X \textit{ is nonsingular.} \tag{4.4}$$

b) *Set $\mathring{r}_0 = X^{-1}r_0$ and $s_0 = X^{-T}\mathring{r}_0$. Let v_n, w_n, α_n, β_n, γ_n respectively $\mathring{v}_n$, $\mathring{\alpha}_n$, $\mathring{\beta}_n$ be the quantities generated by Algorithm 2.1 (applied to M and started with r_0, s_0) respectively Algorithm 4.1 (applied to A and started with $\mathring{r}_0$). Let L denote the termination index for Algorithm 4.1. Then, for $n = 1, 2, \ldots, L$:*

$$\mathring{v}_n = \Big(\prod_{j=1}^{n} \frac{\gamma_j}{\mathring{\beta}_j}\Big) X^{-1} v_n = \Big(\prod_{j=1}^{n} \frac{\beta_j}{\mathring{\beta}_j}\Big) X^T w_n, \quad \mathring{\alpha}_n = \alpha_n, \quad (\mathring{\beta}_n)^2 = \beta_n \gamma_n. \tag{4.5}$$

Proof. Only part b) remains to be proved. First, by means of (4.4), we rewrite Algorithm 2.1 in terms of $A, X^{-1}v_n, X^T w_n$. By comparing the resulting iteration with Algorithm 4.1 and using induction on n, one readily verifies (4.5). □

4.2. A theorem on incurable breakdowns

As seen in the previous section, complex symmetry of a matrix is not enough to exclude breakdowns in the classical Lanczos process. However, it is possible to use the complex symmetric structure to derive a criterion for the occurrence of incurable breakdowns.

In the following, it is assumed that A is diagonalizable. Then (see, *e.g.*, [HJ, Theorem 4.4.13]), A has a complete set of orthonormal (with respect to (4.3)) eigenvectors. In particular, r_0 can be expanded into eigenvectors of A. More precisely, by collecting components corresponding to identical eigenvalues, we get

$$r_0 = \sum_{l=1}^{L_*} \rho_l u_l \tag{4.6}$$
$$\text{where } \rho_l \neq 0, \; Au_l = \lambda_l u_l, \text{ and, if } l \neq j, \; \lambda_l \neq \lambda_j, \; u_l^T u_j = 0.$$

Here, $L_* = L_l = L_r$ is just the grade of $r_0 = s_0$ with respect to A, as defined in (2.3) and (2.4)

Notice that, unless all eigenvalues of A are distinct, quasi-null vectors u_l may occur in (4.6). In view of the following theorem, this is equivalent to an incurable breakdown. Recall from the discussion in Section 2.2 that an incurable breakdown occurs if, and only if, $n_J < L_*$ in (2.15).

Theorem 4.4. *Let $A = A^T$ be a diagonalizable $N \times N$ matrix and $r_0 \in \mathbf{C}^N$. Then, no incurable breakdown can occur in Algorithm 4.2 if, and only if, the eigenvectors in the expansion (4.6) of r_0 satisfy*

$$u_l^T u_l \neq 0 \quad \textit{for all} \quad l = 1, \ldots, L_*. \tag{4.7}$$

Proof. We need to show that (4.7) is equivalent to the existence of a regular FOP of degree $L_* - 1$ with respect to the inner product (2.11) (where now $s_0 = r_0$). By part b) of Proposition 2.3, a regular FOP of degree m exists iff the corresponding moment matrix $M_m := (\mu_{j+l})_{j,l=0,\ldots,m-1}$ is nonsingular. By (2.11) and (4.6), we have

$$\mu_j = r_0^T A^j r_0 = \sum_{l=1}^{L_*} \rho_l^2 \lambda_l^j u_l^T u_l, \quad j = 0, 1, \ldots. \tag{4.8}$$

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

$$\operatorname{rank} M_\infty = \operatorname{rank} M_m = \operatorname{rank} M_{L-1} = L \quad \text{for all} \quad m \geq L - 1, \tag{4.9}$$

where L is the number of poles of the rational function

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

Using (4.8) and $\sum_{j=0}^{\infty} \lambda_l^j / z^{j+1} \equiv 1/(z - \lambda_l)$, one obtains the following expansion of f:

$$f(z) = \sum_{l=1}^{L_*} \frac{\rho_l^2 u_l^T u_l}{z - \lambda_l} \quad \text{for all} \quad |z| > \max_{l=1,\ldots,L_*} |\lambda_l|. \tag{4.10}$$

In particular, by (4.10), $L \leq L_*$ with equality holding iff (4.7) holds true. Hence, in view of (4.9), M_{L_*-1} is nonsingular iff (4.7) is fulfilled. This concludes the proof. □

As mentioned, (4.7) is guaranteed if A has only simple eigenvalues. Thus we have the following

Corollary 4.5. *If $A = A^T$ is an $N \times N$ matrix with N distinct eigenvalues, then incurable breakdowns cannot occur in the complex symmetric look-ahead Lanczos Algorithm 4.2.*

4.3. QMR and related algorithms for complex symmetric matrices

For the QMR approach, one can exploit the complex symmetry of A by setting

$$s_0 = r_0 \tag{4.11}$$

and basing it on the complex symmetric look-ahead Lanczos Algorithm 4.2. We stress that, due to the lack of a criterion for the choice of s_0, one usually sets $s_0 = r_0$ anyway. A sketch of the resulting complex symmetric QMR method is as follows.

Algorithm 4.6. (QMR algorithm for $A = A^T$.)

```
    0) Choose $x_0 \in \mathbf{C}^N$ and set $r_0 = b - Ax_0$, $\rho_0 = \|r_0\|$, $v_1 = r_0/\rho_0$;
    For $n = 1, 2, \ldots$ :
    1) Perform the $n$th iteration of the complex symmetric look-ahead
       Lanczos Algorithm 4.2. This yields matrices $V_n$, $V_{n+1}$, $H_n^{(e)}$
       which satisfy $AV_n = V_{n+1}H_n^{(e)}$;
    2) Update the QR factorization (3.8) of $\Omega_n H_n^{(e)}$ and the vector $t_n$ in (3.9);
    3) Compute $x_n = x_0 + V_n R_n^{-1} t_n$;
    4) If $x_n$ has converged: stop.
```

Due to the savings for the complex symmetric Lanczos Algorithm 4.2, work and storage requirements for Algorithm 4.6 are also roughly halved, compared to the general QMR Algorithm 3.1. In particular, Algorithm 4.6 only requires one matrix-vector product $A \cdot v$ per iteration, as opposed to the two products $A \cdot v$ and $A^T \cdot w$ per iteration for the QMR approach for complex nonsymmetric matrices.

Obviously, the complex symmetric QMR Algorithm 4.6 can also be used in conjunction with a preconditioner (cf. Section 3.6). Again, work and storage per iteration is roughly halved. provided one chooses a complex symmetric preconditioner M decomposed in the form

$$M = M_1 M_2 \quad \text{where} \quad M_2 = M_1^T \tag{4.12}$$

in (3.65). Note that standard techniques, such as incomplete factorization [MvdV] or SSOR preconditioning (see, *e.g.*, [FN1]), applied to $A = A^T$ generate complex symmetric preconditioners which satisfy (4.12).

Finally, we remark that a simpler variant of the complex symmetric QMR method, based on the classical Lanczos Algorithm 4.1 rather than the look-ahead Lanczos Algorithm 4.2, is discussed in detail in the author's paper [Fre4].

In analogy to the complex symmetric variant, Algorithm 4.1, of the classical Lanczos Algorithm 2.1, the general BCG Algorithm 3.3 reduces to a scheme which requires only half the work and storage, if the starting vectors are chosen as in (4.11). The resulting procedure is as follows.

Algorithm 4.7. (BCG for $A = A^T$.)

```
0) Choose $x_0 \in \mathbf{C}^N$;
   Set $q_0 = r_0 = b - Ax_0$;
For $n = 1, 2, \ldots$ :
1) Compute $\delta_n = r_{n-1}^T r_{n-1} / q_{n-1}^T A q_{n-1}$ and set $x_n = x_{n-1} + \delta_n q_{n-1}$;
   Set $r_n = r_{n-1} - \delta_n A q_{n-1}$;
2) Compute $\rho_n = r_n^T r_n / r_{n-1}^T r_{n-1}$;
   Set $q_n = r_n + \rho_n q_{n-1}$;
3) If $r_n = 0$: stop.
```

However, as for the complex symmetric Lanczos Algorithm 4.1, breakdowns in Algorithm 4.7 cannot be excluded. Indeed, both kinds of breakdowns described in Section 3.3 can occur in the complex symmetric BCG method.

Closely related to the BCG method for general linear systems (1.1) is the conjugate gradients squared algorithm (CGS) due to Sonneveld [Son].

Algorithm 4.8. (CGS for general A.)

```
0) 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 $s_0^T r_0$.
For $n = 1, 2, \ldots$ :
1) Compute $\alpha_k = s_0^T r_{k-1} / s_0^T A p_{k-1}$ and set $q_k = u_{k-1} - \alpha_k A p_{k-1}$;
   Set $x_k = x_{k-1} + \alpha_k (u_{k-1} + q_k)$ and $r_k = r_{k-1} - \alpha_k A (u_{k-1} + q_k)$;
2) Compute $\beta_k = s_0^T r_k / s_0^T r_{k-1}$;
   Set $u_k = r_k + \beta_k q_k$ and $p_k = u_k + \beta_k (q_k + \beta_k p_{k-1})$;
3) If $r_n = 0$: stop.
```

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 cannot exploit the complex symmetry of A. In particular, for $A = A^T$, Algorithm 4.8 requires per iteration about twice as much work as the QMR and BCG Algorithms 4.6 and 4.7.

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

Proposition 4.9. *Let $A = A^T$, $r_0 = r_0^{BCG} = r_0^{CGS}$, and, in Algorithm 4.8, $s_0 = r_0$. Then, for $n = 0, 1, \ldots$,*

$$r_n^{BCG} = \Phi_n(A) r_0 \quad \text{and} \quad r_n^{CGS} = \left(\Phi_n(A)\right)^2 r_0$$

for some $\Phi_n \in \Pi_n$ with $\Phi_n(0) = 1$.

5. CG-type algorithms and polynomial preconditioning for shifted Hermitian matrices

In this chapter, we consider CG-type methods for the solution of linear systems (1.1) with coefficient matrices of the form

$$A = T + i\sigma I \quad \text{where} \quad T = T^H \quad \text{is Hermitian,} \quad \sigma \in \mathbf{R}. \tag{5.1}$$

Clearly, by multiplication of the right-hand side b or the unknown vector x by $e^{-i\theta}$ the more general case (1.4) can always be reduced to (5.1). Although our main interest is in non-Hermitian A, we include the case $\sigma = 0$ and assume that $A = T$ is nonsingular then. This guarantees that A is always nonsingular, and the exact solution of $Ax = b$ is denoted by $x_\star = A^{-1}b$. Most of the results in this chapter are taken from the author's paper [Fre3] on shifted Hermitian matrices.

5.1. Three CG-type approaches

We will consider three different CG-type approaches. Recall (cf. Section 1.2) that, for shifted Hermitian matrices, it is possible to have an ideal CG-like method with iterates characterized by the minimal residual (MR hereafter) property (1.3). The first approach we study is the MR method based on (1.3). The second scheme is the GAL method which aims at computing approximations x_n defined by the Galerkin (GAL hereafter) (or orthogonal error [FM2]) condition

$$v^H(b - Ax_n) = 0 \quad \text{for all} \quad v \in K_n(r_0, A), \quad x_n \in x_0 + K_n(r_0, A). \tag{5.2}$$

Note that, for Hermitian positive definite A, this method is equivalent to the classical CG algorithm (see, *e.g.*, [PS]). While MR and GAL are standard approaches for non-Hermitian matrices, the third method we propose is less conventional. Its iterates are defined by the minimal Euclidean error (ME) property

$$\|x_\star - x_n\| = \min_{x \in x_0 + K_n(A^H r_0, A)} \|x_\star - x\|, \quad x_n \in x_0 + K_n(A^H r_0, A). \tag{5.3}$$

Note that for the Krylov subspace in (5.3) one has the identity

$$K_n(A^H r_0, A) = A^H K_n(r_0, A), \tag{5.4}$$

since matrices (5.1) are normal and thus

$$AA^H = A^H A. \tag{5.5}$$

We remark that MINRES and SYMMLQ [PS] are numerically stable implementations of the MR and GAL methods, respectively, for real symmetric matrices A. If A is indefinite, a Galerkin iterate satisfying (5.2) need not exist for every n. Paige and Saunders resolve this problem in SYMMLQ by actually working with a sequence of well-defined auxiliary vectors from which the existing Galerkin iterates can then be computed in a stable manner. The ME approach (5.3) is a generalization of Fridman's method [Fri] for real symmetric matrices A. However, the algorithm he proposed is numerically unstable (see [Fre1, SF] for an explanation of the instability and a simple remedy). Fletcher [Fle] showed that the sequences of the Fridman iterates and the auxiliary vectors generated by SYMMLQ are mathematically equivalent. Therefore, as a by-product, SYMMLQ also yields a numerically stable implementation of Fridman's method.

We now turn to the derivation of algorithms, modeled after SYMMLQ and MINRES, for the actual computation of the iterates defined by (1.3), (5.2), and (5.3). The main ingredient is the Hermitian Lanczos algorithm [Lan1] applied to the Hermitian part T of (5.1) and with r_0 as starting vector.

Algorithm 5.1. (Hermitian Lanczos method.)

```
0) Set $\tilde{v}_1 = r_0$, $v_0 = 0$;
For $n = 1, 2, \ldots$ :
1) Compute $\beta_n = \|\tilde{v}_n\|$;
   If $\beta_n = 0$: set $L = n - 1$, $v_{L+1} = 0$, and stop;
2) Otherwise, set $v_n = \tilde{v}_n / \beta_n$;
3) Compute $\alpha_n = v_n^T T v_n$;
   Set $\tilde{v}_{n+1} = T v_n - \alpha_n v_n - \beta_n v_{n-1}$.
```

Notice that Algorithm 5.1 is just a special case of the classical Lanczos Algorithm 2.1 (applied to T and with starting vector $s_0 = \bar{r}_0$). However, unlike the general Algorithm 2.1, the Hermitian Lanczos process cannot break down prematurely. In exact arithmetic, Algorithm 5.1 stops after a finite number of steps with termination index

$$L = \dim K_N(r_0, A) = \dim K_N(r_0, T). \tag{5.6}$$

Moreover, with V_n defined as in (2.1) and

$$T_n = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & \beta_n & \alpha_n \end{bmatrix}, \tag{5.7}$$

for the Hermitian Lanczos method, the properties (2.5)-(2.7) listed in Proposition 2.2 now reduce as follows:

$$V_n^H V_n = I_n, \tag{5.8}$$

$$TV_n = V_nT_n + [\,0 \quad 0 \quad \cdots \quad 0 \quad \tilde{v}_{n+1}\,], \tag{5.9}$$

$$K_n(r_0, A) = K_n(r_0, T) = \operatorname{span}\{v_1, v_2, \dots, v_n\}. \tag{5.10}$$

Here and in the sequel, it is always assumed that $n \in \{1, 2, \dots, L\}$. Note that, with

$$H_n^{(e)} = \begin{bmatrix} T_n + i\sigma I_n \\ \beta_{n+1} e_n^T \end{bmatrix} \tag{5.11}$$

and by adding $i\sigma V_n$ to both sides of (5.9), we obtain

$$AV_n = V_{n+1} H_n^{(e)}. \tag{5.12}$$

Next, we rewrite the MR, ME, and GAL conditions in terms of V_n and $H_n^{(e)}$. In order to match the notations used in Chapter 3, we set $\rho_0 = \beta_1$, and thus

$$r_0 = \rho_0 v_1, \quad \rho_0 = \beta_1 = \|r_0\|. \tag{5.13}$$

Proposition 5.2.
a) $x_k^{MR} = x_0 + V_k z_k^{MR}$ *where* z_k^{MR} *is the solution of the least squares problem*

$$\min_{z \in \mathbf{C}^n} \left\| \rho_0 e_1^{(n+1)} - H_n^{(e)} z \right\|. \tag{5.14}$$

b) $x_n^{ME} = x_0 + A^H V_n z_n^{ME}$ *where* z_n^{ME} *is the solution of*

$$\rho_0 e_1^{(n)} = (H_n^{(e)})^H H_n^{(e)} z. \tag{5.15}$$

c) $x_n^{GAL} = x_0 + V_n z_n^{GAL}$ *where* z_n^{GAL} *is the solution of*

$$\rho_0 e_1^{(n)} = (T_n + i\sigma I_n) z. \tag{5.16}$$

Moreover, if $\sigma = 0$ *and* T_n *is singular, then no Galerkin iterate satisfying (5.2) exists.*
d) $x_L^{MR} = x_L^{ME} = x_L^{GAL} = x_\star$.

Proof. First, note that, by (5.13) and (5.8),

$$V_j^H r_0 = \rho_0 e_1^{(j)}, \quad j = 1, 2, \dots, L+1. \tag{5.17}$$

Using (5.12) and (5.8), we obtain

$$V_n^H A^H A V_n = (H_n^{(e)})^H H_n^{(e)}. \tag{5.18}$$

a) From (5.13) and (5.12), r_n can be represented as in (3.3). With (3.3) and (5.8), it follows that the MR property (1.3) is equivalent to (5.14).

b) (5.4), (5.10), and (5.5) imply that

$$x_n = x_0 + A^H V_n z_n, \quad r_n = r_0 - A^H A\ V_n z_n, \quad \text{with} \quad z_n \in \mathbf{C}^n.$$

The minimization property (5.3) is equivalent to

$$0 = v^H A(x_* - x_n) = v^H r_n \quad \text{for all} \quad v \in K_n(r_0, A).$$

By (5.10), it suffices to consider these equations for $v = v_j$, $j = 1, \ldots, n$, and it follows that z_n is the solution of

$$V_n^H r_0 = V_n^H A^H A\ V_n z$$

which, by (5.17) (for $j = n$) and (5.18), is just the linear system (5.15).

For c), we similarly obtain that $x_n = x_0 + V_n z_n$ satisfies (5.2) iff z_n solves the linear system

$$\rho_0 e_1 = V_n^H V_{n+1} H_n^{(e)} z$$

whose coefficient matrix, by (5.8) and (5.11), is $T_n + i\sigma I_n$. If $\sigma = 0$ and T_n is singular, the linear system (5.16) could have a solution only if it was consistent. Using the fact that T_{n-1} is nonsingular then and $\beta_n > 0$, one easily verifies that this cannot be the case.

d) In view of (5.6), $K_L = K_L(r_0, A)$ is an A-invariant subspace and, since $r_0 \in K_L$, we conclude that

$$x_* - x_0 = A^{-1} r_0 \in A^{-1} K_L = K_L = A^H K_L.$$

On the other hand, x_* trivially satisfies (1.3), (5.2), and (5.3), and it follows that $x_L = x_*$ for all three methods. □

5.2. Practical implementations

First, consider the MR approach. By comparing (5.14) with (3.6), we conclude that, for shifted Hermitian matrices (5.1), the MR and the QMR methods are identical, provided one sets $\Omega_n = I_{n+1}$ in (3.6) and the QMR Algorithm 3.1 is based on the Hermitian Lanczos Algorithm 5.1. Therefore, an actual MR algorithm for matrices (5.1) can be obtained as a special case of the implementation of the general QMR method described in Sections 3.1 and 3.2. Here, we present a slight modification of the resulting implementation which will help to reduce complex arithmetic.

Since the Lanczos matrix T_n in (5.7) is real symmetric, it follows that

$$(H_n^{(e)})^H H_n^{(e)} = T_n^2 + \sigma^2 I_n + \beta_{n+1}^2 e_n e_n^T$$

is a real matrix. Consequently, one can choose the unitary matrix Q_n in a QR decomposition

$$H_n^{(e)} = Q_n^H \begin{bmatrix} R_n \\ 0 \end{bmatrix} \tag{5.19}$$

of the complex matrix (5.11) such that the upper triangular factor R_n is real. Using standard matrix calculus, one verifies that a factorization (5.19) with real R_n can be constructed with a unitary matrix Q_n of the form

$$Q_n = G_n D_n \begin{bmatrix} G_{n-1} & 0 \\ 0 & 1 \end{bmatrix} D_{n-1} \cdots D_3 \begin{bmatrix} G_2 & 0 \\ 0 & I_{n-2} \end{bmatrix} D_2 \begin{bmatrix} G_1 & 0 \\ 0 & I_{n-1} \end{bmatrix} D_1 \tag{5.20}$$

with complex diagonal matrices

$$D_j = \operatorname{diag}(1, \ldots, 1, \underset{\overset{\uparrow}{j}}{e^{i\varphi_j}}, 1, \ldots, 1), \quad \varphi_j \in \mathbf{R},$$

and real Givens rotations

$$G_j = \begin{bmatrix} I_{j-1} & 0 & 0 \\ 0 & c_j & s_j \\ 0 & -s_j & c_j \end{bmatrix}, \quad \text{with} \quad c_j \in \mathbf{C},\ s_j \in \mathbf{C},\ c_j^2 + s_j^2 = 1.$$

Recall that for the QR factorization in Section 3.2 we have used slightly different unitary matrices Q_n (cf. (3.12) and (3.13)). Also, note that, in contrast to the Lanczos matrix generated by the look-ahead Lanczos process, $H_n^{(e)}$ is now a scalar tridiagonal matrix. Hence, the upper triangular R_n in (5.19) is of the form

$$R_n = \begin{bmatrix} \delta_1 & \epsilon_2 & \theta_3 & 0 & \cdots & 0 \\ 0 & \delta_2 & \epsilon_3 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \delta_3 & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \theta_n \\ \vdots & & & \ddots & \ddots & \epsilon_n \\ 0 & \cdots & \cdots & \cdots & 0 & \delta_n \end{bmatrix} \tag{5.21}$$

with scalar entries δ_k, ϵ_k, θ_k (cf. (3.14)). Moreover, the factorization is easily updated from the one of the previous step $n-1$ by simply setting

$$\begin{aligned} &\theta_n = s_{n-2}\beta_n, \quad \epsilon_n = s_{n-1}\alpha_n + c_{n-1}c_{n-2}\beta_n \cos\varphi_{n-1}, \\ &h_n = -s_{n-1}c_{n-2}\beta_n e^{-i\varphi_{n-1}} + c_{n-1}(\alpha_n - i\sigma), \quad \tilde{\delta}_n = |h_n|, \\ &e^{i\varphi_n} = \begin{cases} h_n/|h_n| & \text{if} \quad h_n \neq 0, \\ 0 & \text{if} \quad h_n = 0, \end{cases} \end{aligned} \tag{5.22}$$

and

$$\delta_n = \sqrt{\tilde{\delta}_n^2 + \beta_{n+1}^2}, \quad c_n = \tilde{\delta}_n/\delta_n, \quad s_n = \beta_{n+1}/\delta_n. \tag{5.23}$$

Based on the QR decomposition (5.19), one then proceeds as in the derivation of the implementation of the QMR method in Section 3.2. We omit the details and only state the resulting algorithm.

Algorithm 5.3. (MR method for matrices (5.1).)

```
0) Choose $x_0 \in \mathbf{C}^N$ and set $v = b - Ax_0$, $v_0 = p_0 = p_{-1} = 0$,
   $\beta_1 = \tilde{\tau}_1 = ||v||$, $c_0 = c_{-1} = 1$, $s_0 = s_{-1} = \varphi_0 = 0$;
For $n = 1, 2, \ldots$ :
1) If $\beta_n = 0$, stop: $x_{n-1}$ solves $Ax = b$;
   Otherwise, compute
2) $v_n = v/\beta_n$, $\alpha_n = v_n^H T v_n$,
   $v = Tv_n - \alpha_n v_n - \beta_n v_{n-1}$, $\beta_{n+1} = ||v||$,
   and then $\theta_n$, $\epsilon_n$, $\delta_n$, $\varphi_n$, $c_n$, $s_n$ using formulas (5.22), (5.23);
3) $p_n = (v_n - \epsilon_n p_{n-1} - \theta_n p_{n-2})/\delta_n$,
   $x_n = x_{n-1} + \tau_n p_n$ with $\tau_n = c_n \tilde{\tau}_n e^{i\varphi_n}$,
   $\tilde{\tau}_{n+1} = -s_n \tilde{\tau}_n e^{i\varphi_n}$.
```

We now turn to the ME and GAL methods. First, note that the characterization (5.2) of the GAL iterates is just a special case of the Galerkin type condition (3.33) (with $s_0 = \overline{r_0}$). Hence, as a special case of the results in Section 3.3 on the connection between QMR and BCG, we can obtain a stable implementation of the GAL approach based on the MR Algorithm 5.3. Instead, we now derive an implementation of the ME approach and show how the GAL iterates can be recovered from the ME method.

With (5.19) and by setting

$$Y_n = [y_1 \quad y_2 \quad \cdots \quad y_n]) := A^H V_n R_n^{-1},$$

it follows from part b) of Proposition 5.2 that

$$x_n^{ME} = x_0 + Y_n u_n \quad \text{where } u_n \text{ is the solution of } \beta_1 e_1 = R_n^T u.$$

Similarly, using that, by (5.11),

$$T_n + i\sigma I_n = \left(H_n^{(e)}\right)_n^T \begin{bmatrix} I_n \\ 0 \end{bmatrix}$$

and with (5.19) and (5.20), one deduces from (5.16) that x_n^{GAL} exists if, and only if, $c_n \neq 0$ and then

$$x_n^{GAL} = x_0 + \tilde{Y}_n \tilde{u}_n, \quad \tilde{Y}_n := V_n Q_{n-1}^T \operatorname{diag}(1, 1, \ldots, 1, e^{i\varphi_n}),$$

where $\tilde{u}_n$ is the solution of

$$\beta_1 e_1 = R_n^T \operatorname{diag}(1, \ldots, 1, c_n)\, \tilde{u}.$$

Clearly, u_n and $\tilde{u}_n$ differ only in their last elements η_n and $\tilde{\eta}_n$. Moreover, with (5.12), (5.19), and (5.20), one easily verifies that $\tilde{Y}_n$ is identical to Y_n up to its last column $\tilde{y}_n$. Hence, we obtain the recursions

$$x_n^{ME} = x_{n-1}^{ME} + \eta_n y_n \quad \text{and, if } c_n \neq 0, \quad x_n^{GAL} = x_{n-1}^{ME} + \tilde{\eta}_n \tilde{y}_n \tag{5.24}$$

(cf. [PS]). The resulting implementations can be summarized as follows.

Algorithm 5.4. (ME/GAL method.)

```
0) Choose $x_0 \in \mathbf{C}^N$ and set $x_0^{ME} = x_0^{GAL} = x_0$, $v = b - Ax_0$, $v_0 = \tilde{y}_0 = 0$.
   $\beta_1 = ||v||$, $\eta_0 = -1$. $c_0 = c_{-1} = 1$, $s_0 = s_{-1} = \varphi_0 = \eta_{-1} = 0$;
   If $\beta_1 > 0$, set $v_1 = v/\beta_1$;
For $n = 1, 2, \dots$ :
1) If $\beta_n = 0$, stop: $x_{n-1}^{ME} = x_{n-1}^{GAL} = x_*$ solves $Ax = b$;
   Otherwise, compute
2) $\alpha_n = v_n^H T v_n$,
   $v = Tv_n - \alpha_n v_n - \beta_n v_{n-1}$, $\beta_{n+1} = ||v||$,
   and then $\theta_n$, $\epsilon_n$, $\tilde{\delta}_n$, $\delta_n$, $\varphi_n$, $c_n$, $s_n$ using formulas (5.22), (5.23);
3) $\tilde{y}_n = e^{i\varphi_n}\ (-s_{n-1}\tilde{y}_{n-1} + c_{n-1}v_n)$
   and, if $\tilde{\delta}_n \neq 0$ and the Galerkin iterate is desired,
   $x_n^{GAL} = x_{n-1}^{ME} + \tilde{\eta}_n \tilde{y}_n$ with $\tilde{\eta}_n = -(\epsilon_n \eta_{n-1} + \theta_n \eta_{n-2})/\tilde{\delta}_n$;
4) Set $v_{n+1} = v/\beta_{n+1}$, if $\beta_{n+1} > 0$, and $v_{n+1} = 0$ otherwise,
   $y_n = c_n \tilde{y}_n + s_n v_{n+1}$,
   $x_n^{ME} = x_{n-1}^{ME} + \eta_n y_n$ with $\eta_n = -(\epsilon_n \eta_{n-1} + \theta_n \eta_{n-2})/\delta_n$.
```

The finite termination property of the Lanczos algorithm does no longer hold in the presence of roundoff error (see, *e.g.*, [GVL, pp. 332]), and the stopping criterion stated in Algorithms 5.3 and 5.4 is not useful in practice. Instead, one should terminate the iteration as soon as $||r_n||$ is sufficiently reduced. Note that, similar to the real symmetric case [PS], $||r_n||$ can be obtained without computing the vector r_n itself by using the following identities:

$$||r_n^{ME}|| = \sqrt{\eta_{n+1}^2 \delta_{n+1}^2 + \eta_n^2 \theta_{n+2}^2},$$

$$||r_n^{MR}|| = ||r_0|| s_1 s_2 \cdots s_n,$$

$$||r_n^{GAL}|| = \beta_{n+1} |s_{n-1}\eta_{n-1} + c_{n-1}\tilde{\eta}_n e^{i\varphi_n}|.$$

Finally, consider linear systems $Ax = b$ with coefficient matrices A of the more general class

$$A = T + i\sigma D \quad \text{where} \quad T = T^H \quad \text{is Hermitian,} \quad \sigma \in \mathbf{R},$$

with D a Hermitian positive definite $N \times N$ matrix. Then, $Ax = b$ is equivalent to the linear system

$$A'x' = b' \quad \text{where} \quad A' = D^{-1/2} A D^{-1/2}, \quad x' = D^{1/2} x, \quad b' = D^{-1/2} b,$$

whose coefficient matrix A' is now of the form (5.1), so that we can use Algorithm 5.3 or 5.4 for its solution. Note that one never needs to form A' and b' explicitly, and it is straightforward to rewrite both Algorithms 5.3 and 5.4 in terms of the original linear system $Ax = b$. We omit the details and only state that the resulting MR, ME, and GAL algorithms generate iterates which are characterized by the properties (1.3) (with $\| \; \| = \| \; \|_{D^{-1}}$ and $K_n(D^{-1}r_0, D^{-1}A)$), (5.3) (with $\| \; \| = \| \; \|_D$ and $K_n(D^{-1}A^H D^{-1} r_0, D^{-1}A)$), and (5.2) (with $K_n(D^{-1}r_0, D^{-1}A)$), respectively.

5.3. Comparisons with other implementations. Operation counts

Several authors [JY, Sid, AMS] have proposed algorithms for the computation of the MR and GAL iterates (1.3) and (5.2), respectively. However, most of these implementations (like Orthomin and Orthores in [JY]) are modeled after variants of the conjugate residual or conjugate gradient algorithm for Hermitian positive definite matrices. It is well known [PS, Cha, SF] that, for Hermitian indefinite A, these approaches are numerically unstable and can even break down. For instance, for the GAL method this occurs whenever a Galerkin iterate does not exist (cf. [PS] and part c) of Proposition 5.2.). The same stability problems can arise for the non-Hermitian matrices (5.1) if σ is small. Hence, all these algorithms derived directly from the positive definite case are stable only for matrices (5.1) which fulfill additional requirements such as T positive definite or $|\sigma|$ bounded away from 0. Note that these two conditions are not satisfied for most of the applications mentioned in Section 1.3.

Here, we consider only implementations which are numerically stable for the general class of matrices (5.1). Among the proposed algorithms in the literature merely the Orthodir approach [JZ, AMS] for the computation of the MR iterates has this property. This algorithm can be stated as follows.

Algorithm 5.5. (Orthodir MR implementation.)

```
0) Choose $x_0 \in \mathbf{C}^N$ and set $s_0 = r_0 = b - Ax_0$,
   $q_0 = As_0$, $s_{-1} = q_{-1} = 0$, $\nu_0 = 0$;
For $n = 0, 1, \ldots$ :
1) If $q_n = 0$, stop: $x_n$ solves $Ax = b$;
   Otherwise, compute
2) $\lambda_n = q_n^H r_n / \|q_n\|^2$,
   $x_{n+1} = x_n + \lambda_n s_n$, $r_{n+1} = r_n - \lambda_n q_n$;
3) $\mu_n = q_n^H T q_n / \|q_n\|^2$ and, if $n > 0$, $\nu_n = \|q_n\|^2 / \|q_{n-1}\|^2$,
   $s_{n+1} = q_n - (\mu_n + i\sigma) s_n - \nu_n s_{n-1}$,
   $q_{n+1} = Tq_n - \mu_n q_n - \nu_n q_{n-1}$.
```

We remark that $q_n = As_n$ and that the search directions s_n are up to scalar factors identical to the vectors p_n in Algorithm 5.3.

Next, the results of operation counts for Algorithms 5.3, 5.4, and 5.5 are presented in Table 5.1. Although we solve complex linear systems, most of the scalars (like α_n and

β_n in the Lanczos step of Algorithm 5.3 and 5.4) occurring in the computations are real. Moreover, on some machines, implementations in real arithmetic are more advantageous. Therefore, we compare work and storage in terms of real quantities. Listed are the number of matrix-vector products $T \cdot v$, $v \in \mathbf{R}^N$, the approximate number m of additional real multiplications per iteration, and the number s of real vectors (of length N) to be stored. The computation of inner products often constitutes a bottleneck on modern computers. For this reason, we also give the number dp of dot products $x \cdot y$, $x, y \in \mathbf{R}^N$ per iteration. Finally, notice that — based on the simple observation stated in Proposition 5.6 below — work and storage for the MR and ME/GAL methods can be significantly reduced if the Hermitian part T of the matrix (5.1) is real. This case occurs frequently in the cited applications, and we included the corresponding operation counts in Table 5.1.

Proposition 5.6. *Let T be real and assume that $r_0 = b - Ax_0 \in \mathbf{R}^N$. Then, all the vectors v_n, $n = 1, 2, \ldots$, in Algorithm 5.3 and 5.4 are real. In addition, for the MR method, all search directions p_n are real vectors.*

Note that often the right-hand side b is a real vector, and then the standard starting guess $x_0 = 0$ guarantees that r_0 is real. In the general case $b \in \mathbf{C}^N$ and if $\sigma \neq 0$, the condition $r_0 \in \mathbf{R}^N$ can always be fulfilled by choosing the starting vector $x_0 = x_0^{(1)} + ix_0^{(2)}$ appropriately, *e.g.*, $x_0^{(2)} = 0$ and $x_0^{(1)} = \operatorname{Im} b/\sigma$. However, such a strategy might not be desirable, if one already knows a good approximation x_0 for the exact solution of $Ax = b$.

	T· v	m	dp	s
MR Algorithm 5.3	2	18N	4	12
ME/GAL Algorithm 5.4	2	18N	4	10
Orthodir Algorithm 5.5	2	26N	8	14
If T and r_0 are real:				
MR Algorithm 5.3	1	9N	2	7
ME/GAL Algorithm 5.4	1	13N	2	7
If $A = T$ and r_0 are real:				
MINRES [PS]	1	8N	2	6
SYMMLQ [PS]	1	8N	2	5

Table 5.1. Work per iteration and storage for the various algorithms. Listed are the number of matrix-vector products $T \cdot v$, $v \in \mathbf{R}^N$, the approximate number m of additional real multiplications, the number of real dot products dp, and the number s of real vectors to be stored.

To explain the numbers given in Table 5.1, a few more comments are necessary. For the ME/GAL algorithm, we have assumed that the Galerkin iterate is, if desired, only computed in the very last step of the iteration. Furthermore, in order to reduce the computational work, note that, in the MR Algorithm 5.3, one computes the vector $\delta_n p_n$ instead of p_n. Similarly, in part 4) of Algorithm 5.4, the vector y_n itself is never needed and, hence, $\eta_n y_n$ is generated directly. Moreover, using fast Givens rotations (*e.g.* [GVL, p.158]), we compute the rescaled vector $f_n \tilde{y}_n$ instead of $\tilde{y}_n$ in step 3) of Algorithm 5.4. Here, $f_n := 1/(c_{n-1} \cos\varphi_n)$ for the case that $s_{n-1} \leq c_{n-1}$ and $|\sin\varphi_n| \leq |\cos\varphi_n|$, and f_n is defined correspondingly for the remaining cases. Note that then only $4n$ real multiplications are needed for updating $f_n \tilde{y}_n$ from $f_{n-1}\tilde{y}_{n-1}$ and v_n.

We conclude this section with a few further remarks. First, Table 5.1 clearly shows that the MR implementation stated in Algorithm 5.3 is less expensive than the Orthodir Algorithm 5.5. For real symmetric linear systems, Algorithm 5.3 and 5.4 reduce to MINRES and SYMMLQ [PS], respectively. Notice that, for the case of complex matrices (5.1) with T and r_0 real, Algorithm 5.3 and 5.4 require only little extra work and storage compared to MINRES and SYMMLQ. Finally, consider real linear systems with matrices

$$A = I - S \quad \text{where} \quad S = -S^T \quad \text{is real and skewsymmetric,} \tag{5.25}$$

(or, equivalently, $A' = iA = T + iI$ with $T = -iS = T^H$ if rewritten in the form (5.1)). Concus, Golub [CG], and Widlund [Wid] were the first to propose a Galerkin type method for the class of matrices. It can be shown, that their algorithm is equivalent to the Galerkin part of Algorithm 5.4 for the special case (5.25). Also, note that, in [Fre1, Sto], we have investigated an Orthodir type implementation of the ME approach for the class $A = I - S$. The first MR-type algorithm for the family of matrices (5.25) was proposed by Rapoport [Rap] (see also [EES, Fre1] for different implementations).

5.4. Error bounds

In this section, we derive error bounds for the MR and ME methods. Let $\alpha \leq \lambda_{\min}(T)$ and $\beta \geq \lambda_{\max}(T)$ be given bounds for the extreme eigenvalues of T. Therefore, all eigenvalues of A are contained in the complex line segment $S := [\alpha + i\sigma, \beta + i\sigma]$. For the rest of this chapter, we assume that in the Hermitian case $\sigma = 0$, $A = T$ is positive definite and $0 < \alpha < \beta$. This guarantees that $0 \notin S$.

By the standard technique, using

$$K_n(r_0, A) = \{\Psi(A) r_0 \mid \Psi \in \Pi_{n-1}\} \tag{5.26}$$

and an expansion of r_0 into orthonormal eigenvectors of A (recall that, by (5.5), A is normal!), one obtains from (1.3) the estimate

$$\|r_n^{MR}\|/\|r_0\| \leq \min_{\Psi \in \Pi_{n-1}} \max_{\lambda \in S} |1 - \lambda\Psi(\lambda)|. \tag{5.27}$$

Similarly, with (5.4), (5.26), we deduce from (5.3) that

$$||x_* - x_n^{ME}||/||x_* - x_0|| \leq \min_{\Psi \in \Pi_{n-1}} \max_{\lambda \in S} |1 - |\lambda|^2 \Psi(\lambda)|. \tag{5.28}$$

With the linear transformation

$$z = z(\lambda) = \frac{2(i\sigma - \lambda) + \beta + \alpha}{\beta - \alpha}, \tag{5.29}$$

which maps S onto the unit interval $[-1,1]$, the right-hand side of (5.27) can be rewritten in the form

$$(E_n(a) :=) \min_{\Phi \in \Pi_n : \Phi(a) = 1} \max_{z \in [-1,1]} |\Phi(z)| \tag{5.30}$$

where

$$a := \frac{2i\sigma + \beta + \alpha}{\beta - \alpha} \notin [-1,1]. \tag{5.31}$$

Furthermore, using the identity

$$4\,|\lambda|^2 = (\beta - \alpha)^2 (z(\lambda) - a)(z(\lambda) - \bar{a}), \quad \lambda \in S,$$

(note that $z(\lambda) = \overline{z(\lambda)}$ for all $z \in S$) one easily verifies that the upper bound in (5.28) is just $E_{n+1}^{(r)}(a)$ where

$$\begin{gathered}(E_n^{(r)}(a) :=) \min_{\Phi \in \Pi_n(a)} \max_{z \in [-1,1]} |\Phi(z)|, \\ \Pi_n(a) := \{\Phi \in \Pi_n \mid \Phi(a) = \Phi(\bar{a}) = 1 \quad \text{and, if} \quad a \in \mathbf{R},\ \Phi'(a) = 0\}.\end{gathered} \tag{5.32}$$

We now turn our attention to the two approximation problems (5.30) and (5.32). It will be convenient to represent a in the form

$$\begin{gathered} a = a(\psi) = a(R)\cos\psi + ib(R)\sin\psi, \quad R > 1, \quad 0 \leq \psi < 2\pi, \\ a(R) := \frac{1}{2}(R + \frac{1}{R}), \quad b(R) := \frac{1}{2}(R - \frac{1}{R}); \end{gathered} \tag{5.33}$$

clearly, this is possible for any $a \notin [-1,1]$. For fixed $R > 1$, we set $\mathcal{B}_R = \{a = a(\psi) | 0 \leq \psi < 2\pi\}$ and remark that $\mathcal{B}_R = \partial\mathcal{E}_R$ just describes the boundary of the ellipse $\mathcal{E}_R$ (defined as in (3.59), with r replaced by R) with foci at ±1 and semi-axes $a(R)$, $b(R)$.

First, we consider the complex approximation problem (5.30). Its solution is classical for the case of real a where $T_n(z)/T_n(a)$ is the optimal polynomial. Here, T_n denotes the nth Chebyshev polynomial which, by means of the Joukowsky map, is given by

$$T_n(z) \equiv \frac{1}{2}(v^n + \frac{1}{v^n}), \quad z \equiv \frac{1}{2}(v + \frac{1}{v}). \tag{5.34}$$

For purely imaginary a, the extremal polynomials were found by Freund and Ruscheweyh [FR], but for general complex a the solution of (5.30) is not explicitly known. The following upper bound for the optimal value of (5.30) will be used in Section 5.5.

Theorem 5.7. *Let $R > 1$ and $n = 1, 2, \ldots$. Then,*

$$\frac{1}{R^n} < E_n(a) \leq \frac{2}{R^n + 1/R^n}, \quad a \in \mathcal{B}_R. \tag{5.35}$$

Proof. The lower bound follows immediately from an inequality due to S.N. Bernstein (*e.g.* [Mei, Theorem 74]). The upper bound is just the special case $r = 1$ of Theorem 3.6. □

We remark that, for fixed $R > 1$, the upper bound in (5.35) is optimal, with equality holding for the two real points of $\mathcal{B}_R$. The optimal lower bound is unknown, but it is conjectured to be $2/(R^n + R^{n-2})$ which is just the optimal value of (5.30) for the two purely imaginary points of $\mathcal{B}_R$ (cf. [FR]).

Next, we study the approximation problem (5.32), and we will show that it is closely related to the classical Zolotarev problem

$$\min_{\Psi \in \Pi_{n-2}} \max_{z \in [-1,1]} |z^n + \eta n z^{n-1} - \Psi(z)|, \quad \eta \in \mathbf{R}, \quad n = 2, 3, \ldots . \tag{5.36}$$

It is well known that there always exists a unique best approximation $\Psi_n(z; \eta)$ for (5.36) and the corresponding polynomials

$$Z_n(z; \eta) \equiv z^n + \eta n z^{n-1} - \Psi_n(z; \eta), \quad \eta \in \mathbf{R}, \quad n = 2, 3, \ldots,$$

are called Zolotarev polynomials. We refer the reader to [CT] for a detailed study of these polynomials. Note that

$$Z_n(z; \eta) \equiv 2^{1-n}(1 + |\eta|)^n \, T_n(\frac{z + \eta}{1 + |\eta|}) \quad \text{for} \quad |\eta| \leq \tan^2 \frac{\pi}{2n}, \tag{5.37}$$

and for the remaining values of η there are representations of $Z_n(z; \eta)$ in terms of elliptic functions.

Theorem 5.8. *Let $a = a(\psi) \in \mathcal{B}_R$, $R > 1$, $n = 2, 3, \ldots$. Then, there exists a unique optimal polynomial $\Phi_n(z; a)$ for (5.32). If $\psi = j\pi/(n-1)$ with an integer $j \not\equiv 0 \bmod n-1$, then*

$$\Phi_n(z; a) \equiv \frac{T_{n-1}(z)}{T_{n-1}(a)} \quad \text{and} \quad E_n^{(r)}(a) = \frac{2}{R^{n-1} + 1/R^{n-1}}.$$

Otherwise,

$$\Phi_n(z; a) \equiv \frac{Z_n(z; \eta)}{Z_n(a; \eta)} \tag{5.38}$$

where $\eta = \eta(a)$ is the unique solution of

$$\operatorname{Im} Z_n(a; \eta) = 0 \quad (\textit{respectively } Z_n'(a; \eta) = 0, \textit{ if } a \in \mathbf{R}), \quad \eta \in \mathbf{R}. \tag{5.39}$$

In particular, if ψ satisfies for some integer $j \not\equiv 0 \bmod n$

$$\cos\psi = \cos\frac{j\pi}{n} - \frac{\eta\,\sin^2\frac{j\pi}{n}}{a(R) + \operatorname{sign}\eta\,\cos\frac{j\pi}{n}} \quad \textit{with} \quad |\eta| \le \tan^2\frac{\pi}{2n}, \quad \eta \in \mathbf{R}, \tag{5.40}$$

then

$$\Phi_n(z;a) \equiv \frac{T_n(\frac{z+\eta}{1+|\eta|})}{T_n(\frac{a+\eta}{1+|\eta|})} \quad \textit{and} \quad E_n^{(r)}(a) = \frac{2}{\rho^n + 1/\rho^n} \tag{5.41}$$

with ρ defined by

$$\frac{1}{2}(\rho + \frac{1}{\rho}) = a(R) - \frac{|\eta|}{1+|\eta|}\;\frac{b(R)^2}{a(R) + \operatorname{sign}\eta\,\cos\frac{j\pi}{n}}, \quad \rho > 1. \tag{5.42}$$

Proof. Writing $\Phi \in \Pi_n(a)$ in the form $\Phi(z) \equiv 1 - (z-a)(z-\bar{a})\Psi(z)$, $\Psi \in \Pi_{n-2}$, one recognizes (5.32) as a linear Chebyshev approximation problem, for which, since $a \notin [-1,1]$, Haar's condition is satisfied. Standard results from approximation theory (see, *e.g.*, [Mei]) guarantee that there always exists a unique optimal polynomial $\Phi_n(z;a)$ for (5.32). Moreover, because of the symmetry of the problem with respect to the real axis, Φ_n is a real polynomial, and Φ_n is characterized by assuming its maximum absolute value at at least n points in $[-1,1]$ with alternating signs. This alternation property implies that Φ_n has degree $n-1$ or n. First, consider the case $n-1$. Since the scalar multiples of T_{n-1} are the only polynomials of degree $n-1$ with an alternating set of length at least n, we conclude that $\Phi_n(z;a) \equiv T_{n-1}(z)/T_{n-1}(a)$, and, in view of $\Phi_n \in \Pi_n(a)$, this case occurs iff $T_{n-1}(a) \in \mathbf{R}$ and $a \notin \mathbf{R}$. With (5.33) and (5.34), one readily verifies that these are just the points $a = a(\psi)$ with $\psi = j\pi/(n-1)$, $j \not\equiv 0 \bmod n-1$. Now we turn to the case that Φ_n is of degree n. Since the optimal polynomials for the Zolotarev problem (5.36) are characterized by the same alternating property as Φ_n, it follows that Φ_n is of the form (5.38) with a suitable $\eta \in \mathbf{R}$. In order to guarantee $\Phi_n \in \Pi_n(a)$, η must be the solution of (5.39).

Now, let $\eta \in \mathbf{R}$, $|\eta| \le \tan^2\frac{\pi}{2n}$. With (5.37) and (5.34), we conclude that a satisfies (5.39) iff

$$(\tilde{a} :=)\ \frac{a+\eta}{1+|\eta|} = \frac{1}{2}(\rho + \frac{1}{\rho})\cos\frac{j\pi}{n} + \frac{i}{2}(\rho - \frac{1}{\rho})\sin\frac{j\pi}{n} \tag{5.43}$$

for some $\rho > 1$ and some integer $j \not\equiv 0 \bmod n$. By using the representation (5.33) of a and by equating the real (imaginary) parts of (5.43), one arrives at two real nonlinear equations for the unknowns $\cos\psi$ and ρ, and a straightforward, but lengthy calculation shows that the solutions are given by (5.40) and (5.42). Finally, note that the first identity in (5.41) is a consequence of (5.38) and (5.37); the second one follows from $E_n^{(r)}(a) = 1/|T_n(\tilde{a})|$ and (5.43). □

For general a, (5.38) and (5.39) lead to rather complicated and not very useful formulas for $E_n^{(r)}(a)$ in terms of elliptic integrals. Next, we derive simple bounds for this quantity.

Theorem 5.9. *Let $R > 1$ and $n = 2, 3, \dots$. Then, for $a = a(\psi) \in \mathcal{B}_R$:*

$$\frac{2}{R^n + 1/R^n} \;\le\; E_n^{(r)}(a) \;\le\; 2\frac{b_{n-1}(R)|f_{n-1}(\psi)| + b_n(R)|f_n(\psi)|}{b_{2n-1}(R) + b_1(R) f_{2n-1}(\psi)} \quad \left(=: B_n^{(r)}(a)\right) \qquad (5.44)$$

where

$$b_j(R) = \frac{1}{2}(R^j - \frac{1}{R^j}), \quad f_j(\psi) = \begin{cases} \sin(j\psi)/\sin\psi & \text{if } \sin\psi \neq 0, \\ (-1)^{(j-1)l} j & \text{if } \psi = l\pi. \end{cases}$$

Both bounds in (5.44) are attained if $\psi = j\pi/n$, $j \not\equiv 0 \bmod n$. In addition, the upper estimate is sharp for $\psi = j\pi/(n-1)$, $j \not\equiv 0 \bmod n-1$.

Proof. Duffin and Schaeffer [DS] showed that for any real polynomial of degree at most n, $|\Phi(z)| \le M$ on $[-1,1]$ implies $|\Phi(a)| \le M(R^n + 1/R^n)/2$ for all $a \in \mathcal{B}_R$. Application of this result to $\Phi_n(z; a)$ yields the lower bound in (5.44). In order to obtain the upper bound, we consider polynomials $\Phi(z) \equiv \gamma T_n(z) + \delta T_{n-1}(z) \in \Pi_n(a)$ with $\gamma, \delta \in \mathbf{R}$. With (5.33) and (5.34), one readily verifies that $\Phi \in \Pi_n(a)$ iff γ and δ satisfy

$$\begin{bmatrix} (R^n + 1/R^n)\cos(n\psi) & (R^{n-1} + 1/R^{n-1})\cos(n-1)\psi \\ (R^n - 1/R^n) f_n(\psi) & (R^{n-1} - 1/R^{n-1}) f_{n-1}(\psi) \end{bmatrix} \begin{bmatrix} \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

A routine calculation shows that this linear system has a unique solution and that

$$\max_{z \in [-1,1]} |\Phi(z)| \;=\; |\gamma| + |\delta| \;=\; B_n^{(r)}(a).$$

Finally, the statements on the sharpness of (5.44) follow from Theorem 5.8. □

Note that the bounds in (5.44) are asymptotically optimal, and we have the following

Corollary 5.10. *Let $R > 1$ and $a \in \mathcal{B}_R$. Then,*

$$\lim_{n\to\infty} (E_n^{(r)}(a))^{1/n} \;=\; \lim_{n\to\infty} (B_n^{(r)}(a))^{1/n} \;=\; \frac{1}{R}.$$

The typical behavior of the optimal values of (5.30) and (5.32) and the bounds stated in Theorems 5.7 and 5.9 is illustrated in Figure 5.1. For fixed $R = 1.103\dots$ and $n = 30$, the four curves

$$E_n(a) \le \frac{2}{R^n + 1/R^n} \le E_n^{(r)}(a) \le B_n^{(r)}(a), \quad a = a(\psi) \in \mathcal{B}_R, \quad 0 \le \psi \le \pi/2,$$

are plotted. Note that $E_n(a) = E_n(\bar{a}) = E_n(-a)$ (and analogously for $E_n^{(r)}(a)$), and hence it suffices to consider only the points a in the first quadrant.

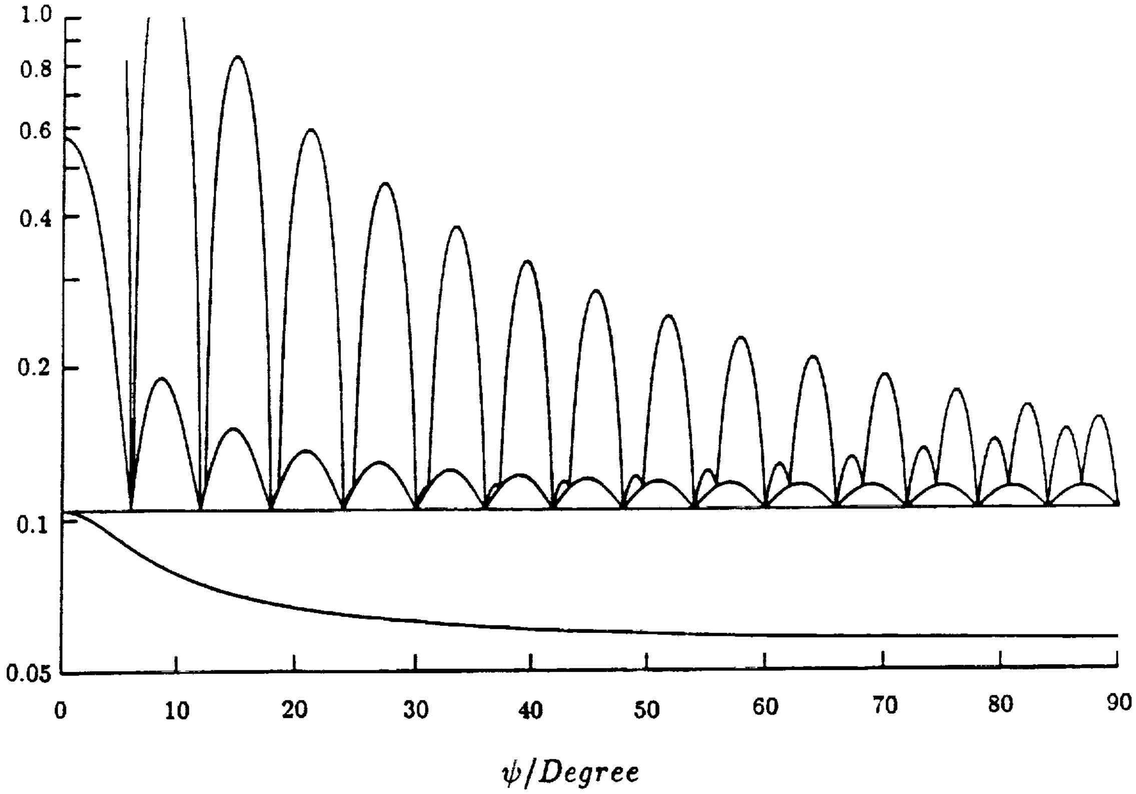

Figure 5.1. The optimal values $E_k(a)$ and $E_k^{(r)}(a)$ of the approximation problems (5.30) and (5.32) are shown for the case $k = 30$ and with $a = a(\psi)$ moving along the quarter of the ellipse $\mathcal{B}_R$, $R = 1.103\dots$. The lowest curve is $E_{30}(a)$. The other three curves display $E_{30}^{(r)}(a)$ and its lower and upper bounds as stated in Theorem 5.9.

The following theorem summarizes our results on error bounds for the MR and ME methods. For the special case of matrices $A = T + i\sigma I$ with positive definite Hermitian part T, we also derive an error bound for the GAL method.

Theorem 5.11. *Let $\alpha \leq \lambda_{\min}(T)$ and $\beta \geq \lambda_{\max}(T)$ be given, and assume that $0 < \alpha < \beta$ if $\sigma = 0$. Let a be given by (5.31), and let R be the unique solution of*

$$\frac{1}{2}(R + \frac{1}{R}) = \frac{\sqrt{\beta^2 + \sigma^2} + \sqrt{\alpha^2 + \sigma^2}}{\beta - \alpha}, \quad R > 1. \tag{5.45}$$

Then, for $n = 1, \dots$:

a)

$$\frac{||b - Ax_n^{MR}||}{||b - Ax_0||} \leq E_n(a) \leq \frac{2}{R^n + 1/R^n}. \tag{5.46}$$

b)

$$\frac{||x_* - x_n^{ME}||}{||x_* - x_0||} \leq E_{n+1}^{(r)}(a) \leq B_{n+1}^{(r)}(a). \tag{5.47}$$

c) If T is positive definite, then

$$\frac{\|x_* - x_n^{GAL}\|_T}{\|x_* - x_0\|_T} \leq \sqrt{1 + \frac{\sigma^2(\sqrt{\kappa} - \frac{1}{\sqrt{\kappa}})^2}{4\sigma^2 + \alpha^2(\sqrt{\kappa} + \frac{1}{\sqrt{\kappa}})^2}} \, \frac{2}{R^n + 1/R^n} \quad \text{where} \quad \kappa = \frac{\beta}{\alpha}. \tag{5.48}$$

Proof of part c). We set $e_n = x_* - x_n$ and $\mu_j = e_n^H T^j e_n$, $j = -1, 0.1$. With (5.1) and since $r_n = Ae_n$, one obtains

$$e_n^H r_n = \mu_1 + i\sigma\mu_0 \quad \text{and} \quad \|r_n\|_{T^{-1}}^2 = \mu_1 + \sigma^2\mu_{-1}. \tag{5.49}$$

Now let $u \in x_0 + K_n(r_0, A)$ be arbitrary. By (5.2), $(u - x_n)^H r_n = 0$, and therefore

$$e_n^H r_n = (x_* - u)^H r_n = \left(T^{1/2}(x_* - u)\right)^H (T^{-1/2} r_n). \tag{5.50}$$

By application of the Cauchy-Schwartz inequality to (5.50) and with (5.49), we arrive at

$$\mu_1^2 + \sigma^2\mu_0^2 \leq \|x_* - u\|_T^2 \, (\mu_1 + \sigma^2\mu_{-1}). \tag{5.51}$$

Next, recall that, by the Kantorovich inequality (*e.g.* [Hou, p. 83]),

$$s^2\mu_1\mu_{-1} \leq \mu_0^2 \quad \text{where} \quad s := \left(\frac{1}{2}(\sqrt{\kappa} + \frac{1}{\sqrt{\kappa}})\right)^{-1}. \tag{5.52}$$

Using (5.52) and the estimate $\mu_1/\mu_{-1} \geq \lambda_{\min}(T^2) = \alpha^2$, we obtain from (5.51)

$$\mu_1 \leq \|x_* - u\|_T^2 \, \frac{1 + \sigma^2\mu_{-1}/\mu_1}{1 + \sigma^2 s^2 \mu_{-1}/\mu_1} \leq \|x_* - u\|_T^2 \, \frac{\alpha^2 + \sigma^2}{\alpha^2 + \sigma^2 s^2}. \tag{5.53}$$

Since $u \in x_0 + K_n(r_0, A)$ is arbitrary, $\|x_* - u\|_T$ in (5.53) can be replaced by

$$\min_{u \in x_0 + K_n(r_0, A)} \|x_* - u\|_T = \min_{\Phi \in \Pi_n : \Phi(0) = 1} \|\Phi(A) e_0\|_T. \tag{5.54}$$

By expanding e_0 into orthonormal eigenvectors of the normal matrix A and with (5.29), (5.30), (5.31), and (5.35), we obtain

$$\min_{\Phi \in \Pi_n : \Phi(0) = 1} \|\Phi(A) e_0\|_T \leq \|e_0\|_T \, E_n(a) \leq \|e_0\|_T \, \frac{2}{R^n + 1/R^n}. \tag{5.55}$$

Finally, combining (5.53)-(5.55) yields the desired bound (5.48). □

We remark that, for the special case of $\sigma = 0$, (5.48) and (5.46) reduce to the usual error bounds (see, *e.g.*, [Sto]) for the classical conjugate gradient and conjugate residual algorithms.

In Theorem 5.11, we excluded the case of Hermitian indefinite matrices $A = T$. Error bounds for this case can be found in Chandra [Cha] for the MR method and in [SF, Frel, Szy] for the ME method.

Finally, we note that for the GAL method there are no satisfactory error bounds for the general class of matrices (5.1).

5.5. Polynomial preconditioning

Polynomial preconditioning aims at speeding up the convergence of conjugate gradient type methods for the solution of $Ax = b$ by applying them to one of the two equivalent linear systems

$$\Upsilon(A)Ax = \Upsilon(A)b \tag{5.56}$$

(left preconditioning), or

$$\Upsilon(A)Ay = b, \quad x = \Upsilon(A)y \tag{5.57}$$

(right preconditioning). Here Υ is a suitably chosen polynomial of small degree. For the case of Hermitian positive definite A, Rutishauser [Rut] proposed polynomial preconditioning in the 50's as a remedy for roundoff in the classical CG algorithm. The revival [JMP] of Rutishauser's method and the general interest in polynomial preconditioning is mainly motivated by the attractive features of this technique for vector and parallel computers (see [Saa2] for a survey). It is interesting to note that Lanczos seems to have been the first to consider polynomial preconditioning. The idea already appeared in his 1953 paper [Lan3] which, alas, is never referenced.

In this section, we study polynomial preconditioning for the class of matrices (5.1) $A = T + i\sigma I$. Let $l \geq 2$ be any fixed integer. We seek a polynomial $\Upsilon \in \Pi_{l-1}$ with the following two properties:

(i) the coefficient matrix $\Upsilon(A)A$ of (5.56) and (5.57) is again a shifted Hermitian matrix of the form (5.1);

(ii) the convergence of conjugate gradient type methods, applied to the preconditioned systems (5.56) or (5.57), is speeded up optimally.

As in the previous section, let $\alpha, \beta \in \mathbf{R}$ be given such that

$$\alpha \leq \mu \leq \beta \quad \text{for all eigenvalues } \mu \text{ of } T, \tag{5.58}$$

and assume that $0 < \alpha < \beta$ if $\sigma = 0$. Our criteria for optimal convergence in (ii) will be based on (5.58) as the only available information on the spectrum A and on the error bounds stated in Theorem 5.11.

First, consider requirement (i). For any $\Upsilon \in \Pi_{l-1}$, we can represent $\Upsilon(A)A$ in the form

$$\Upsilon(A)A = (T + i\sigma I)\Upsilon(T + i\sigma I) = \Psi(T) + i\tau I, \tag{5.59}$$

with $\Psi \in \Pi_l$ and $\tau \in \mathbf{R}$. Note that Υ, Ψ, and τ are related by

$$(\mu + i\sigma)\Upsilon(\mu + i\sigma) \equiv \Psi(\mu) + i\tau \quad \text{and} \quad \tau := i\Psi(-i\sigma). \tag{5.60}$$

Since $\Psi(T)$ is Hermitian if, and only if, Ψ is a real polynomial, it follows from (5.59) that (i) is fulfilled if, and only if, $\Psi \in \Pi_l^{(r)}$ and $\tau \in \mathbf{R}$. Therefore, from now on, it is assumed that $\Upsilon \in \Pi_{l-1}$ satisfies (5.60) with $\Psi \in \Pi_l^{(r)}$ and $\tau \in \mathbf{R}$.

Next, we turn to the question of optimal choice of Ψ and τ. A first, very tempting strategy is to require $\tau = 0$ and to choose Ψ such that $\Upsilon(A)A = \Psi(T)$ is positive definite. The preconditioned system (5.56) can then be solved by the standard CG method. Clearly, $\Psi(T) \approx I$ should approximate the identity matrix as best as possible. Using (5.58) and (5.60), we conclude that such an optimal Ψ is given as the best approximation in

$$\min_{\Psi \in \Pi_l^{(r)}:\, \Psi(-i\sigma)=0} \; \max_{\mu \in [\alpha,\beta]} |1 - \Psi(\mu)|. \tag{5.61}$$

For positive definite matrices $A = T$, this approach just leads to Rutishauser's method [Rut]. For the non-Hermitian case $\sigma \neq 0$, (5.61) turns out to be equivalent to the approximation problem (5.32), and we have the following

Theorem 5.12. *Let $\sigma \neq 0$ and $l \geq 2$. Then, there exists a unique best approximation in (5.61) given by*

$$\Psi^*(\mu) = 1 - \Phi_l(\frac{\beta + \alpha - 2\mu}{\beta - \alpha}; a), \quad a = \frac{\beta + \alpha + 2i\sigma}{\beta - \alpha}, \tag{5.62}$$

where $\Phi_l(z; a)$ is the extremal polynomial of (5.32) (for $n = l$) with optimal value $E_l^{(r)}(a)$ (cf. Theorem 5.8). Moreover, the matrix $\Upsilon(A)A = \Psi(T)$ is positive definite with eigenvalues in $[1 - E_l^{(r)}(a), 1 + E_l^{(r)}(a)]$, and for the iterates x_n of the CG method. applied to (5.56), the estimates

$$\frac{||x_* - x_n||_{\Psi(T)}}{||x_* - x_0||_{\Psi(T)}} \leq \frac{2}{\tilde{R}^n + 1/\tilde{R}^n}, \quad n = 1, 2, \dots, \quad \tilde{R} := \frac{1 + \sqrt{1 - (E_l^{(r)}(a))^2}}{E_l^{(r)}(a)}, \tag{5.63}$$

hold.

Proof. The linear transformation $z(\mu) \equiv (\beta + \alpha - 2\mu)/(\beta - \alpha)$ maps $[\alpha, \beta]$ onto $[-1, 1]$. Moreover, $\Phi(z(\mu)) \equiv 1 - \Psi(\mu)$ defines a one-to-one correspondence between all $\Psi \in \Pi_l^{(r)}$ with $\Psi(-i\sigma) = 0$ and all real polynomials $\Phi \in \Pi_l(a)$. This shows that (5.61) and (5.32) are equivalent (recall that the optimal polynomial for (5.32) is real), and, hence, Ψ^* is indeed the unique best approximation in (5.61). The error bounds (5.63) follow from (5.48) and (5.45) (with $\sigma = 0$, $\alpha = 1 - E_l^{(r)}(a)$, and $\beta = 1 + E_l^{(r)}(a)$). □

Recall (see Figure 5.1) that for fixed l of moderate size and fixed R, $E_l^{(r)}(a)$ strongly depends on the position of a on the ellipse $\mathcal{B}_R$. In particular, if a is close to the real points of the ellipse, $E_l^{(r)}(a)$ is significantly larger than for the other points of $\mathcal{B}_R$. Therefore, (5.63) suggests that the polynomial (5.62) will yield a poor preconditioner for matrices A which are nearly Hermitian positive definite. This will be confirmed by numerical results presented in Section 7.3. Therefore, in order to obtain a polynomial preconditioner which is satisfactory for all $a \in \mathcal{B}_r$, it is crucial to treat τ in (5.59) as a free parameter, and, next, we determine optimal choices of p and τ for speeding up the MR and ME algorithms.

First, consider the MR method. For it, right preconditioning (5.57) is the more natural choice between (5.56) and (5.57), since residual vectors for (5.57) are also residual vectors of the original linear system. Let y_n denote the nth iterate of the MR algorithm applied to $\Upsilon(A)Ay = b$, and set $x_n^{PP} = \Upsilon(A)y_n$. Moreover, let x_n be the nth approximation generated by the MR method applied to the original system $Ax = b$. Then, assuming that $x_0 = x_0^{PP}$, it follows with (5.57) that $K_n(\Upsilon(A)r_0, \Upsilon(A)A) \subset K_{nl}(r_0, A)$ and $x_n^{PP}, x_{nl} \in x_0 + K_{nl}(r_0, A)$. Hence, the minimization property (5.3) implies that $||b - Ax_{nl}|| \leq ||b - Ax_n^{PP}||$. Therefore, in view of (5.46), we conclude that, based on (5.58) as the only information on the spectrum of A, the best possible choice of $\Upsilon \in \Pi_{l-1}$ is one which guarantees the estimates

$$\frac{||b - Ax_n^{PP}||}{||b - Ax_0||} \leq \frac{2}{R^{nl} + 1/R^{nl}}, \quad n = 1, 2, \ldots. \tag{5.64}$$

with R defined in (5.45). We call $\Upsilon \in \Pi_{l-1}$ an *optimal* polynomial preconditioner for the MR algorithm if it leads to the error bounds (5.64).

Similarly, for the ME method with left polynomial preconditioning (5.56), the error bounds (5.47) and Corollary 5.10 suggest that the best possible choice of $\Upsilon \in \Pi_{l-1}$ is one for which the iterates x_n^{PP} satisfy

$$\frac{||x_* - x_n^{PP}||}{||x_* - x_0||} \leq E_{n+1}^{(r)}(\tilde{a}), \quad n = 1, 2, \ldots, \tag{5.65}$$

for some $\tilde{a} \in \mathcal{B}_{R^l}$. A polynomial $\Upsilon \in \Pi_{l-1}$ is called an *optimal* preconditioner for the ME approach if it guarantees (5.65).

With this notion of optimality, we can now state the main result of this section as follows.

Theorem 5.13. *Let $l \geq 2$. Then,*

$$\Upsilon_{l-1}(\lambda) \equiv \frac{\Psi_l(\lambda - i\sigma) + i\tau}{\lambda} \tag{5.66}$$

where

$$\Psi_l(\mu) \equiv T_l\left(\frac{2\mu - \beta - \alpha}{\beta - \alpha}\right) - \operatorname{Re} T_l(-a) \quad \textit{and} \quad \tau = -\operatorname{Im} T_l(-a), \tag{5.67}$$

is an optimal polynomial preconditioner for the MR and ME methods. Here. T_l denotes the lth Chebyshev polynomial (cf. (5.34)) and a is given in (5.62).

Proof. First, note that. by (5.67), $\Psi_l(-i\sigma) = -i\tau$, and thus (5.66) defines indeed a polynomial $\Upsilon \in \Pi_{l-1}$. Next, consider the preconditioned matrix $\tilde{A} = \Upsilon(A)A$. With (5.58) and since T_l maps the interval $[-1,1]$ onto itself, it follows that the eigenvalues of the Hermitian part $\Psi_l(T)$ of $\tilde{A}$ are contained in $[\tilde{\alpha},\tilde{\beta}]$ where $\tilde{\alpha} := -1 - \operatorname{Re} T_l(-a)$ and $\tilde{\beta} := 1 - \operatorname{Re} T_l(-a)$. Now we apply Theorem 5.11 (with $\alpha = \tilde{\alpha}$, $\beta = \tilde{\beta}$, and $\sigma = \tau$) and note that, by (5.33) and (5.34),

$$\tilde{a} = \frac{\tilde{\beta} + \tilde{\alpha} + 2i\tau}{\tilde{\beta} - \tilde{\alpha}} = -T_l(-a) \in \mathcal{B}_{R^l}.$$

The error bounds (5.64) and (5.65) are then an immediate consequence of parts a) and b) of Theorem 5.11, respectively. Hence Υ_{l-1} is an optimal polynomial preconditioner, and the proof is complete. □

We remark that, in [ELV], Eiermann, Li, and Varga developed a general theory for polynomial preconditioning for asymptotically optimal semi-iterative methods. In particular, by means of Theorem 5.13 from [ELV], one can show that the polynomial preconditioner (5.66) is also best possible for semi-iterative procedures for the class of matrices (5.1).

Also, recall that, for the GAL approach, there are in general no error bounds on which we could base the choice of a best possible polynomial Υ. However, in analogy to the case of real symmetric matrices (see [SW, SF, Szy]), preconditioning for the GAL method can be motivated by its close connection (cf. (5.24)) to the ME algorithm. Therefore, we regard (5.66) also as an optimal polynomial preconditioner for the GAL method.

Finally, note that polynomial preconditioning is easily incorporated into the MR and ME/GAL Algorithms 5.3 and 5.4. Right preconditioning leads to slightly more economical implementations, and only this choice is considered in the sequel. The idea is to apply the CG type methods to the linear system $\Upsilon_{l-1}(A)Ay = b - Ax_0$ with starting guess $y_0 = 0$. The resulting iterates y_n of the MR and ME/GAL approaches are generated by Algorithm 5.3 and 5.4, respectively, modified in the following way: substitute y_n for x_n, replace, in (5.22), σ by τ (defined in (5.67)), and finally, in step 2) of Algorithm 5.3 and 5.4, perform the following Lanczos recursion

$$\begin{gathered} v = z^{(n)} - \tilde{\alpha}_n v_n - \beta_n v_{n-1}, \\ \text{where} \quad z^{(n)} := T_l\Big(\frac{2}{\beta - \alpha}T - \frac{\beta + \alpha}{\beta - \alpha}I\Big)v_n, \quad \tilde{\alpha}_n := v_n^H z^{(n)}, \end{gathered} \tag{5.68}$$

and set $\alpha_n = \tilde{\alpha}_n - \operatorname{Re} T_l(-a)$. We remark that for this computation only T_l, but never the complex polynomial (5.66), is used. The actual preconditioner Υ_{l-1} appears only in the translation of the y_n into the corresponding iterates

$$x_n = x_0 + \Upsilon_{l-1}(A)y_n \tag{5.69}$$

for the original system $Ax = b$. However, we do not need to generate x_n in each step. Note that the norm $||r_n||$ of the residual $r_n = b - Ax_n$ is available (cf. Section 5.2) from the procedure generating y_n, and the iteration is stopped as soon as $||r_n||$ is sufficiently reduced. Hence, x_n is computed only once, namely in the very last step of the algorithm.

Finally, notice that $z^{(n)}$ in (5.68) can be obtained by performing l steps of the classical Chebyshev semi-iterative method (see Golub and Varga [GV]). More precisely, setting

$$z_j^{(n)} := T_j(\frac{2}{\beta - \alpha}T - \frac{\beta + \alpha}{\beta - \alpha}I)v_n, \quad T' := T - \frac{\beta + \alpha}{2}I, \quad \omega := \frac{2}{\beta - \alpha}, \tag{5.70}$$

the three-term recurrence formula of the Chebyshev polynomials leads to the following

Algorithm 5.14. (Computation of $z^{(n)}$ in (5.68).)

```
0) Set $z_0^{(n)} = v_n$ and $z_1^{(n)} = \omega\, T'v_n$;
1) For $j = 2, \ldots, l$, compute
     $z_j^{(n)} = 2\omega T' z_{j-1}^{(n)} - z_{j-2}^{(n)}$;
2) Set $z^{(n)} = z_l^{(n)}$.
```

We remark that the computation of $z^{(n)}$ via Algorithm 5.14 requires $2l$ matrix-vector products $T \cdot v$, $v \in \mathbf{R}^N$, and $2l$ additional real multiplications. If T and r_0 are real (cf. Section 5.3), all $z_j^{(n)}$ are real too, and the work is halved.

Similarly, using (5.66), (5.67), and again the three-term recurrence formula of the Chebyshev polynomials, a routine calculation shows that the following algorithm just yields the iterate (5.69).

Algorithm 5.15. (Computation of x_n in (5.69).)

```
0) Set $h_o^{(n)} = y_n$ and $h_1^{(n)} = 2\omega(T'y_n - (\frac{\beta+\alpha}{2} + i\sigma)y_n)$;
1) For $j = 2, \ldots, l-1$, compute
     $h_j^{(n)} = 2\omega T' h_{j-1}^{(n)} - h_{j-2}^{(n)} + 2T_j(-a)y_n$;
2) Set $x_n = x_0 + \omega h_{l-1}^{(n)}$.
```

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 assumed to be a general complex $N \times N$ matrix. Recall that, in view of (1.10), the iterates of any Krylov subspace method (1.2) for solving (1.1) are of the form

$$x_n = x_0 + \Phi(A) r_0, \quad \Phi \in \Pi_{n-1}. \tag{6.1}$$

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

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

A second real version of (1.1) is

$$A_{**} \begin{bmatrix} \operatorname{Re} x \\ -\operatorname{Im} x \end{bmatrix} = \begin{bmatrix} \operatorname{Re} b \\ \operatorname{Im} b \end{bmatrix}, \quad A_{**} := \begin{bmatrix} \operatorname{Re} A & \operatorname{Im} A \\ \operatorname{Im} A & -\operatorname{Re} A \end{bmatrix}. \tag{6.3}$$

Obviously, (6.2) and (6.3) are the only essentially different possibilities of rewriting (1.1) as a real $2N \times 2N$ system. Furthermore, note that A_* is nonsymmetric if, and only if, $A \neq A^H$ is non-Hermitian, whereas A_{**} is symmetric if, and only if, $A = A^T$. Hence, for complex symmetric linear systems the approach (6.3) appears to be especially attractive since it permits the use of simple CG-type methods such as SYMMLQ and MINRES for real symmetric matrices.

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

Proposition 6.1.

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

$$\begin{bmatrix} J & 0 \\ 0 & \overline{J} \end{bmatrix} = X_*^{-1} A_* X_* \quad \textit{where} \quad X_* := \frac{1}{\sqrt{2}} \begin{bmatrix} X & -i\overline{X} \\ -iX & \overline{X} \end{bmatrix}. \tag{6.4}$$

In particular,

$$\lambda(A_*) = \lambda(A) \cup \overline{\lambda(A)}. \tag{6.5}$$

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

$$-\lambda, \overline{\lambda}, -\overline{\lambda} \in \lambda(A_{**}) \quad \textit{for all} \quad \lambda \in \lambda(A_{**}). \tag{6.6}$$

Moreover,

$$\lambda(A_{**}) = \{\lambda \in \mathbf{C} \mid \lambda^2 \in \lambda(\overline{A}A)\}.$$

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

$$A = U\Sigma U^T, \quad U \text{ unitary}, \quad \Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N) \geq 0. \tag{6.7}$$

Moreover, A_{**} is a real symmetric matrix with spectral decomposition

$$A_{**} = \begin{bmatrix} Y & -Z \\ Z & Y \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix} \begin{bmatrix} Y & -Z \\ Z & Y \end{bmatrix}^T \quad \text{where} \quad Y = \operatorname{Re} U, \; Z = \operatorname{Im} U. \tag{6.8}$$

Proof. a) First, note that

$$X_* = S \begin{bmatrix} X & 0 \\ 0 & \overline{X} \end{bmatrix} \quad \text{where} \quad S := \frac{1}{\sqrt{2}} \begin{bmatrix} I_N & -iI_N \\ -iI_N & I_N \end{bmatrix} \quad \text{is unitary.} \tag{6.9}$$

In particular, (6.9) shows that with X also X_* is nonsingular. One readily verifies that

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

and, in view of (6.9), this implies (6.4). (6.5) is an obvious consequence of (6.4).

b) Since

$$\begin{bmatrix} 0 & I_N \\ -I_N & 0 \end{bmatrix}^{-1} A_{**} \begin{bmatrix} 0 & I_N \\ -I_N & 0 \end{bmatrix} = -A_{**},$$

the real matrices A_{**} and $-A_{**}$ are similar. Hence, (6.6) holds true. The relation between $\lambda(A_{**})$ and $\lambda(\overline{A}A)$ is known (see [HJ, p. 214] for a proof).

c) (6.7) is the well-known Takagi singular value decomposition for symmetric matrices (*e.g.* [HJ, Corollary 4.4.4]). By rewriting (6.7) in terms of the real and imaginary parts of A and U, one obtains (6.8) (cf. [HJ, 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 [Eis, Fre1, Fre2] for examples), iterations based on Krylov subspaces generated by A offer no advantage over solving the normal equations (1.8) by standard CG. See Theorem 6.3 below for a theoretical result along these lines.

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

$$\lambda(A) \subset \{\lambda \in \mathbf{C} \mid \operatorname{Im} \lambda \geq 0\}. \tag{6.10}$$

However, by rewriting (1.1) as real linear systems (6.2) respectively (6.3), one deliberately creates coefficient matrices whose spectra are most unfavorable for Krylov subspace methods. The case (6.3) is especially bad since, in view of (6.6). $\lambda(A_{**})$ is symmetric with respect to real and imaginary axis and hence the eigenvalues always embrace the origin. Similarly, by (6.5), the coefficient matrix A_* of (6.2) in general has eigenvalues in the upper as well as in the lower half-plane. In particular, if (6.10) 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 6.1. Consider the subclass of 5.1 of complex symmetric matrices of the form

$$A = T + i\sigma I \quad \text{where} \quad T = T^T \quad \text{is real} \quad \text{and} \quad \sigma > 0. \tag{6.11}$$

Obviously,

$$\begin{aligned} \lambda(A) &= \{\lambda = \mu + i\sigma \mid \mu \in \sigma(T)\} \\ &\subset S := \left[\mu_m + i\sigma, \mu_M + i\sigma\right] \end{aligned} \tag{6.12}$$
,

where $\mu_m = \lambda_{\min}(T)$ and $\mu_M = \lambda_{\max}(T)$. 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 (6.5), (6.12) implies

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

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 (6.11) and part b) of Proposition 6.1, we obtain

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

Note that the class (6.11) is closely related to shifted skewsymmetric matrices. Indeed, if, instead of $Ax = b$, we rewrite $-iAx = -ib$ as a real system (6.2), one obtains

$$(-iA)_* = \begin{bmatrix} \sigma I_N & T \\ -T & \sigma I_N \end{bmatrix} = \sigma I_{2N} - S, \quad S := \begin{bmatrix} 0 & -T \\ T & 0 \end{bmatrix} \quad \left(= -S^T\right). \tag{6.13}$$

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

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

6.2. Correspondence of Krylov subspace methods

In analogy to (6.1) for complex linear systems (1.1), a Krylov subspace method for the solution of the equivalent real systems (6.2) respectively (6.3) generates iterates

$$\begin{bmatrix} \operatorname{Re} x_n \\ \operatorname{Im} x_n \end{bmatrix} = \begin{bmatrix} \operatorname{Re} x_0 \\ \operatorname{Im} x_0 \end{bmatrix} + \Phi(A_*) \begin{bmatrix} \operatorname{Re} r_0 \\ \operatorname{Im} r_0 \end{bmatrix}, \quad \Phi \in \Pi_{n-1}^{(r)}, \tag{6.14}$$

respectively

$$\begin{bmatrix} \operatorname{Re} x_n \\ -\operatorname{Im} x_n \end{bmatrix} = \begin{bmatrix} \operatorname{Re} x_0 \\ -\operatorname{Im} x_0 \end{bmatrix} + \Phi(A_{**}) \begin{bmatrix} \operatorname{Re} r_0 \\ \operatorname{Im} r_0 \end{bmatrix}, \quad \Phi \in \Pi_{n-1}^{(r)}. \tag{6.15}$$

In the sequel, the notation

$$K_n^{(r)}(c,B) := \{\Phi(B)c \mid \Phi \in \Pi_{n-1}^{(r)}\} \quad \bigl(\subset K_n(c,B)\bigr)$$

will be used.

At first glance, it might appear that Krylov subspace iterations (6.1) respectively (6.14–6.15) for the original complex systems respectively its equivalent real versions correspond to each other. However, as the following proposition shows this is not the case in general.

Proposition 6.2. *Let $n \in \mathbb{N}$.*
a) Let $\Phi \in \Pi_{n-1}$. Then, $x_n = x_0 + \Phi(A)r_0$ is equivalent to

$$\begin{bmatrix} \operatorname{Re} x_n \\ \operatorname{Im} x_n \end{bmatrix} = \begin{bmatrix} \operatorname{Re} x_0 \\ \operatorname{Im} x_0 \end{bmatrix} + \Phi_1(A_*) \begin{bmatrix} \operatorname{Re} r_0 \\ \operatorname{Im} r_0 \end{bmatrix} + \Phi_2(A_*) \begin{bmatrix} \operatorname{Im} r_0 \\ -\operatorname{Re} r_0 \end{bmatrix} \tag{6.16}$$

where $\Phi = \Phi_1 + i\Phi_2$, $\Phi_1, \Phi_2 \in \Pi_{n-1}^{(r)}$.
b) Let $\Phi \in \Pi_{n-1}^{(r)}$. Then, (6.15) is equivalent to

$$x_n = \operatorname{Re} x_n + i \operatorname{Im} x_n = x_0 + \Psi(\overline{A}A)\overline{r_0} + \Upsilon(\overline{A}A)\overline{A}r_0 \tag{6.17}$$

where $\Psi \in \Pi_{\lfloor (n-1)/2 \rfloor}^{(r)}$ and $\Upsilon \in \Pi_{\lfloor (n-2)/2 \rfloor}^{(r)}$ are defined by $\Phi(\lambda) \equiv \Psi(\lambda^2) + \lambda\Upsilon(\lambda^2)$.

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

$$(A_*)^j = \begin{bmatrix} \operatorname{Re} A^j & -\operatorname{Im} A^j \\ \operatorname{Im} A^j & \operatorname{Re} A^j \end{bmatrix} \quad \text{and} \quad (A_{**})^{2j} = \begin{bmatrix} \operatorname{Re}(\overline{A}A)^j & \operatorname{Im}(\overline{A}A)^j \\ -\operatorname{Im}(\overline{A}A)^j & \operatorname{Re}(\overline{A}A)^j \end{bmatrix}, \tag{6.18}$$

as is easily verified by induction on j.

a) Let γ_j and δ_j be the coefficients of the real polynomials Φ_1 and Φ_2, respectively. Then,

$$\begin{aligned} \operatorname{Re}\Phi(A) &= \sum_{j=0}^{n-1}(\gamma_j \operatorname{Re} A^j - \delta_j \operatorname{Im} A^j), \\ \operatorname{Im}\Phi(A) &= \sum_{j=0}^{n-1}(\gamma_j \operatorname{Im} A^j + \delta_j \operatorname{Re} A^j). \end{aligned} \tag{6.19}$$

By reformulating $x_n = x_0 + \Phi(A)r_0$, by means of (6.19) and the first relation in (6.18), in terms of real and imaginary parts, one immediately obtains (6.16).

b) A routine calculation, using the second identity in (6.18), shows that (6.15) can be rewritten as

$$\begin{bmatrix} \operatorname{Re} x_n \\ -\operatorname{Im} x_n \end{bmatrix} = \begin{bmatrix} \operatorname{Re} x_0 \\ -\operatorname{Im} x_0 \end{bmatrix} + \begin{bmatrix} \operatorname{Re}\{\Psi(\overline{A}A)\overline{r}_0 + \Upsilon(\overline{A}A)\overline{A}r_0\} \\ -\operatorname{Im}\{\Psi(\overline{A}A)\overline{r}_0 + \Upsilon(\overline{A}A)\overline{A}r_0\} \end{bmatrix}.$$

Hence (6.15) and (6.17) are equivalent. □

In view of part a) of Proposition 6.2, the corresponding real equivalent of complex Krylov schemes (6.1) are iterations of the type (6.16) and not the obvious real Krylov subspace methods (6.14). Clearly, the actual choice of the polynomials in (6.1) respectively (6.14–6.15) 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 (6.14), from the first, one gives up n of the $2n$ real parameters which are available for optimizing complex Krylov subspace methods (6.1). Consequently, it is always preferable to solve the complex system (1.1) rather than the real version (6.2) by Krylov subspace methods. Furthermore, numerical tests reveal that the convergence behavior of the two approaches can be drastically different (see Chapter 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

$$\lambda(A_{**}) = \{\pm\sigma_j \mid j = 1, \dots, N\}. \tag{6.20}$$

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

Since there are simple extensions, namely SYMMLQ and MINRES, (cf. Section 5.1) of classical CG to real symmetric indefinite matrices, it is especially tempting to solve (6.3) by one of these methods. Recall that SYMMLQ generates iterates defined by a Galerkin condition, whereas MINRES is based on a minimal residual MR property (cf. (1.3)). Here, we consider only the MR approach. Applied to (6.3) it generates a sequence of iterates z_n, $n = 1, 2, \ldots$, which are characterized by

$$\|b_{**} - A_{**} z_n\| = \min_{z \in z_0 + K_n^{(r)}(r_0^{**}, A_{**})} \|b_{**} - A_{**} z\|, \quad z_n \in z_0 + K_n^{(r)}(r_0^{**}, A_{**}). \qquad (6.21)$$

Here, we have set

$$b_{**} := \begin{bmatrix} \operatorname{Re} b \\ \operatorname{Im} b \end{bmatrix}, \quad z_n := \begin{bmatrix} \operatorname{Re} x_n \\ -\operatorname{Im} x_n \end{bmatrix} \text{ for } n = 0, 1, \ldots, \quad r_0^{**} := b_{**} - A_{**} z_0. \qquad (6.22)$$

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 (6.20), $\lambda(A_{**})$ is even symmetric to the origin, A_{**} exhibits this undesirable property. Indeed, numerical tests show that the convergence behavior of the MR method (6.21) is practically identical to that of the tabooed approach to (1.1) via solving the normal equations (1.8) by standard CG [HS]. In the sequel, we refer to this latter method as CGNR. Notice that the iterates x_n of CGNR are defined by the minimization property

$$\|b - Ax_l\| = \min_{x \in x_0 + K_l(A^H r_0, A^H A)} \|b - Ax\|, \quad x_l \in x_0 + K_l(A^H r_0, A^H A). \qquad (6.23)$$

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 (6.8), r_0^{**} can be expanded into eigenvectors of A_{**} as follows:

$$r_0^{**} = \begin{bmatrix} Y & -Z \\ Z & Y \end{bmatrix} c \quad \text{with} \quad c = \begin{bmatrix} c_1 \\ \vdots \\ c_{2n} \end{bmatrix} \in \mathbb{R}^{2n}. \qquad (6.24)$$

Theorem 6.3. *Let x_n^{MR} and x_l^{CGNR} denote the iterates generated by (6.21-6.22) and (6.23), respectively, both started with the same initial guess $x_0 \in \mathbb{C}^N$. Assume that c in the expansion (6.24) of r_0^{**} satisfies*

$$|c_j| = |c_{N+j}|, \quad j = 1, 2, \ldots, N. \qquad (6.25)$$

Then,

$$x_l^{CGNR} = x_{2l}^{MR} = x_{2l+1}^{MR}, \quad l = 0, 1, \ldots. \qquad (6.26)$$

Proof. First, note that, in view of (6.8) and (6.24), 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 (6.20) and (6.25). respectively, the MR method generates iterates with $z_n \in z_0 + K_{\lfloor n/2 \rfloor}^{(r)}(A_{**}r_0^{**}, A_{**}^2)$ (see. *e.g.*, [Fre2]). Consequently, the iterates defined by (6.21) satisfy

$$z_{2l} = z_{2l+1} \in z_0 + K_l^{(r)}(A_{**}r_0^{**}, A_{**}^2). \tag{6.27}$$

In particular, by (6.22), (6.27) shows that $x_{2l}^{MR} = x_{2l+1}^{MR}$.

It remains to prove the first relation in (6.26). To this end, we remark that

$$\|b_{**} - A_{**}z\| = \|b - Ax\| \quad \text{for all} \quad z = \begin{bmatrix} \operatorname{Re} x \\ -\operatorname{Im} x \end{bmatrix}, \ x \in \mathrm{C}^N. \tag{6.28}$$

Moreover, by using (6.22) and part b) of Proposition 6.2 (applied to polynomials $\Phi(\lambda) \equiv \lambda\Upsilon(\lambda^2)$), we deduce

$$z_0 + K_l^{(r)}\left(A_{**}r_0^{**}, A_{*star}^2\right) = \left\{ \begin{bmatrix} \operatorname{Re} x \\ -\operatorname{Im} x \end{bmatrix} \,\middle|\, x \in x_0 + K_l^{(r)}(A^H r_0, A^H A) \right\} \tag{6.29}$$

(notice that $\overline{A} = A^H$ in (6.17)!). In view of (6.27–6.29), (6.21) (for $n = 2l$) can be rewritten in the form

$$\|b - Ax_{2l}^{MR}\| = \min_{x \in x_0 + K_l^{(r)}(A^H r_0, A^H A)} \|b - Ax\|, \quad x_{2l}^{MR} \in x_0 + K_l^{(r)}(A^H r_0, A^H A). \tag{6.30}$$

Finally, remark 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 (6.23) with (6.30), we conclude that $x_l^{CGNR} = x_{2l}^{MR}$. □

Clearly, the special symmetry condition (6.25) will not be satisfied in general. Nevertheless, all our numerical experiments showed (see Examples 7.3 and 7.4) that (6.26) is still fulfilled approximately, *i.e.*,

$$x_l^{CGNR} \approx x_{2l}^{MR} \approx x_{2l+1}^{MR}, \quad l = 0, 1, \ldots\,. \tag{6.31}$$

7. Numerical experiments

We have performed extensive numerical tests with the QMR algorithm and all the other iterative schemes considered in this thesis. In this chapter, we present a few typical results of these experiments for complex symmetric and shifted Hermitian linear systems arising from the Helmholtz equation (1.5). Numerical experiments with the QMR method applied to real nonsymmetric matrices are reported in [FN1, FN2].

7.1. The test problems

Consider (1.5) 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 (1.5) 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

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

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 (1.5), *i.e.*, $\sigma_2 \equiv 0$, but now with a typical complex boundary condition such as

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

(which is discretized using forward differences) and Dirichlet boundary conditions on the other three sides of the boundary of G, one again arrives at (7.1) where

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

The test problems presented in this chapter are all linear systems $Ax = b$ with complex symmetric coefficient matrices of the type (7.1). Note that (7.1) is also a shifted Hermitian matrix if D is a multiple of the identity matrix.

For Examples 7.1 and 7.5, 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. Example 7.6 was run on a 128×128 grid leading to a 16384×16384 matrix A. 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 $1 + i$, and of Example 7.5, where the exact solution $x_\star$ was generated with random components in $[-1,1] + i[-1,1]$ and then the right-hand side was set to $b := Ax_\star$. As starting vector always $x_0 = 0$ was chosen.

As stopping criterion, we used

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

In Figures 7.1–4, the relative residual norm (7.3), R_n, is plotted versus the number N_n of matrix-vector products with A, A_*, or A_{**}. Note that $N_n = n$ is identical to the iteration number, except for CGS respectively CGNR which both require two matrix-vector products $A \cdot v$ respectively $A \cdot v$, $\overline{A} \cdot v$ per iteration and for which $N_n = 2n$. For GMRES [SS2], work and storage per iteration step n grows linearly with n and in practice it is necessary to use restarts. In the sequel, GMRES(n_0) and GMRES$_*$(n_0) refer to complex and real versions — restarted after every n_0 iterations — of the GMRES method applied to (1.1) and (6.2), respectively.

7.2. Complex symmetric linear systems

In a first series of experiments, QMR (with different weighting strategies) and BCG were compared. The natural choice (3.7) turned out to be the best strategy in all cases. In the following, QMR always refers to Algorithm 3.1 with weights (3.7). 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 an unweighted version of the QMR approach (based on the Lanczos vectors v_n, as generated by the complex symmetric Lanczos Algorithm 4.1) is displayed.

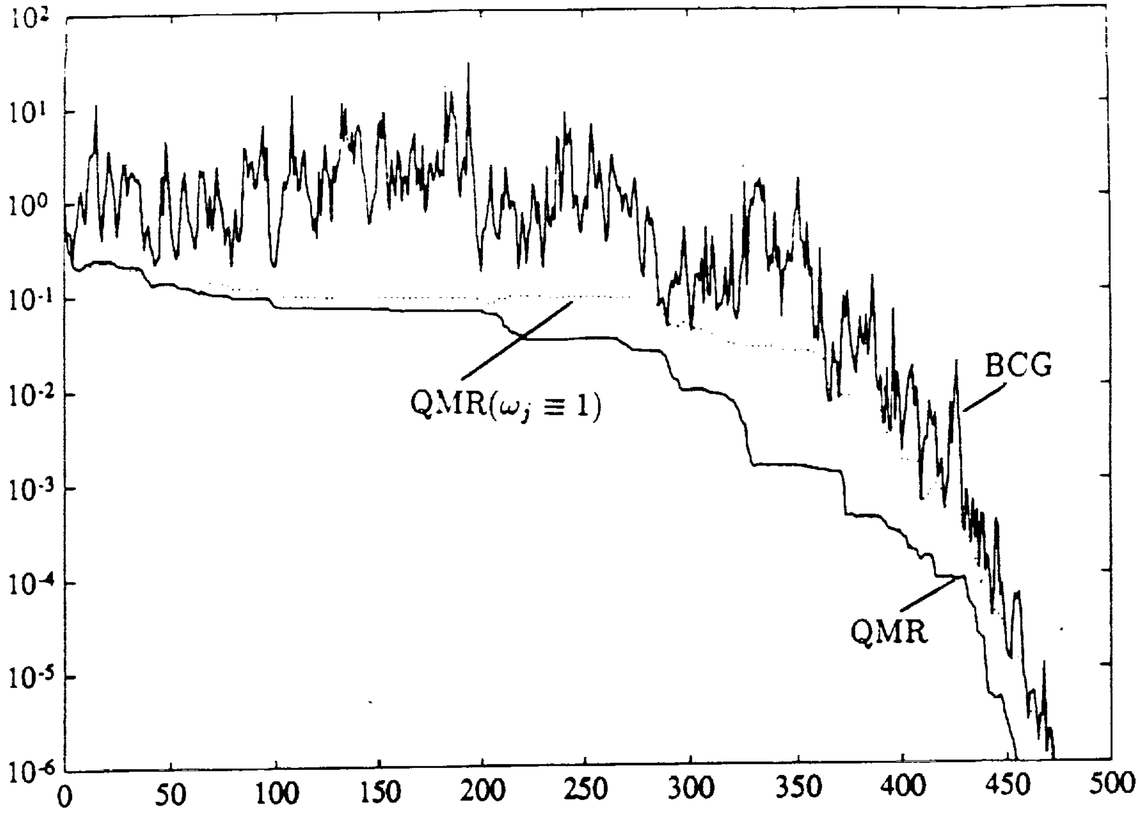

Figure 7.1. Convergence behavior of BCG, QMR, and an unweighted version of the QMR approach for Example 7.1.

Next, we compared the CGS Algorithm 4.8 and complex GMRES 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 and thus CGS is more expensive than QMR or BCG for complex symmetric matrices. Due to the necessary restarts, GMRES was never competitive with QMR, BCG, or CGS.

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 GMRES(20), QMR, BCG, and two runs of CGS with different starting vectors s_0, namely $s_0 = r_0$ respectively s_0 with random components in $[-1, 1] + i[-1, 1]$. Notice the extremely large residual norms in the early stage of the CGS iteration.

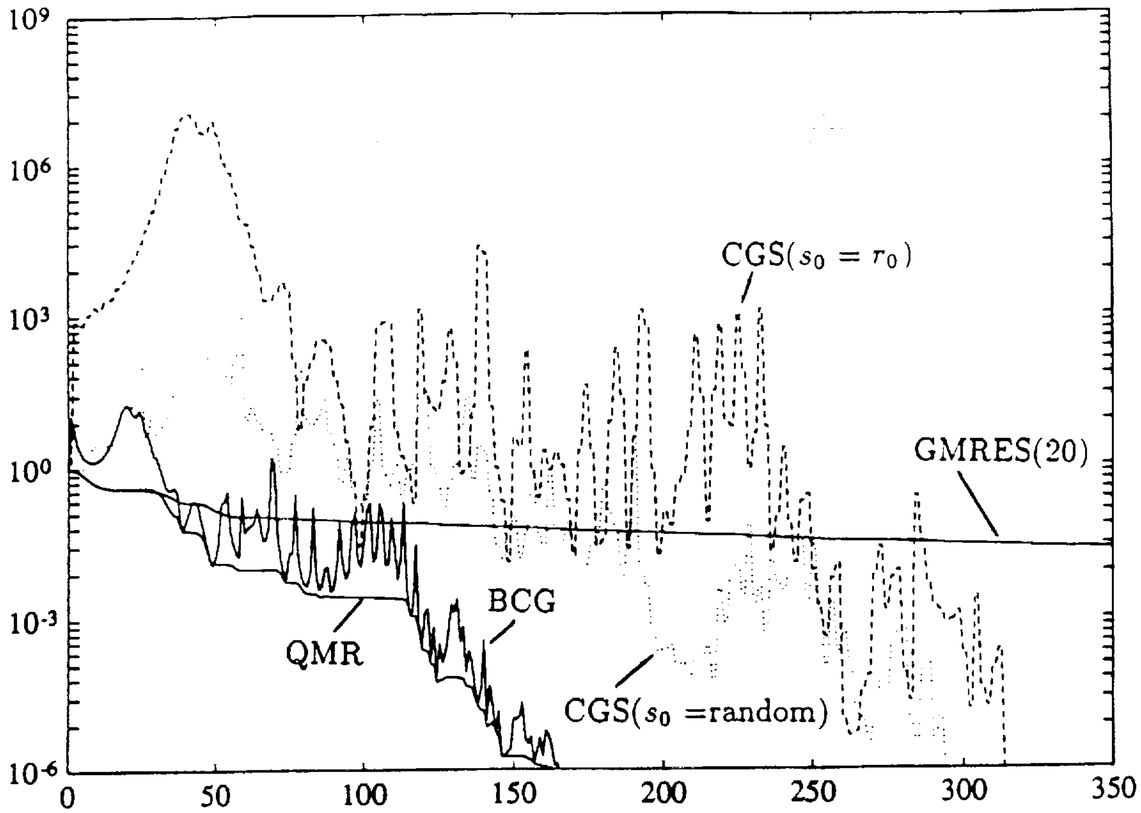

Figure 7.2. Convergence behavior of GMRES(20), QMR, BCG, and two runs of CGS with different starting vectors s_0 for Example 7.2.

In the following two examples, we compared CG-type methods for $Ax = b$ with real schemes for the equivalent real systems (6.2) respectively (6.3).

MR(A_{**}) denotes the MR method (6.21) applied to the real symmetric system (6.3).

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, the convergence behavior of QMR, MR(A_{**}), GMRES(20), GMRES(5), GMRES$_*$(5), and CGNR is shown. Notice that, although the symmetry condition (6.25) is not fulfilled, the curves for CGNR and MR(A_{**}) are almost identical. This confirms (6.31). Finally, we tried GMRES(k_0) and GMRES$_*$(k_0) also with other restart parameters k_0. For this example, both methods did never converge.

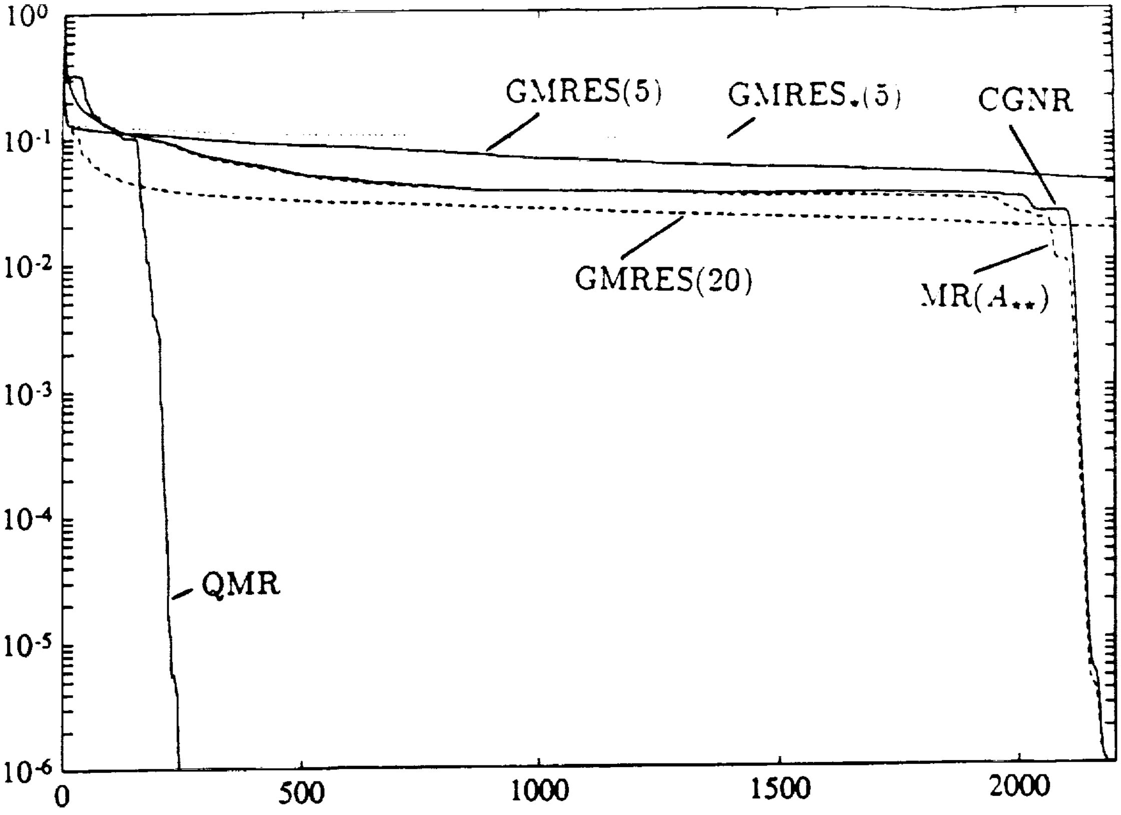

Figure 7.3. Convergence behavior of QMR, MR(A_{**}), GMRES(20), GMRES(5), GMRES$_*$(5), and CGNR for Example 7.3.

7.3. Shifted Hermitian linear systems

Now we choose $D = \sigma_2 I$ in (7.1). Then, (7.1) is a shifted Hermitian matrix of the form

$$A = T + i\sigma I, \quad T = A_0 - \sigma_1 h^2 I, \quad \sigma := \sigma_2 h^2. \tag{7.4}$$

Note that A is a shifted Hermitian matrix of the form (6.11) (cf. Example 6.1). In particular, A belongs to the class of matrices (5.1) and we can apply the algorithms developed in Chapter 5 to $Ax = b$.

Example 7.4. Let A be the 961 × 961 matrix (7.4) with $\sigma_1 = 1000$ and $\sigma_2 = 100$. Here, we denote by $MR(A)$ the run with MR Algorithm 5.3 applied to the original system $Ax = b$. Recall that, by rewriting $-iAx = -ib$ as a real system (6.2), one obtains a shifted skewsymmetric matrix (6.13), $(-iA)_*$. Again, for such matrices an efficient true minimal residual algorithm, denoted by MR($(-iA)_*$), exists [EES, Fre1]. Figure 7.4 shows the convergence behavior of MR(A), MR(A_{**}), MR($(-iA)_*$),CGNR, and GMRES(20). Notice that MR($(-iA)_*$) 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. $(-iA)_*$ in (6.13) is skewsymmetric, CGNR and MR($(-iA)_*$) are even equivalent [Fre1].

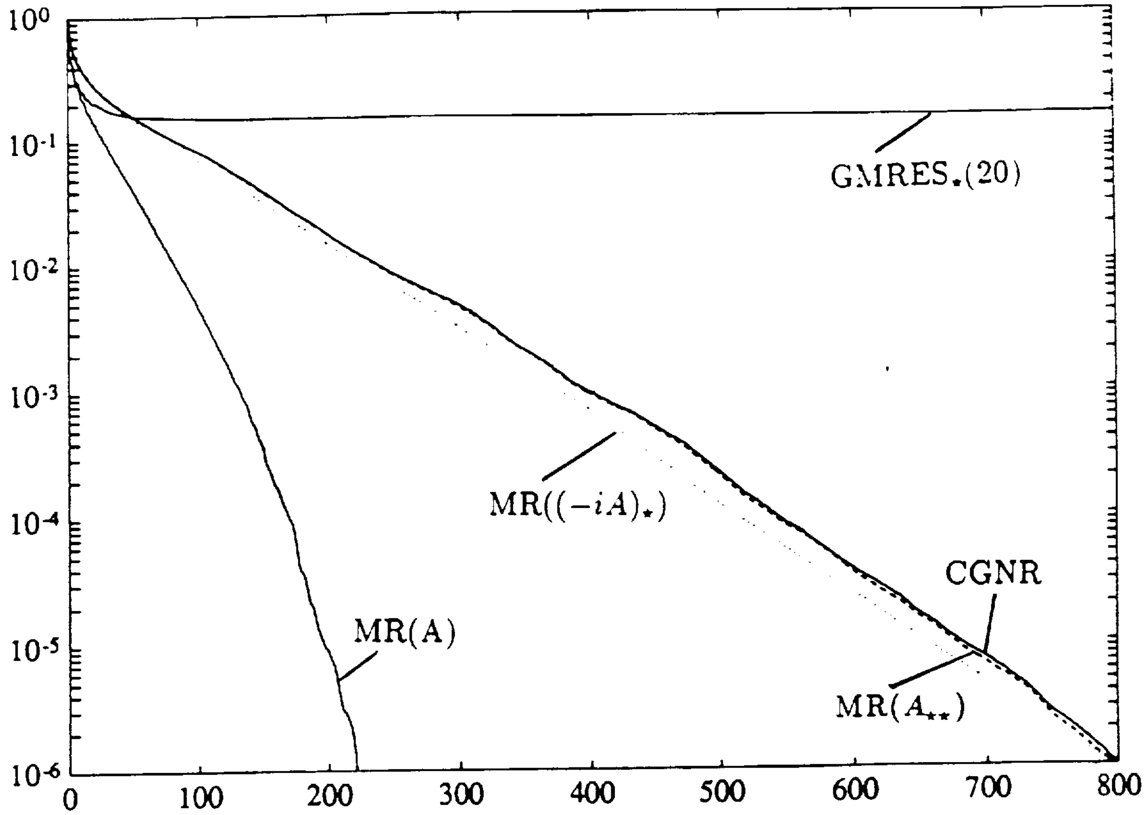

Figure 7.4. Convergence behavior of MR(A), MR($A_{\star\star}$), MR($(-iA)_\star$),CGNR, and GMRES(20) for Example 7.4.

In the next example, we tested the various polynomial preconditioners discussed in Section 5.5. Note that the eigenvalues of A_0 are known, and for our experiments with polynomial preconditioning we have used the true values

$$\alpha = \lambda_{\min}(A_0) - \sigma_1 h^2 \quad , \quad \beta = \lambda_{\max}(A_0) - \sigma_1 h^2 \qquad (7.5)$$

of the extreme eigenvalues of T (cf. (5.58)).

Examples 7.5. The matrix A is 3969×3969. For the constants in (7.4), values of the form $\sigma_1 = \sigma_1(\psi)$, $\sigma_2 = \sigma_2(\psi)$ were chosen. Here $0 \leq \psi \leq \pi/2$ is a parameter such that the points $a(\psi) = (\beta + \alpha + 2i\sigma)/(\beta - \alpha)$ all lie on the same ellipse $\mathcal{B}_R$, $R > 1$ fixed, with ψ describing the position of $a(\psi)$ on $\mathcal{B}_R$ (see (5.31) and (5.33)). The case $\psi = 0$ corresponds to a symmetric positive definite matrix (7.4), and for our experiments, we have chosen $R > 1$ such that $A = A_0$ for $\psi = 0$. Moreover, notice that with increasing ψ, the symmetric part T of (7.4) becomes more and more indefinite and $\alpha = -\beta$ for $\psi = \pi/2$. Also, the shift σ increases with ψ. Finally, we remark that the error bounds of Theorem 5.11 suggest that the MR and ME methods should display similar convergence rates for all ψ. In Tables 7.1–4, for several values of ψ (stated in degree!) and the various CG-type methods, we list the number of iterations which were necessary to reach (7.3). A "$\star$" indicates that the process still had not converged after 200 steps. In Table 7.1 the results for the MR, ME, and GAL Algorithms 5.3 respectively 5.4 (without preconditioning) are given. The Tables 7.2, 7.3, and 7.4 display the behavior of the three methods combined with the polynomial preconditioner (5.66) with $l = 6, 11$. and 16, respectively. Also listed are the results for the ZPCG method consisting of the classical CG algorithm with Zolotarev polynomial preconditioner (5.62) (see Theorem 5.12).

ψ/*Degree*	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
MR	120	126	148	165	175	183	190	197	203	208	212	217	221	224	228	232	234	237	239
ME	183	177	166	186	191	210	210	215	224	231	236	237	244	245	250	252	259	260	263
GAL	129	144	165	182	198	208	213	222	225	231	236	240	244	248	253	255	259	261	264

Table 7.1. Number of iterations after which the various algorithms had reduced the norm of the starting residual by 10^{-6}. Listed are the numbers for the basic methods without preconditioning. The family (depending on the parameter ψ) of test problems is the one described in Example 7.5.

ψ/*Degree*	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
PPMR	47	47	47	47	47	47	48	47	47	47	47	47	47	47	47	47	47	47	47
PPME	63	47	47	47	47	47	64	47	47	47	47	47	63	47	47	47	47	47	63
PPGAL	49	49	49	49	50	50	50	50	50	49	49	49	49	49	49	49	50	50	50
ZPCG	⋆	⋆	148	99	74	59	49	56	62	63	59	53	48	53	57	58	56	52	49

Table 7.2. Same as Table 7.1, but with polynomial preconditioning of degree $l = 6$.

ψ/*Degree*	0	5	10	15	20	25	30	35	40	45
PPMR	26	26	26	26	26	26	26	26	26	26
PPME	33	26	27	29	26	26	28	27	26	27
PPGAL	28	28	28	28	28	28	28	28	28	28
ZPCG	$\star$	87	44	29	32	34	29	29	31	29

ψ/*Degree*	50	55	60	65	70	75	80	85	90
PPMR	26	26	26	26	26	26	26	26	26
PPME	30	27	26	30	27	26	26	28	27
PPGAL	28	28	28	28	28	28	28	28	28
ZPCG	27	30	29	27	29	29	27	28	29

Table 7.3. Same as Table 7.1, but with polynomial preconditioning of degree $l = 11$.

ψ/*Degree*	0	5	10	15	20	25	30	35	40	45
PPMR	18	18	18	18	18	18	18	18	18	18
PPME	23	19	18	18	17	17	18	18	17	23
PPGAL	20	20	19	19	20	20	19	19	19	20
ZPCG	146	41	21	23	21	20	21	19	20	19

ψ/*Degree*	50	55	60	65	70	75	80	85	90
PPMR	18	18	18	18	18	18	18	18	18
PPME	17	18	18	17	17	17	17	17	23
PPGAL	19	19	19	19	19	19	19	19	20
ZPCG	20	19	20	19	19	20	19	20	19

Table 7.4. Same as Table 7.1, but with polynomial preconditioning of degree $l = 16$.

From these results, we draw the following conclusions. If used without preconditioning, the MR method appears to be superior to the ME and GAL approaches. However, note that the stopping criterion (7.3) is based on the norm of the residual, and this is more favorable for the MR method. A comparison based on the Euclidean norm of the error vector $x_\star - x_n$ displays a similar convergence behavior for the ME and MR approaches. In combination with polynomial preconditioning, the performance of all three methods PPMR, PPME, and PPGAL is nearly identical. Also, note that the polynomial (5.66) yields a very efficient preconditioner which reduces the number of iterations significantly in all examples. Finally, as already suspected in the previous section, the strategy leading to the ZPCG method is a very dangerous one, and the algorithm even fails to converge if A is close to a positive definite matrix.

Examples 7.6. Here A is a 16384 × 16384 matrix of the form (7.4) with $\sigma_1 = \sigma_2 = 100$. We applied the PPMR method based on the MR Algorithm 5.3 combined with polynomial preconditioning (5.66) of various degrees l. This example was run on a massively parallel computer, the CM-2, with 16,384 processors. In Figure 7.5, we plot the number of iterations after which the PPMR method had reached (7.3) versus l. In Figure 7.6, we plot the actual computing time (in seconds) versus l. Clearly, polynomial preconditioning is an efficient technique on the CM-2.

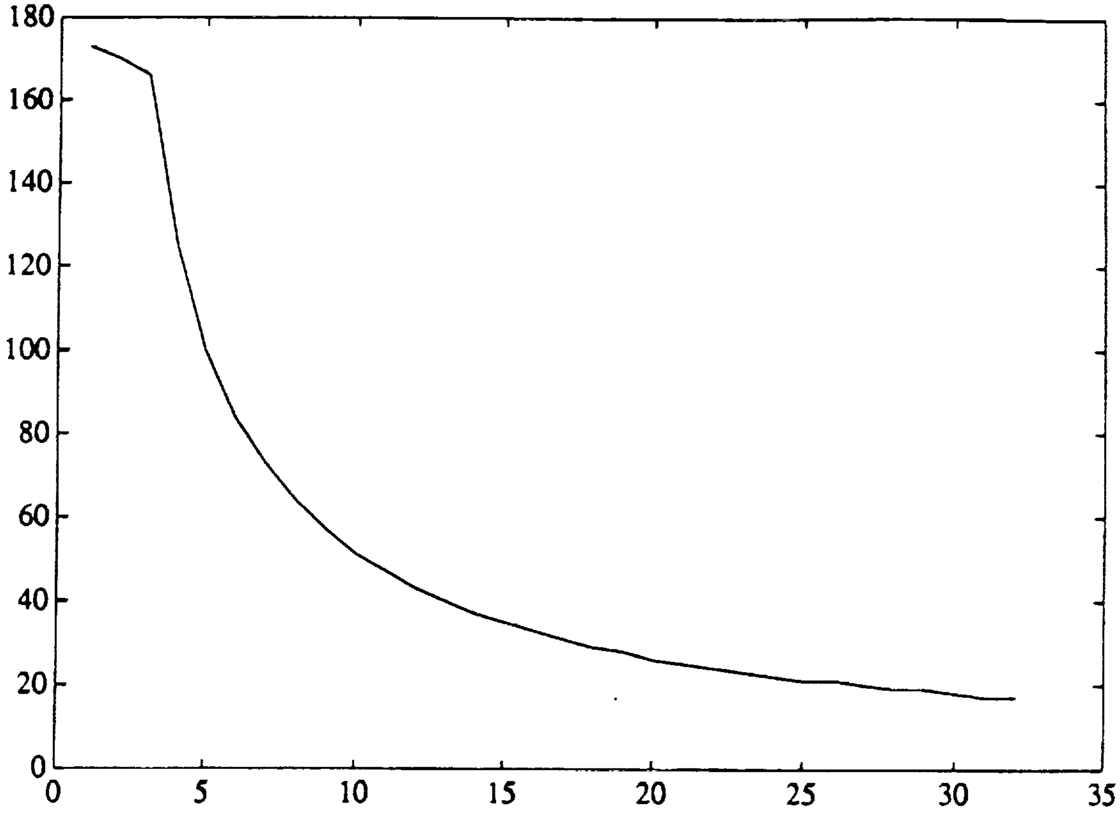

Figure 7.5. Number of iterations for PPMR versus the degree l of the preconditioner for Example 7.6.

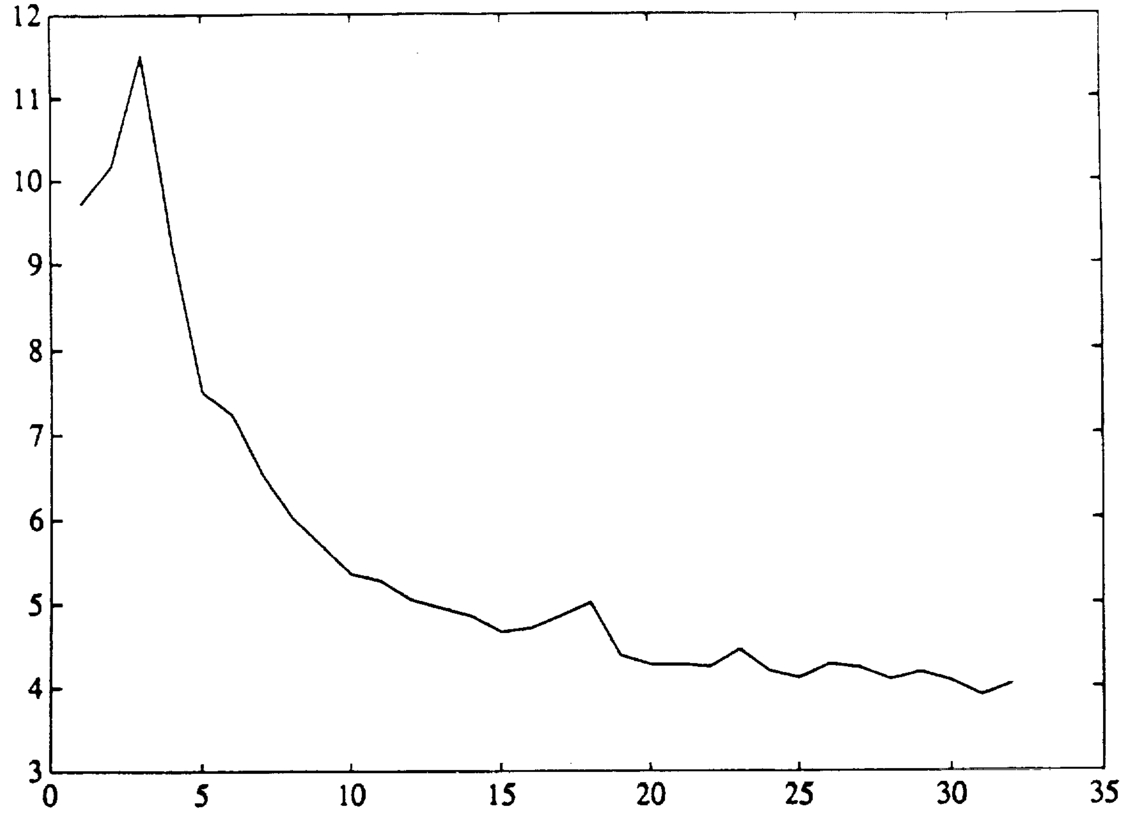

Figure 7.6. Actual computing time (in seconds) for PPMR on the CM-2 versus the degree l of the preconditioner for Example 7.6.

We conclude this section with two further remarks. all the results for the PPMR, PPME, and PPGAL methods were obtained with right polynomial preconditioning (RPP) (cf. (5.57)). Experiments with left polynomial preconditioning (LPP) (see (5.56)) gave nearly identical results. However, since implementations of RPP are slightly more economical, we therefore recommend RPP over LPP. Finally, recall that for our tests, the true extreme eigenvalues (7.5) of T were used. Of course, in general, such information is not available. However, it is possible to obtain good estimates of these quantities after relatively few steps of the Hermitian Lanczos Algorithm 5.1.

8. Concluding remarks

Complex non-Hermitian linear systems arise in important applications, such as the numerical solution of the complex Helmholtz equation. Often their coefficient matrices exhibit special structures, such as complex symmetry, or they are shifted Hermitian matrices. Here, we have considered Krylov subspace methods for the solution of complex non-Hermitian linear systems.

First, we have presented a novel Krylov subspace iteration, the QMR method, for general nonsingular non-Hermitian linear systems. The method uses a recently proposed [FGN, FN1] robust implementation of the look-ahead Lanczos algorithm to generate basis vectors for the Krylov subspaces $K_n(r_0, A)$. The QMR iterates are characterized by a quasi-minimal residual property over $K_n(r_0, A)$. Both the look-ahead Lanczos algorithm and the computation of the actual QMR iterates can be implemented using only short recurrences. The QMR approach is closely related to the BCG algorithm; however, unlike BCG, the QMR algorithm has smooth convergence curves and good numerical properties. Furthermore, we have derived bounds for the QMR residuals which are essentially the same as the standard bounds for GMRES. To the best of our knowledge, this is the first convergence result for a BCG-like algorithm for general non-Hermitian matrices.

Second, we discussed various CG-type methods designed for two special classes of complex non-Hermitian matrices. In particular, we have shown that work and storage for the QMR and BCG methods is roughly halved for complex symmetric linear systems. For shifted Hermitian matrices, we have investigated three different CG-type approaches with iterates defined by a minimal residual property, a Galerkin type condition, and an Euclidean error minimization. Numerically stable implementations were proposed and error bounds were derived for all three methods. Moreover, it was shown how the special shift structure can be preserved by using polynomial preconditioning, and results on the optimal choice of the polynomial preconditioner were given.

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.

An important question, that we have not addressed here, is how to construct efficient preconditioners for complex symmetric linear systems, such as the ones arising from the complex Helmholtz equation. This will be the subject of a forthcoming report.

Acknowledgments

First of all, I would like to thank Susanne for her infinite support and patience and for putting up with my crazy working hours over all these years. She also helped generously with the TEXing of this thesis.

I am grateful to Prof. Josef Stoer who first introduced me to conjugate gradient methods. Also, I would like to thank him for his understanding for several delays in finishing this thesis. I am indebted to Prof. Gene Golub who first brought the complex Helmholtz equation and complex linear systems to my attention.

Most of the work on the look-ahead Lanczos algorithm and the QMR method is joint with Nöel Nachtigal whom I wish to thank for the fruitful collaboration. I am grateful to Steve Hammond who implemented the MR algorithm for shifted Hermitian matrices on the CM-2 and generated Figures 7.5 and 7.6. Finally, I would like to thank Stratis Gallopoulos and Youcef Saad for providing some important references, and Marlis Hochbruck for her careful reading of parts of this thesis.

I gratefully acknowledge the financial support I received over the years. In particular, parts of this work were supported by the German Research Association (DFG), the National Science Foundation under Grant DCR-8412314, and by DARPA and NASA via Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

References

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

[BBGRM] Barbour, I.M., Behilil, N.-E., Gibbs, P.E., Rafiq, M., Moriarty, K.J.M.. Schierholz, G.: Updating fermions with the Lanczos method. *J. Comput. Phys.* 68. 227–236 (1987)

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

[BGoT] Bayliss, A., Goldstein, C.I., Turkel, E.: The numerical solution of the Helmholtz equation for wave propagation problems in underwater acoustics. *Comp. Maths. Appl.* **11**, 655–665 (1985)

[BGuT] Bayliss, A., Gunzburger, M., Turkel, E.: Boundary conditions for the numerical solution of elliptic equations in exterior regions. *SIAM J. Appl. Math.* **42**, 430–451 (1982)

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

[Bre] Brezinski, C.: Padé-type approximation and general orthogonal polynomials. Basel: Birkhäuser 1980

[BBG] Bunse, W., Bunse-Gerstner, A.: Numerische lineare Algebra. Stuttgart: Teubner 1985

[CT] Carlson, B.C., Todd, J.: Zolotarev's first problem – the best approximation by polynomials of degree $\leq n-2$ to $x^n - n\sigma x^{n-1}$ in $[-1,1]$. *Aequationes Math.* 26, 1–33 (1983)

[Cha] Chandra, R.: Conjugate gradient methods for partial differential equations. Ph. D. Thesis, Computer Science Department, Research Report 129, Yale University, January 1978

[CG] Concus, P., Golub, G.H.: A generalized conjugate gradient method for nonsymmetric systems of linear equations. In: Computing methods in applied sciences and engineering (R. Glowinski and J.L. Lions, eds.), pp. 56–65. Lecture Notes in Economics and Mathematical Systems 134. Berlin, Heidelberg, New York: Springer 1976

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

[CW] Cullum, J., Willoughby, R.A.: Lanczos algorithms for large symmetric eigenvalue computations. Volume 1, Theory Basel: Birkhäuser 1985

[DFP] Delfour, M., Fortin, M.. Payre, G.: Finite-difference solutions of a non-linear Schrödinger equation. *J. Comput. Phys.* **44**. 277–288 (1981)

[Dra] Draux, A.: Polynômes orthogonaux formels – applications. Lecture Notes in Mathematics, vol. 974, Berlin: Springer-Verlag 1983

[DGL] Duff, I.S., Grimes, R.G., Lewis, J.G.: Sparse matrix test problems. *ACM Trans. Math. Softw.* **15**, 1–14 (1989)

[DS] Duffin, R., Schaeffer, A.C.: Some properties of functions of exponential type. *Bull. Am. Math. Soc.* **44**, 236–240 (1938)

[ELV] Eiermann, M., Li, X., Varga, R.S.: On hybrid semiiterative methods. *SIAM J. Numer. Anal.* **26**, 152–168 (1989)

[ENV] Eiermann, M., Niethammer, W., Varga, R.S.: A study of semiiterative methods for nonsymmetric systems of linear equations. *Numer. Math.* **47**, 505–533 (1985)

[Eis] Eisenstat, S.C.: Some observations on the generalized conjugate gradient method. In: Numerical methods, Proceedings, Caracas 1982 (V. Pereyra and A. Reinoza, eds.), pp. 99–107. Lecture Notes in Mathematics 1005. Berlin, Heidelberg, New York: Springer 1983

[EES] Eisenstat, S.C., Elman, H.C., Schultz, M.H.: Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.* **20**, 345–357 (1983)

[EH] Elmore, W.C., Heald, M.A.: Physics of waves. New York: McGraw-Hill 1969

[FM1] Faber, V., Manteuffel, T.: Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.* **21**, 352–362 (1984)

[FM2] Faber, V., Manteuffel, T.: Orthogonal error methods. '*SIAM J. Numer. Anal.*. **24**, 170–187 (1987)

[FF] Fischer, B., Freund, R.: On the constrained Chebyshev approximation problem on ellipses. *J. Approx. Theory* **62**, 297–315 (1990)

[Fle] Fletcher, R.: Conjugate gradient methods for indefinite systems. In: Numerical Analysis Dundee 1975 (G.A. Watson, ed.), pp. 73–89. Lecture Notes in Mathematics 506. Berlin, Heidelberg, New York: Springer 1976

[Fre1] Freund, R.: Über einige cg-ähnliche Verfahren zur Lösung linearer Gleichungssysteme. Doctoral Thesis, Universität Würzburg, F.R. of Germany, May 1983

[Fre2] Freund, R.: Pseudo Ritz values for indefinite Hermitian matrices. Tech. Report 89.33, RIACS, NASA Ames Research Center, August 1989

[Fre3] Freund, R.: On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices. *Numer. Math.* **57**, 285–312 (1990)

[Fre4] Freund, R.: Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices. Technical Report 89.54, RIACS, NASA Ames Research Center, December 1989

[FGN] Freund, R.W., Gutknecht, M.H., Nachtigal, N.M.: An implementation of the look-ahead Lanczos algorithm, Part I. Technical Report 90.45, RIACS, NASA Ames Research Center, November 1990

[FN1] Freund, R. W., Nachtigal, N. M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part II. Technical Report 90.46, RIACS, NASA Ames Research Center, November 1990

[FN2] Freund, R. W., Nachtigal, N. M.: QMR: a quasi-minimal residual method for non-Hermitian linear systems. Technical Report 90.51, RIACS, NASA Ames Research Center, December 1990

[FR] Freund, R., Ruscheweyh, St.: On a class of Chebyshev approximation problems which arise in connection with a conjugate gradient type method. *Numer. Math.* 48, 525–542 (1986)

[Fri] Fridman, V.M.: The method of minimum iterations with minimum errors for a system of linear algebraic equations with a symmetrical matrix. *USSR Comput. Math. and Math. Phys.* **2**, 362–363 (1963)

[GS1] Gallopoulos, E., Saad, Y.: On the parallel solution of parabolic equations. In: Proc. 1989 ACM International Conference on Supercomputing, Herakleion, Greece, June 1989, pp. 17–28

[GS2] Gallopoulos, E., Saad, Y.: Efficient parallel solution of parabolic equations: implicit methods on the Cedar multicluster. In: Proc. Fourth SIAM Conf. Parallel Processing for Scientific Computing (J. Dongarra, P. Messina, D.C. Sorensen, and R.G. Voigt, eds.), pp. 251–256. Philadelphia: SIAM 1990

[Gan] Gantmacher, F.R.: The theory of matrices. Volume 2, New York: Chelsea Publishing Company 1959

[Gol] Goldstein, C.I.: Multigrid preconditioners applied to three-dimensional parabolic equation type models. In: Computational acoustics: wave propagation (D. Lee, R.L. Sternberg, M.H. Schultz, eds.), pp. 57–74. Amsterdam: North-Holland 1988

[GVL] Golub, G.H., Van Loan, C.F.: Matrix computations. First edition, Baltimore: The Johns Hopkins University Press 1983

[GV] Golub, G.H., Varga, R.S.: Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods. *Numer. Math.* **3**, 147–168 (1961)

[Gra] Gragg, W.B.: Matrix interpretations and applications of the continued fraction algorithm. *Rocky Mountain J. Math.* **4**. 213–225 (1974)

[GL] Gragg, W.B, Lindquist, A.: On the partial realization problem. *Linear Algebra Appl.* **50**, 277–319 (1983)

[Gut] Gutknecht, M.H.: A completed theory of the unsymmetric Lanczos process and related algorithms, Part I. IPS Research Report No. 90–10, Zürich. June 1990

[HS] Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.* **49**, 409–436 (1952)

[HJ] Horn, R.A., Johnson, C.R.: Matrix analysis. Cambridge: Cambridge University Press 1985

[Hou] Householder, A.S.: The theory of matrices in numerical analysis. New York: Blaisdell 1964

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

[JMP] Johnson, O.G., Micchelli, C.A., Paul, G.: Polynomial preconditioners for conjugate gradient calculations. *SIAM J. Numer. Anal.* **20**, 362–376 (1983)

[JY] Joubert, W.D., Young, D.M.: Necessary and sufficient conditions for the simplification of generalized conjugate-gradient algorithms. *Lin. Alg. Appl.* **88/89**, 449–485 (1987)

[KG] Keller, J.B., Givoli, D.: Exact non-reflecting boundary conditions. *J. Comput. Phys.* **82**, 172–192 (1989)

[Kun] Kung, S.: Multivariable and multidimensional systems: analysis and design. Ph.D. Dissertation, Stanford University, Stanford, June 1977

[Lan1] Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand.* **45**, 255–282 (1950)

[Lan2] Lanczos, C.: Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand.* **49**, 33–53 (1952)

[Lan3] Lanczos, C.: Chebyshev polynomials in the solution of large-scale linear systems. In: Proceedings of the Association for Computing Machinery, Toronto, 1952, pp. 124–133. Washington, D.C.: Sauls Lithograph Co. 1953

[Lan4] Lanczos, C.: Applied analysis. Englewood Cliffs, N.J.: Prentice-Hall 1956

[Lau] Laub, A.J.: Efficient multivariable frequency response computations. *IEEE Trans. Automat. Contr.* **AC-26**, 407–408 (1981)

[Man] Manteuffel, T.A.: The Tchebychev iteration for nonsymmetric linear systems. *Numer. Math.* **28**, 307–327 (1977)

[Mar] Marfurt, K.J.: Finite element modeling of elastodynamic and electromagnetic wave propagation for geophysical exploration. In: Computing methods in applied sciences and engineering VII (R. Glowinski and J.L. Lions, eds.), pp. 517–547. Amsterdam: North Holland 1986

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

[Mei] Meinardus, G.: Approximation of functions: Theory and numerical methods. Berlin, Heidelberg, New York: Springer 1967

[MF] Moro, G., Freed, J.H.: Calculation of ESR spectra and related Fokker-Planck forms by the use of the Lanczos algorithm. *J. Chem. Phys.* **74**, 3757–3773 (1981)

[PS] Paige, C.C., Saunders, M.A.: Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* **12**, 617–629 (1975)

[Par] Parlett, B.N.: Reduction to tridiagonal form and minimal realizations. Preprint. Berkeley, January 1990

[PTL] Parlett, B.N., Taylor, D.R., Liu, Z.A.: A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp.* **44**, 105–124 (1985)

[PM] Peterson, A.F., Mittra, R.: Method of conjugate gradients for the numerical solution of large-body electromagnetic scattering problems. *J. Opt. Soc. Am. A* **2**, 971–977 (1985)

[Pie] Pierce, A.D.: Acoustics: An introduction to its physical principles and applications. New York: McGraw-Hill 1981

[Rap] Rapoport, D.: A nonlinear Lanczos algorithm and the stationary Navier-Stokes equation. Ph. D. Thesis, Department of Mathematics, New York University, October 1978

[Rut] Rutishauser, H.: Theory of gradient methods. In: Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems, pp. 24–49. Mitteilungen aus dem Institut für Angewandte Mathematik an der ETH Zürich **8**. Basel: Birkhäuser 1959

[Saa1] Saad, Y.: The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems. *SIAM J. Numer. Anal.* **19**, 485–506 (1982)

[Saa2] Saad, Y.: Krylov subspace methods on supercomputers. *SIAM J. Sci. Stat. Comput.* **10**, 1200–1232 (1989)

[SS1] Saad, Y., Schultz, M.H.: Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Math. Comp.* **44**, 417–424 (1985)

[SS2] Saad, Y., Schultz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* 7, 856–869 (1986)

[SLJ] Schultz, M.H., Lee, D., Jackson, K.R.: Application of the Yale sparse technique to solve the 3-dimensional parabolic wave equation. In: Recent progress in the development and application of the parabolic equation (P.D. Scully-Power and D. Lee, eds.). Naval Underwater Systems Center, Technical Document 7145, May 1984

[Sid] Sidi, A.: Extrapolation vs. projection methods for linear systems of equations. *J. Comput. Appl. Math.* **22**, 71–88 (1988)

[SPM] Smith, C.F., Peterson, A.F., Mittra, R.: The biconjugate gradient method for electromagnetic scattering. *IEEE Trans. Antennas Propagat.* **AP-38**, 938–940 (1990)

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

[Sto] Stoer, J.: Solution of large linear systems of equations by conjugate gradient type methods. In: Mathematical programming – The state of the art (A. Bachem, M. Grötschel, and B. Korte, eds.), pp. 540–565. Berlin, Heidelberg, New York, Tokyo: Springer 1983

[SF] Stoer, J., Freund, R.: On the solution of large indefinite systems of linear equations by conjugate gradient algorithms. In: Computing methods in applied sciences and engineering V (R. Glowinski and J.L. Lions, eds.), pp. 35–53. Amsterdam: North Holland 1982

[Szy] Szyld, D.B.: A two-level iterative method for large sparse generalized eigenvalue calculations. Ph. D. Thesis, Department of Mathematics, New York University, October 1983

[SW] Szyld, D.B., Widlund, O.: Applications of conjugate gradient type methods to eigenvalue calculations. In: Advances in computer methods for partial differential equations III (R. Vichnevetsky and R.S. Stepleman, eds.), pp. 167–173. New Brunswick: IMACS 1979

[Tap] Tappert, F.D.: The parabolic approximation method. In: Wave propagation and underwater acoustics (J.B. Keller and J.S. Papadakis, eds.), pp. 224–287. Lecture Notes in Physics 70. Berlin, Heidelberg, New York: Springer 1977

[Tay] Taylor, D.R.: Analysis of the look ahead Lanczos algorithm. Ph.D. Dissertation, University of California, Berkeley, November 1982

[Tru] Trummer, M.: An efficient implementation of a conformal mapping method based on the Szegö kernel. *SIAM J. Numer. Anal.* **23**, 853–872 (1986)

[Van] van der Vorst, H.A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. Preprint, Utrecht, September 1990

C-2

[Voe] Voevodin, V.V.: The problem of a non-selfadjoint generalization of the conjugate gradient method has been closed. *USSR Comput. Math. and Math. Phys.* **23**. 143–144 (1983)

[Wid] Widlund, O.: A Lanczos method for a class of nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.* **15**, 801–812 (1978)

[Wil] Wilkinson, J.H.: The algebraic eigenvalue problem. Oxford: Oxford University Press 1965