Acta Numerica (1997), *pp.* 271–397

Lanczos-type solvers for nonsymmetric linear systems of equations

Martin H. Gutknecht
Swiss Center for Scientific Computing
ETH-Zentrum, CH-8092 Zürich, Switzerland
E-mail: mhg@scsc.ethz.ch

Among the iterative methods for solving large linear systems with a sparse (or, possibly, structured) nonsymmetric matrix, those that are based on the Lanczos process feature short recurrences for the generation of the Krylov space. This means low cost and low memory requirement. This review article introduces the reader not only to the basic forms of the Lanczos process and some of the related theory, but also describes in detail a number of solvers that are based on it, including those that are considered to be the most efficient ones. Possible breakdowns of the algorithms and ways to cure them by look-ahead are also discussed.

CONTENTS

1. Introduction

The task of solving huge sparse systems of linear equations comes up in many if not most large-scale problems of scientific computing. In fact, if tasks were judged according to hours spent on them on high-performance computers, the one of solving linear systems might be by far the most important one. There are two types of approach: direct methods, which are basically ingenious variations of Gaussian elimination, and iterative ones, which come in many flavours. The latter are the clear winners when we have to solve equations arising from the discretization of three-dimensional partial differential equations, while for two-dimensional problems none of the two approaches can claim to be superior in general.

Among the many existing iterative methods, those based on the Lanczos process – and we consider the conjugate gradient (CG) method to be included in this class – are definitely among the most effective ones. For symmetric positive definite systems, CG is normally the best choice, and

arguments among users are restricted to which preconditioning technique to use and whether it is worthwhile, or even necessary, to combine the method with other techniques such as domain decomposition or multigrid.

For nonsymmetric (or, more correctly, non-Hermitian) systems it would be hard to make a generally accepted recommendation. There are dozens of algorithms that are generalizations of CG or are at least related to it. They fall basically into two classes: (i) methods based on orthogonalization, many of which feature a minimality property of the residuals with respect to some norm, but have to make use of long recurrences involving all previously found iterates and residuals (or direction vectors), unless truncated or restarted, in which case the optimality is lost; and (ii) methods based on biorthogonalization (or, duality) that feature short recurrences and a competitive speed of convergence. It is the latter class that is the topic of this article. The gain in memory requirement and computational effort that comes from short recurrences is often crucial for making a problem solvable. While computers get faster and memory cheaper, users turn to bigger problems, so that efficient methods become more rather than less important.

The application of recursive biorthogonalization to the numerical solution of eigenvalue problems and linear systems goes back to Lanczos (1950, 1952) and is therefore referred to as the Lanczos process. In its basic form, the process generates a pair of biorthogonal (or, dual) bases for a pair of Krylov spaces, one generated by the coefficient matrix $\mathbf{A}$ and the other by its Hermitian transpose or adjoint $\mathbf{A}^*$. This process features a three-term recurrence and is here called Lanczos biorthogonalization or BIO algorithm (see Section 2). A variation of it, described in the second Lanczos paper, applies instead a pair of coupled two-term recurrences and is here referred to as BIOC algorithm, because it produces additionally a second pair of biconjugate bases (see Section 7). Both these algorithms can be applied for solving a linear system $\mathbf{Ax} = \mathbf{b}$ or for finding a part of the spectrum of $\mathbf{A}$. For the eigenvalue problem it has so far been standard to use the BIO algorithm, but there are indications that this may change in the future (see the comments in Section 7).

The emphasis here is neither on eigenvalue problems nor on symmetric linear systems, but on solving nonsymmetric systems. Although the determination of eigenvalues is based on the same process, its application to this problem has a different flavour and is well known for additional numerical difficulties. Moreover, for the eigenvalue problem, the spectrum of a tridiagonal matrix has to be determined in a postprocessing step. For generality, our formulations include complex systems, although the methods are mainly applied to real data.

For symmetric positive definite systems, the Lanczos process is equivalent to the conjugate gradient (CG) method of Hestenes and Stiefel (1952), which has been well understood for a long time and is widely treated in the

literature. Related algorithms for indefinite symmetric systems are also well known. Therefore, we can concentrate on the nonsymmetric case.

For an annotated bibliography of the early work on the CG and the Lanczos methods we refer to Golub and O'Leary (1989). The two Lanczos papers are briefly reviewed in Stewart (1994).

There are several ways of solving linear systems iteratively with the Lanczos process. Like any other Krylov space method, the Lanczos process generates a nested sequence of Krylov subspaces[1] $\mathcal{K}_n$: at each step, the so far created basis $\{\mathbf{y}_0, \ldots, \mathbf{y}_{n-1}\}$ is augmented by a new (right) Lanczos vector $\mathbf{y}_n$ that is a linear combination of $\mathbf{A}\mathbf{y}_{n-1}$ and the old basis. The starting vector $\mathbf{y}_0$ is the residual of some initial approximation $\mathbf{x}_0$, that is, $\mathbf{y}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$. The nth approximation $\mathbf{x}_n$ (the nth 'iterate') is then chosen to have a representation

$$\mathbf{x}_n = \mathbf{x}_0 + \sum_{j=0}^{n} \mathbf{y}_j \kappa_j \quad \text{so that} \quad \mathbf{x}_n - \mathbf{x}_0 \in \mathcal{K}_n. \tag{1.1}$$

This is what characterizes a Krylov space solver. Note, however, that the algorithms to be discussed will not make use of this representation since it would require us to store the whole Krylov space basis. It is a feature of all competitive Lanczos-type solvers that the iterates can be obtained with short recurrences.

The Lanczos process is special in that it generates two nested sequences of Krylov spaces, one, $\{\mathcal{K}_n\}$, from $\mathbf{A}$ and some $\mathbf{y}_0$, the other, $\{\widetilde{\mathcal{K}}_n\}$ from $\mathbf{A}^\star$ and some $\widetilde{\mathbf{y}}_0$. The iterates of the classical Lanczos-type solver, the biconjugate gradient (BiCG) method, are then characterized by $\widetilde{\mathcal{K}}_n \perp \mathbf{r}_n$, where $\mathbf{r}_n := \mathbf{b} - \mathbf{A}\mathbf{x}_n$ is the nth residual.

In the 'symmetric case', when $\mathbf{A}$ is Hermitian and $\widetilde{\mathbf{y}}_0 = \mathbf{y}_0$, so that $\widetilde{\mathcal{K}}_n = \mathcal{K}_n$ (for all n), this orthogonality condition is a Galerkin condition, and the method reduces to the conjugate gradient (CG) method, which is known to minimize the error in the $\mathbf{A}$-norm when the matrix is positive definite. Of course, the minimization is subject to the condition (1.1). By replacing in the symmetric case the orthogonality by $\mathbf{A}$-orthogonality, that is, imposing $\widetilde{\mathcal{K}}_n \perp \mathbf{A}\mathbf{r}_n$, we obtain the conjugate residual (CR) or minimum residual method – a particular algorithm due to Paige and Saunders (1975) is called MinRes, see Section 5 – with the property that the residual is minimized. As a consequence, the norm of the residuals decreases monotonically.

For non-Hermitian matrices $\mathbf{A}$ the two Krylov spaces are different, and $\widetilde{\mathcal{K}}_n \perp \mathbf{r}_n$ becomes a Petrov–Galerkin condition. In the BiCG method the short recurrences of CG and CR survive, but, unfortunately, the error and residual minimization properties are lost. In fact, in practice the norm of

[1] Throughout the paper we refer for simplicity mostly to Krylov spaces, not subspaces, as do many other authors.

the residuals sometimes increases suddenly by orders of magnitude, but also reduces soon after to the previous level. This is referred to as a *peak* in the residual norm plot, and when this happens several times in an example, BiCG is said to have an erratic convergence behaviour.

There are two good ways to achieve a smoother residual norm plot of BiCG. First, one can pipe the iterates $\mathbf{x}_n$ and the residuals $\mathbf{r}_n$ of BiCG through a simple smoothing process that determines smoothed iterates $\widetilde{\mathbf{x}}_n$ and $\widetilde{\mathbf{r}}_n$ according to

$$\widetilde{\mathbf{x}}_n := \widetilde{\mathbf{x}}_{n-1}(1-\theta_n) + \mathbf{x}_n\theta_n, \quad \widetilde{\mathbf{r}}_n := \widetilde{\mathbf{r}}_{n-1}(1-\theta_n) + \mathbf{r}_n\theta_n,$$

where θ_n is chosen such that the 2-norm of $\widetilde{\mathbf{r}}_n$ is as small as possible. This simple recursive weighting process is very effective. It was proposed by Schönauer (1987) and further investigated by Weiss (1990). Now it is referred to as minimal residual (MR) smoothing; we will discuss it in Section 17.

An alternative is the quasi-minimal residual (QMR) method of Freund and Nachtigal (1991), which does not use the BiCG iterates directly, but only the basis $\{\mathbf{y}_j\}$ of $\mathcal{K}_n$ that is generated by the Lanczos process. The basic idea is to determine iterates $\mathbf{x}_n$ so that the coordinate vector of their residual with respect to the Lanczos basis has minimum length. This turns out to be a least squares problem with an $(n+1) \times n$ tridiagonal matrix, the same problem as in MinRes; see Section 5.

QMR can also be understood as a harmonic mean smoothing process for the residual norm plot, and therefore, theoretically, neither QMR nor MR smoothing can really speed up the convergence considerably. In practice, however, it often happens that straightforward implementations of the BiCG method converge more slowly than a carefully implemented QMR algorithm (or not at all).

At this point we should add that there are various ways to realize the BiCG method (see Sections 4, 8 9), and that they all allow us to combine it with a smoothing process. In theory, the various algorithms are mathematically nearly equivalent, but with respect to round-off they differ. And round-off is indeed a serious problem for all Lanczos process based algorithms with short recurrences: when building up the dual bases we only enforce the orthogonality to the two previous vectors; the orthogonality to the earlier vectors is inherited and, with time, is more and more lost due to round-off. We will discuss this and other issues of finite precision arithmetic briefly in Section 18.

In contrast to smoothing, an idea due to Sonneveld (1989) really increases the speed of convergence. At once, it eliminates the following two disadvantages of the nonsymmetric Lanczos process: first, in addition to the subroutine for the product $\mathbf{Ax}$ that is required by all Krylov space solvers, BiCG also needs one for $\mathbf{A}^*\mathbf{x}$; second, each step of BiCG increases

the dimension of both $\mathcal{K}_n$ and $\widetilde{\mathcal{K}}_n$ by one and, naturally, needs two matrix-vector multiplications to do so, but only one of them helps to reduce the approximation error by increasing the dimension of the approximation space. To explain Sonneveld's idea we note that every vector in the Krylov space $\mathcal{K}_{n+1}$ has a representation of the form $p_n(\mathbf{A})\mathbf{y}_0$ with a polynomial of degree at most n. In the standard version of the BiCG method, the basis vectors generated are linked by this polynomial:

$$\mathbf{y}_n = p_n(\mathbf{A})\mathbf{y}_0, \quad \widetilde{\mathbf{y}}_n = \overline{p_n}(\mathbf{A}^\star)\widetilde{\mathbf{y}}_0.$$

Here, the bar denotes complex conjugation of the coefficients. Since $\mathbf{y}_n$ happens to be the residual $\mathbf{r}_n$ of $\mathbf{x}_n$, p_n is actually the residual polynomial. Sonneveld found with his conjugate gradient squared (CGS) algorithm a method where the nth residual is instead given by $\mathbf{r}_n = p_n^2(\mathbf{A})\mathbf{y}_0 \in \mathcal{K}_{2n}$. Per step it increases the Krylov space by two dimensions. Moreover, the transpose or adjoint matrix is no longer needed. In practice, the convergence is indeed typically nearly twice as fast as for BiCG, and thus also in terms of matrix-vector multiplications the method is nearly twice as effective. However, it turns out that the convergence is even more erratic. We will refer to this method more appropriately as biconjugate gradient squared (BiCGS) method and treat various forms of it in Section 14.

To get smoother convergence, van der Vorst (1992) modified the approach in his BiCGStab algorithm by choosing residuals of the form $\mathbf{r}_n = p_n(\mathbf{A})t_n(\mathbf{A})\mathbf{y}_0 \in \mathcal{K}_{2n}$, where the polynomials t_n are built up in factored form, with a new zero being added in each step in such a way that the residual undergoes a one-dimensional minimization process. This method was soon enhanced further and became the germ of a group of methods that one might call the BiCGStab family. It includes BiCGStab2 with two-dimensional minimization every other step, and the more general BiCGStab(ℓ) with ℓ-dimensional minimization after a compound step that costs 2ℓ matrix-vector products and increases the dimension of the Krylov space by 2ℓ. The BiCGStab family is a subset of an even larger class, the Lanczos-type product methods (LTPMs), which are characterized by residual polynomials that are products of a Lanczos polynomial p_n and another polynomial t_n of the same degree. All LTPMs are transpose-free and gain one dimension of the Krylov space per matrix-vector multiplication. Basically an infinite number of such methods exist, but it is not so easy to find one that can outperform, say, BiCGStab2. One that seems to be slightly better is based on the same idea as BiCGStab2 and, in fact, requires only the modification of a single condition in the code. We call it BiCG×MR2. It does a two-dimensional minimization in each step. LTPMs in general and the examples mentioned here are discussed in Section 16.

Another good idea is to apply the QMR approach suitably to BiCGS. The resulting transpose-free QMR (TFQMR) algorithm of Freund (1993)

can compete in efficiency with the best LTPMs; we treat it in Section 15. It is also possible to apply the QMR approach to LTPMs; see the introduction of Section 16 for references.

A disadvantage of Krylov space methods based on the Lanczos process is that they can break down for several reasons (even in exact arithmetic). Particularly troublesome are the 'serious' Lanczos breakdowns that occur when

$$\langle \tilde{\mathbf{y}}_n, \mathbf{y}_n \rangle = 0, \quad \text{but} \quad \tilde{\mathbf{y}}_n \neq \mathbf{o}, \ \ \mathbf{y}_n \neq \mathbf{o}.$$

Though true ('exact') breakdowns are extremely unlikely (except in contrived or specially structured examples), near-breakdowns can be (but need not be) the cause for an interruption or a slow-down of the convergence. These breakdowns were surely among the reasons why the BICG method was rarely used over decades. However, as problem sizes grew, it became more and more important to have a method with short recurrences. Fortunately, it turned out that there is a mathematically correct way to circumvent both exact and near-breakdowns. The first to come up with such a procedure for the BIO algorithm for eigenvalue computations were Parlett, Taylor and Liu (1985), who also coined the term 'look-ahead'. In their view, this was a generalization of the two-sided Gram–Schmidt algorithm. In 1988 look-ahead was rediscovered by Gutknecht from a completely different perspective: for him, look-ahead was a translation of general recurrences in the Padé table, and a realization of what Gragg (1974) had indicated in a few lines long before. Although classical Padé and continued fraction theory put no emphasis on singular cases, the recurrences for what corresponds to an exact breakdown had been known for a long time. Moreover, it was possible to generalize them in order to treat near-breakdowns. This theory and the application to several versions of the BICG method are compiled in Gutknecht (1992, 1994*a*). The careful implementation and the numerical tests of Freund, Gutknecht and Nachtigal (1993) ultimately proved, that all this can be turned into robust and efficient algorithms. Many other authors have also contributed to look-ahead Lanczos algorithms; see our references in Section 19. Moreover, the basic idea can be adapted to many other related recurrences in numerical analysis.

Here, we describe in Section 3 in detail the various ways in which the BIO algorithm can break down or terminate. Throughout the manuscript we then point out under which conditions the other algorithms break down. Although breakdowns are rare, knowing where and why they occur is important for learning how to avoid them and for gaining a full understanding of Lanczos-type algorithms. Quality software should be able to cope with breakdowns, or at least indicate to the user when a critical situation occurs. Seemingly, we only indicate exact breakdowns, but, of course, to include near-breakdowns the conditions '$= \mathbf{o}$' and '$= 0$' just have to be replaced

by other appropriate conditions. In practice, however, the question of what 'appropriate' means is not always so easy to answer. It is briefly touched on in Section 19.

The derivations of the look-ahead BIO and BIOC algorithms we present are based on the interpretation as two-sided Gram–Schmidt process. The algorithms themselves are improved versions with reduced memory requirement. The look-ahead BIO algorithm makes use of a trick hidden in Freund and Zha (1993) and pointed out in Hochbruck (1996). We establish this simplification in a few lines, making use of a result from Gutknecht (1992). For the look-ahead BIOC algorithm the simplification is due to Hochbruck (1996), but also proved here in a different way.

We give many pointers to the vast literature on Krylov space solvers based on the Lanczos process, but treating all the algorithms proposed is far beyond our scope. In fact, this literature has grown so much in the last few years that we have even had to limit the number of references. Moreover, there must exist papers we are unaware of. Our intention is mainly to give an easy introduction to the nonsymmetric Lanczos process and some of the linear system solvers based on it. We have tried in particular to explain the underlying ideas and to make clear the limitations and difficulties of this family of methods. We have chosen to treat those algorithms that we think are important for understanding, as well as those we think are among the most effective.

There are many aspects that are not covered or only briefly referred to. First of all, the important question of convergence and its deterioration due to numerical effects is touched on only superficially. Also, we do not give any numerical results, since giving a representative set would have required considerable time and space. In fact, while straightforward implementation of the algorithms requires little work, we believe that the production of quality software taking into account some of the possible enhancements we mention in Section 18 – not to speak of the look-ahead option that should be included – requires considerable effort. Testing and evaluating such software is not easy either, as simple examples do not show the effects of the enhancements, while complicated ones make it difficult to link causes and effects. To our knowledge, the best available software is Freund and Nachtigal's QMRPACK, which is freely available from NETLIB, but is restricted to various forms of the QMR and TFQMR methods; see Freund and Nachtigal (1996).

While we are aware that in practice preconditioning can improve the convergence of an iterative method dramatically and can reduce the overall costs, we describe only briefly in Section 11 how preconditioners can be integrated into the algorithms, but not how they are found. For a survey of preconditioning techniques we refer, for example, to Saad (1996).

Also among the things we skip are the adaptations of the Lanczos method to systems with multiple right-hand sides. Recent work includes Aliaga,

Hernandez and Boley (1994) and Freund and Malhotra (1997), which is an extension of Ruhe's band Lanczos algorithm (Ruhe 1979) blended with QMR.

The Lanczos process is closely linked to other topics with the same mathematical background, in particular, formal orthogonal polynomials, Padé approximation, continued fractions, the recursive solution of linear systems with Hankel matrix, and the partial realization problem of system theory. We discuss formal orthogonal polynomials (FOPs) briefly in Section 12, but the other topics are not touched on. For the Padé connection, which is very helpful for understanding breakdowns and the look-ahead approach to cure them, we refer to Gutknecht (1994*b*) for a simple treatment. The relationship is described in much more detail and capitalized upon in Gutknecht (1992, 1994*a*), where Lanczos look-ahead for both the BIO and the BIOC process was introduced based on this connection. The relations between various look-ahead recurrences in the Padé table and variations of look-ahead Lanczos algorithms are also a topic of Hochbruck (1996). A fast Hankel solver based on look-ahead Lanczos was worked out in detail in Freund and Zha (1993). For the connection to the partial realization problem; see Boley (1994), Boley and Golub (1991), Golub, Kågström and Van Dooren (1992), and Parlett (1992).

In the wider neighbourhood of the Lanczos process we also find the Euclidean algorithm, Gauss quadrature, the matrix moment problem and its modified form, certain extrapolation methods, and the interpretation of conjugate direction methods as a special form of Gauss elimination. But these subjects are not treated here.

A preliminary version of a few sections of this overview was presented as an invited talk at the Copper Mountain Conference on Iterative Methods 1990 and was printed in the *Preliminary Proceedings* distributed at the conference. The present Section 14 (plus some additional material) was made available on separate sheets at that conference. We refer here to these two sources as Gutknecht (1990).

Notation

Matrices and vectors are denoted by upper and lower case boldface letters. In particular, $\mathbf{O}$ and $\mathbf{o}$ are the zero matrix and vector, respectively. The transpose of $\mathbf{A}$ is $\mathbf{A}^\top$; its conjugate transpose is $\mathbf{A}^\star$. Blocks of block vectors and matrices are sometimes printed in roman instead of boldface. For coefficients and other scalars we normally choose lower case Greek letters. However, for polynomials (and some function values) we use lower case roman letters. Sets are denoted by calligraphic letters; for instance, $\mathcal{P}_n$ is the set of polynomials of degree at most n.

We write scalars on the right-hand side of vectors, for instance $\mathbf{x}\alpha$, so that this product becomes a special case of matrix multiplication[2]. The inner product in $\mathbb{C}^N$ is assumed to be defined by $\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^\star \mathbf{y}$, so that $\langle \mathbf{x}\alpha, \mathbf{y}\beta \rangle = \overline{\alpha}\beta \langle \mathbf{x}, \mathbf{y} \rangle$. The symbol := is used both for definitions and for algorithmic assignments.

To achieve a uniform nomenclature for the methods and algorithms we consider, we have modified some of the denominations used by other authors (but we also refer to the original name if the identity is not apparent). We hope that this will help readers to find their way in the maze of Lanczos-type solvers.

2. The unsymmetric Lanczos or BIO algorithm

In this section we describe the basic nonsymmetric Lanczos process that generates a pair of biorthogonal vector sequences. These define a pair of nested sequences of Krylov spaces, one generated by the given matrix or operator $\mathbf{A}$, the other by its Hermitian transpose $\mathbf{A}^\star$. The Lanczos algorithm is based on a successive extension of these two spaces, coupled with a two-sided Gram–Schmidt process for the construction of dual bases for them. The recurrence coefficients are the elements of a tridiagonal matrix that is, in theory, similar to $\mathbf{A}$ if the algorithm does not stop early. In the next section we will discuss the various ways in which the process can terminate or break down.

2.1. Derivation of the BIO *algorithm*

Let $\mathbf{A} \in \mathbb{C}^{N \times N}$ be any real or complex $N \times N$ matrix, and let $\mathbf{B}$ be a nonsingular matrix that commutes with $\mathbf{A}$ and is used to define the formal (*i.e.*, not necessarily symmetric definite) inner product $\langle \widetilde{\mathbf{y}}, \mathbf{y} \rangle_{\mathbf{B}} := \widetilde{\mathbf{y}}^\star \mathbf{B} \mathbf{y}$ on $\mathbb{C}^N \times \mathbb{C}^N$. Orthogonality will usually be referred to with respect to this formal inner product. For simplicity, we do *not* call it $\mathbf{B}$-orthogonality, except when this inaccuracy becomes misleading. However, we are mostly interested in the case where $\mathbf{B}$ is the identity $\mathbf{I}$ and thus $\langle \widetilde{\mathbf{y}}, \mathbf{y} \rangle_{\mathbf{B}}$ is the ordinary Euclidean inner product of $\mathbb{C}^N$. The case $\mathbf{B} = \mathbf{A}$ will also play a role. Due to $\mathbf{AB} = \mathbf{BA}$ we will have $\langle \widetilde{\mathbf{y}}, \mathbf{A}\mathbf{y} \rangle_{\mathbf{B}} = \langle \mathbf{A}^\star \widetilde{\mathbf{y}}, \mathbf{y} \rangle_{\mathbf{B}}$. Finally, we let $\widetilde{\mathbf{y}}_0, \mathbf{y}_0 \in \mathbb{C}^N$ be two non-orthogonal initial vectors: $\langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle_{\mathbf{B}} \neq 0$.

The *Lanczos biorthogonalization* (BIO) *algorithm*, called *method of minimized iterations* by Lanczos (1950, 1952), but often referred to as the *unsymmetric* or *two-sided Lanczos algorithm*, is a process for generating two finite sequences $\{\mathbf{y}_n\}_{n=0}^{\nu-1}$ and $\{\widetilde{\mathbf{y}}_n\}_{n=0}^{\nu-1}$, whose length ν depends on $\mathbf{A}$, $\mathbf{B}$,

[2] To see the rationale, think of the case where α turns out to be an inner product or where we generalize to block algorithms and α becomes a block of a vector.

$\mathbf{y}_0$, and $\tilde{\mathbf{y}}_0$, such that, for $m, n = 0, 1, \ldots, \nu - 1$,

$$\begin{aligned} \mathbf{y}_n &\in \mathcal{K}_{n+1} := \operatorname{span}(\mathbf{y}_0, \mathbf{A}\mathbf{y}_0, \ldots, \mathbf{A}^n\mathbf{y}_0), \\ \tilde{\mathbf{y}}_m &\in \tilde{\mathcal{K}}_{m+1} := \operatorname{span}(\tilde{\mathbf{y}}_0, \mathbf{A}^\star\tilde{\mathbf{y}}_0, \ldots, (\mathbf{A}^\star)^m\tilde{\mathbf{y}}_0) \end{aligned} \tag{2.1}$$

and

$$\langle \tilde{\mathbf{y}}_m, \mathbf{y}_n \rangle_{\mathbf{B}} = \begin{cases} 0 & \text{if} \quad m \neq n, \\ \delta_n \neq 0 & \text{if} \quad m = n. \end{cases} \tag{2.2}$$

$\mathcal{K}_n$ and $\tilde{\mathcal{K}}_m$ are *Krylov (sub)spaces* of $\mathbf{A}$ and $\mathbf{A}^\star$, respectively. The condition (2.2) means that the sequences are *biorthogonal*. Their elements $\mathbf{y}_n$ and $\tilde{\mathbf{y}}_m$ are called *right* and *left Lanczos vectors*, respectively. In view of (2.1) $\mathbf{y}_n$ and $\tilde{\mathbf{y}}_n$ can be expressed as

$$\mathbf{y}_n = p_n(\mathbf{A})\mathbf{y}_0, \quad \tilde{\mathbf{y}}_n = \tilde{p}_n(\mathbf{A}^\star)\tilde{\mathbf{y}}_0, \tag{2.3}$$

where p_n and $\tilde{p}_n$ are polynomials of degree at most n into which $\mathbf{A}$ and $\mathbf{A}^\star$ are substituted for the variable. We call p_n the nth *Lanczos polynomial*[3]. We will see in a moment that p_n has exact degree n and that the sequence $\{\tilde{\mathbf{y}}_m\}$ can be chosen such that $\tilde{p}_n = \overline{p_n}$, where $\overline{p_n}$ is the polynomial with the complex conjugate coefficients. In the general case, $\tilde{p}_n$ will be seen to be a scalar multiple of $\overline{p_n}$.

The biorthogonal sequences of Lanczos vectors are constructed by a two-sided version of the well-known Gram–Schmidt process, but the latter is not applied to the bases used in the definition (2.1), since these are normally very ill-conditioned, that is, close to linearly dependent. Instead, the vectors that are orthogonalized are of the form $\mathbf{A}\mathbf{y}_n$ and $\mathbf{A}^\star\tilde{\mathbf{y}}_n$; that is, they are created in each step from the most recently constructed pair by multiplication with $\mathbf{A}$ and $\mathbf{A}^\star$, respectively.

The length ν of the sequences is determined by the impossibility of extending them such that the conditions (2.1) and (2.2) still hold with $\delta_n \neq 0$. We will discuss the various reasons for a termination or a breakdown of the process below.

Clearly $\mathcal{K}_{n+1} \supseteq \mathcal{K}_n$, $\tilde{\mathcal{K}}_{n+1} \supseteq \tilde{\mathcal{K}}_n$, and from (2.2) it follows that equality cannot hold when $n < \nu$, since $\langle \tilde{\mathbf{y}}, \mathbf{y}_n \rangle_{\mathbf{B}} = 0$ for all $\tilde{\mathbf{y}} \in \tilde{\mathcal{K}}_n$ and $\langle \tilde{\mathbf{y}}_n, \mathbf{y} \rangle_{\mathbf{B}} = 0$ for all $\mathbf{y} \in \mathcal{K}_n$, or, briefly,

$$\tilde{\mathcal{K}}_n \perp_{\mathbf{B}} \mathbf{y}_n, \quad \tilde{\mathbf{y}}_n \perp_{\mathbf{B}} \mathcal{K}_n, \tag{2.4}$$

but $\tilde{\mathcal{K}}_{n+1} \not\perp_{\mathbf{B}} \mathbf{y}_n$, $\tilde{\mathbf{y}}_n \not\perp_{\mathbf{B}} \mathcal{K}_{n+1}$. Consequently, for $n = 1, \ldots, \nu - 1$,

$$\mathbf{y}_n \in \mathcal{K}_{n+1} \backslash \mathcal{K}_n, \quad \tilde{\mathbf{y}}_n \in \tilde{\mathcal{K}}_{n+1} \backslash \tilde{\mathcal{K}}_n \tag{2.5}$$

[3] In classical analysis, depending on the normalization, the Lanczos polynomials are called *Hankel polynomials* or *Hadamard polynomials*, the latter being monic (Henrici 1974).

(where the backslash denotes the set-theoretic difference), and thus

$$\mathcal{K}_{n+1} = \text{span}\,(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_n), \quad \widetilde{\mathcal{K}}_{n+1} = \text{span}\,(\widetilde{\mathbf{y}}_0, \widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_n). \tag{2.6}$$

Note that the relation on the left of (2.4) is equivalent to $\mathbf{y}_n \perp_{\mathbf{B}^\star} \widetilde{\mathcal{K}}_n$, but not to $\mathbf{y}_n \perp_{\mathbf{B}} \widetilde{\mathcal{K}}_n$, in general.

From (2.5) and (2.6) we conclude that for some complex constants $\tau_{k,n-1}$, $\widetilde{\tau}_{k,n-1}$ $(k = 0, \ldots, n;\ n = 1, \ldots, \nu - 1)$

$$\begin{aligned} \mathbf{y}_n \tau_{n,n-1} &= \mathbf{A}\mathbf{y}_{n-1} - \mathbf{y}_{n-1}\tau_{n-1,n-1} - \cdots - \mathbf{y}_0 \tau_{0,n-1}, \\ \widetilde{\mathbf{y}}_n \widetilde{\tau}_{n,n-1} &= \mathbf{A}^\star \widetilde{\mathbf{y}}_{n-1} - \widetilde{\mathbf{y}}_{n-1}\widetilde{\tau}_{n-1,n-1} - \cdots - \widetilde{\mathbf{y}}_0 \widetilde{\tau}_{0,n-1}, \end{aligned} \tag{2.7}$$

where $\tau_{n,n-1} \neq 0$ and $\widetilde{\tau}_{n,n-1} \neq 0$ can be chosen arbitrarily, for instance, for normalizing $\mathbf{y}_n$ and $\widetilde{\mathbf{y}}_n$. The choice will of course affect the constants δ_m and the coefficients $\tau_{n,m}$ and $\widetilde{\tau}_{n,m}$ for $m \geq n$, but only in a transparent way.

For $n = \nu$, (2.7) can still be satisfied for any nonzero values of $\tau_{\nu,\nu-1}$ and $\widetilde{\tau}_{\nu,\nu-1}$ by some $\mathbf{y}_\nu \perp_{\mathbf{B}^\star} \widetilde{\mathcal{K}}_\nu$ and some $\widetilde{\mathbf{y}}_\nu \perp_{\mathbf{B}} \mathcal{K}_\nu$, but these two vectors may be orthogonal to each other, so that (2.2) does not hold. In particular, one or even both may be the zero vector $\mathbf{o}$. Nevertheless, (2.4) and (2.6)–(2.7) also hold for $n = \nu$.

For $n \leq \nu$, let us introduce the $N \times n$ matrices

$$\mathbf{Y}_n := [\ \mathbf{y}_0 \quad \mathbf{y}_1 \quad \cdots \quad \mathbf{y}_{n-1}\ ], \quad \widetilde{\mathbf{Y}}_n := [\ \widetilde{\mathbf{y}}_0 \quad \widetilde{\mathbf{y}}_1 \quad \cdots \quad \widetilde{\mathbf{y}}_{n-1}\ ], \tag{2.8}$$

the diagonal matrices

$$\mathbf{D}_{\delta;n} := \text{diag}(\delta_0, \delta_1, \ldots, \delta_{n-1}),$$

and the $n \times n$ Hessenberg matrices $\mathbf{T}_n := [\tau_{k,j}]_{k,j=0}^{n-1}$ and $\widetilde{\mathbf{T}}_n := [\widetilde{\tau}_{k,j}]_{k,j=0}^{n-1}$. Then (2.2) and (2.7) become

$$\widetilde{\mathbf{Y}}_\nu^\star \mathbf{B} \mathbf{Y}_\nu = \mathbf{D}_{\delta;\nu} \tag{2.9}$$

and

$$\mathbf{A}\mathbf{Y}_\nu = \mathbf{Y}_\nu \mathbf{T}_\nu + \mathbf{y}_\nu \gamma_{\nu-1} \mathbf{l}_\nu^\top, \quad \mathbf{A}^\star \widetilde{\mathbf{Y}}_\nu = \widetilde{\mathbf{Y}}_\nu \widetilde{\mathbf{T}}_\nu + \widetilde{\mathbf{y}}_\nu \widetilde{\gamma}_{\nu-1} \mathbf{l}_\nu^\top, \tag{2.10}$$

where $\gamma_{n-1} := \tau_{n,n-1}$ and $\widetilde{\gamma}_{n-1} := \widetilde{\tau}_{n,n-1}$, and where $\mathbf{l}_n^\top$ is the last row of the $n \times n$ identity matrix. $n \leq \nu$. Using (2.9) and $\widetilde{\mathcal{K}}_\nu \perp_{\mathbf{B}} \mathbf{y}_\nu$, $\widetilde{\mathbf{y}}_\nu \perp_{\mathbf{B}} \mathcal{K}_\nu$ we conclude further that

$$\widetilde{\mathbf{Y}}_\nu^\star \mathbf{B} \mathbf{A} \mathbf{Y}_\nu = \mathbf{D}_{\delta;\nu} \mathbf{T}_\nu, \quad \mathbf{Y}_\nu^\star \mathbf{B}^\star \mathbf{A}^\star \widetilde{\mathbf{Y}}_\nu = \overline{\mathbf{D}_{\delta;\nu}} \widetilde{\mathbf{T}}_\nu, \tag{2.11}$$

and hence

$$\mathbf{D}_{\delta;\nu} \mathbf{T}_\nu = \widetilde{\mathbf{T}}_\nu^\star \mathbf{D}_{\delta;\nu}. \tag{2.12}$$

Here, on the left-hand side we have a lower Hessenberg matrix, while on the right-hand side there is an upper Hessenberg matrix. Consequently, both

$\mathbf{T}_\nu$ and $\widetilde{\mathbf{T}}_\nu$ must be tridiagonal. We simplify the notation by letting

$$\mathbf{T}_n =: \begin{bmatrix} \alpha_0 & \beta_0 & & & \\ \gamma_0 & \alpha_1 & \beta_1 & & \\ & \gamma_1 & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \beta_{n-2} \\ & & & \gamma_{n-2} & \alpha_{n-1} \end{bmatrix},$$

and naming the elements of $\widetilde{\mathbf{T}}_n$ analogously with tildes.

By comparing diagonal and off-diagonal elements on both sides of (2.12) we obtain

$$\alpha_n = \overline{\widetilde{\alpha}_n} \tag{2.13}$$

and

$$\delta_n \beta_n = \delta_{n+1} \overline{\widetilde{\gamma}_n}, \quad \delta_{n+1} \gamma_n = \delta_n \overline{\widetilde{\beta}_n}. \tag{2.14}$$

Hence,

$$\widetilde{\beta}_n \widetilde{\gamma}_n = \overline{\beta_n \gamma_n}, \tag{2.15}$$

which could be satisfied by setting[4]

$$\widetilde{\beta}_n := \overline{\beta_n}, \quad \widetilde{\gamma}_n := \overline{\gamma_n}, \quad i.e., \quad \widetilde{\mathbf{T}}_\nu := \overline{\mathbf{T}_\nu}. \tag{2.16}$$

This would further allow us to set $\gamma_n := 1$ (for all n) Another frequent choice is

$$\widetilde{\beta}_n := \overline{\gamma_n}, \quad \widetilde{\gamma}_n := \overline{\beta_n}, \quad i.e., \quad \widetilde{\mathbf{T}}_\nu := \mathbf{T}_\nu^\star, \tag{2.17}$$

which according to (2.14) yields $\delta_n = \delta_0$ for all n.

However, we want to keep the freedom to scale $\mathbf{y}_n$ and $\widetilde{\mathbf{y}}_n$, since most algorithms that apply the Lanczos process for solving linear systems make use of it. (As, for instance, the standard BICG algorithm discussed in Section 9 and the QMR approach of Section 5). Moreover, $\gamma_n = 1$ (for all n) can cause overflow or underflow. In view of (2.14), we have in general

$$\beta_n = \overline{\widetilde{\gamma}_n} \delta_{n+1} / \delta_n, \quad \widetilde{\beta}_n = \overline{\gamma_n \delta_{n+1} / \delta_n} = \overline{\beta_n \gamma_n} / \widetilde{\gamma}_n, \tag{2.18}$$

in accordance with (2.15). The choice between (2.16) and (2.17), and more generally, any choice that can be made to satisfy (2.14)–(2.15) only affects the scaling (including the sign, or, in the complex case, the argument of all components) of $\mathbf{y}_n$ and $\widetilde{\mathbf{y}}_n$. As we will see, it just leads to diagonal similarity transformations of the matrices $\mathbf{T}_\nu$ and $\widetilde{\mathbf{T}}_\nu$.

[4] By replacing the Euclidean inner product $\langle \widetilde{\mathbf{y}}, \mathbf{y} \rangle = \widetilde{\mathbf{y}}^\star \mathbf{y}$ by the bilinear form $\widetilde{\mathbf{y}}^\top \mathbf{y}$, and replacing $\mathbf{A}^\star$ by $\mathbf{A}^\top$, we could avoid the complex conjugation of the coefficients of $\mathbf{A}$ and of the recurrence coefficients; see, for instance, Freund et al. (1993). We prefer here to use the standard inner product.

After shifting the index n by 1, (2.7) can now be written as

$$\begin{aligned} \mathbf{y}_{n+1}\gamma_n &= \mathbf{A}\mathbf{y}_n - \mathbf{y}_n\alpha_n - \mathbf{y}_{n-1}\beta_{n-1}, \\ \widetilde{\mathbf{y}}_{n+1}\widetilde{\gamma}_n &= \mathbf{A}^\star\widetilde{\mathbf{y}}_n - \widetilde{\mathbf{y}}_n\widetilde{\alpha}_n - \widetilde{\mathbf{y}}_{n-1}\widetilde{\beta}_{n-1}, \end{aligned} \tag{2.19}$$

$n = 0, \ldots, \nu - 1$, with $\mathbf{y}_{-1} := \widetilde{\mathbf{y}}_{-1} := \mathbf{o}$, $\beta_{-1} := \widetilde{\beta}_{-1} := 0$. Taking inner products of the first formula with $\widetilde{\mathbf{y}}_{n-1}$, $\widetilde{\mathbf{y}}_n$, and $\widetilde{\mathbf{y}}_{n+1}$ we get three relations involving δ_{n+1}, δ_n, δ_{n-1}, and the recurrence coefficients α_n, β_{n-1}, γ_n. In particular, $\alpha_n = \langle \widetilde{\mathbf{y}}_n, \mathbf{A}\mathbf{y}_n\rangle_{\mathbf{B}} / \delta_n$ and, as is seen by inserting $\mathbf{A}^\star\widetilde{\mathbf{y}}_{n-1}$ according to (2.19),

$$\beta_{n-1} = \langle \widetilde{\mathbf{y}}_{n-1}, \mathbf{A}\mathbf{y}_n\rangle_{\mathbf{B}} / \delta_{n-1} = \langle \mathbf{A}^\star\widetilde{\mathbf{y}}_{n-1}, \mathbf{y}_n\rangle_{\mathbf{B}} / \delta_{n-1} = \overline{\widetilde{\gamma}_{n-1}}\delta_n/\delta_{n-1} \tag{2.20}$$

as in (2.18). Since (2.13) and the second relation in (2.18) determine $\widetilde{\alpha}_n$ and $\widetilde{\beta}_{n-1}$ we are left with two degrees of freedom that can be used to fix two of the three coefficients γ_n, $\widetilde{\gamma}_n$, and δ_{n+1}. We just exploit here the fact that the relations (2.9)–(2.11) can be used to determine $\mathbf{Y}_\nu$, $\widetilde{\mathbf{Y}}_\nu$, $\mathbf{T}_\nu$, $\widetilde{\mathbf{T}}_\nu$, and $\mathbf{D}_{\delta;\nu}$ column by column. Altogether we get the following general version of the unsymmetric Lanczos or biorthogonalization (BIO) algorithm.

ALGORITHM 1. (BIO ALGORITHM)
Choose $\mathbf{y}_0, \widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0\rangle_{\mathbf{B}} \neq 0$, and set $\beta_{-1} := 0$. For $n = 0, 1, \ldots$ compute

$$\alpha_n := \langle \widetilde{\mathbf{y}}_n, \mathbf{A}\mathbf{y}_n\rangle_{\mathbf{B}} / \delta_n, \tag{2.21a}$$

$$\widetilde{\alpha}_n := \overline{\alpha_n}, \tag{2.21b}$$

$$\beta_{n-1} := \overline{\widetilde{\gamma}_{n-1}}\delta_n/\delta_{n-1} \quad (\text{if } n > 0), \tag{2.21c}$$

$$\widetilde{\beta}_{n-1} := \overline{\gamma_{n-1}\delta_n/\delta_{n-1}} = \overline{\beta_{n-1}\gamma_{n-1}}/\widetilde{\gamma}_{n-1} \quad (\text{if } n > 0), \tag{2.21d}$$

$$\mathbf{y}_{\text{temp}} := \mathbf{A}\mathbf{y}_n - \mathbf{y}_n\alpha_n - \mathbf{y}_{n-1}\beta_{n-1}, \tag{2.21e}$$

$$\widetilde{\mathbf{y}}_{\text{temp}} := \mathbf{A}^\star\widetilde{\mathbf{y}}_n - \widetilde{\mathbf{y}}_n\widetilde{\alpha}_n - \widetilde{\mathbf{y}}_{n-1}\widetilde{\beta}_{n-1}, \tag{2.21f}$$

$$\delta_{\text{temp}} := \langle \widetilde{\mathbf{y}}_{\text{temp}}, \mathbf{y}_{\text{temp}}\rangle_{\mathbf{B}}; \tag{2.21g}$$

if $\delta_{\text{temp}} = 0$, choose $\gamma_n \neq 0$ and $\widetilde{\gamma}_n \neq 0$, set

$$\nu := n + 1, \quad \mathbf{y}_\nu := \mathbf{y}_{\text{temp}}/\gamma_n, \quad \widetilde{\mathbf{y}}_\nu := \widetilde{\mathbf{y}}_{\text{temp}}/\widetilde{\gamma}_n, \quad \delta_{n+1} := 0, \tag{2.21h}$$

and stop; otherwise, choose $\gamma_n \neq 0$, $\widetilde{\gamma}_n \neq 0$, and δ_{n+1} such that

$$\gamma_n\overline{\widetilde{\gamma}_n}\delta_{n+1} = \delta_{\text{temp}}, \tag{2.21i}$$

set

$$\mathbf{y}_{n+1} := \mathbf{y}_{\text{temp}}/\gamma_n, \quad \widetilde{\mathbf{y}}_{n+1} := \widetilde{\mathbf{y}}_{\text{temp}}/\widetilde{\gamma}_n, \tag{2.21j}$$

and proceed with the next step.

The definitions in (2.21h) will guarantee that formulae we derive below remain valid for $n = \nu$.

In exact arithmetic, which we assume throughout most of the text, the following basic result holds.

Theorem 2.1 The two sequences $\{\mathbf{y}_n\}_{n=0}^{\nu-1}$ and $\{\widetilde{\mathbf{y}}_n\}_{n=0}^{\nu-1}$ generated by the BIO algorithm satisfy (2.1) and (2.2), and hence (2.4)–(2.5). Moreover, (2.1) and (2.2) also hold when $m = \nu$ or $n = \nu$, except that $\delta_\nu = 0$.

Conversely, if (2.1) and (2.2) hold for $m, n = 0, \ldots, \nu$, except that $\delta_\nu = 0$, and if the choice (2.16) for satisfying (2.15) is made, then the recurrence formulae of the BIO algorithm are valid for $n = 0, \ldots, \nu - 1$, but the algorithm will stop at $n = \nu - 1$ due to $\delta_{\text{temp}} = 0$. The conditions (2.1) and (2.2) determine $\{\mathbf{y}_n\}_{n=0}^{\nu}$ and $\{\widetilde{\mathbf{y}}_n\}_{n=0}^{\nu}$ uniquely up to scaling.

Proof. The second part is covered by our derivation. It remains to prove the first part by induction. Assume that the sequences $\{\mathbf{y}_n\}_{n=0}^{\nu-1}$ and $\{\widetilde{\mathbf{y}}_m\}_{m=0}^{\nu-1}$ have been generated by the BIO algorithm and that (2.1) and (2.2) hold for $m, n = 0, \ldots, k$ $(< \nu)$. (For $k = 0$ this is clearly the case.) Then the validity of (2.1) for $m = k + 1$ or $n = k + 1$ is obvious from (2.21e) and (2.21f). Moreover, by (2.21j), (2.21e), and (2.21f),

$$\langle \widetilde{\mathbf{y}}_m, \mathbf{y}_{k+1} \rangle_{\mathbf{B}} = \langle \widetilde{\mathbf{y}}_m, \mathbf{A}\mathbf{y}_k - \mathbf{y}_k\alpha_k - \mathbf{y}_{k-1}\beta_{k-1} \rangle_{\mathbf{B}} \, / \gamma_k \tag{2.22a}$$

$$= \Big(\langle \widetilde{\gamma}_m \widetilde{\mathbf{y}}_{m+1} + \widetilde{\alpha}_m \widetilde{\mathbf{y}}_m + \widetilde{\beta}_{m-1} \widetilde{\mathbf{y}}_{m-1}, \mathbf{y}_k \rangle_{\mathbf{B}} - \langle \widetilde{\mathbf{y}}_m, \mathbf{y}_k\alpha_k \rangle_{\mathbf{B}} - \langle \widetilde{\mathbf{y}}_m, \mathbf{y}_{k-1}\beta_{k-1} \rangle_{\mathbf{B}} \Big) / \gamma_k. \tag{2.22b}$$

If $m \leq k - 2$, all terms on the right-hand side of (2.22b) vanish. For $m = k - 1$, we use (2.20) and (2.2) to obtain on the right-hand side of (2.22a) $(\beta_{k-1}\delta_{k-1} - 0 - \beta_{k-1}\delta_{k-1})/\gamma_k = 0$. Next, using (2.21a) and (2.2) we get for $m = k$ in (2.22a) $(\alpha_k\delta_k - \alpha_k\delta_k)/\gamma_k = 0$. Analogous arguments yield $\langle \widetilde{\mathbf{y}}_{k+1}, \mathbf{y}_n \rangle_{\mathbf{B}} = 0$ for $n = 0, \ldots, k$. Finally, by (2.21g)–(2.21j), $\langle \widetilde{\mathbf{y}}_{k+1}, \mathbf{y}_{k+1} \rangle_{\mathbf{B}} = \delta_k$. This completes the induction. □

In summary, the Lanczos process generates a pair of biorthogonal bases of a pair of Krylov spaces and does this with a pair of three-term recurrences that are closely linked to each other. In each step only two pairs of orthogonality conditions need to be enforced, which, due to the linkage, reduce to just one pair and thus require only two inner products in total. All the other orthogonality conditions are inherited: they are satisfied automatically, at least in theory. Of course, if we want normalized basis vectors, we need to invest another two inner products per step. Clearly, we also need two matrix-vector products per step to expand the two Krylov spaces.

2.2. Matrix relations

The matrix relations (2.9)–(2.11) can be considered as a shorthand notation for the BIO algorithm: the relations (2.10) describe the recurrences for $\mathbf{y}_n$

and $\tilde{\mathbf{y}}_n$, while (2.9) and (2.11) summarize the formulae for determining the elements of $\mathbf{T}_\nu$, $\widetilde{\mathbf{T}}_\nu$, and $\mathbf{D}_{\delta;\nu}$. These matrix relations hold as well after $n < \nu$ steps, and in fact are obtained for such an intermediate stage by considering submatrices of the appropriate size.

Equation (2.10) can be simplified by introducing the $(n+1) \times n$ leading principal submatrices of $\mathbf{T}_\nu$ and $\widetilde{\mathbf{T}}_\nu$, the *extended* tridiagonal matrices[5]

$$\underline{\mathbf{T}}_n := \left[\begin{array}{c} \mathbf{T}_n \\ \hline \gamma_{n-1}\mathbf{l}_n^\top \end{array}\right] = \begin{bmatrix} \alpha_0 & \beta_0 & & & \\ \gamma_0 & \alpha_1 & \beta_1 & & \\ & \gamma_1 & \alpha_2 & \ddots & \\ & & \ddots & \ddots & \beta_{n-2} \\ & & & \gamma_{n-2} & \alpha_{n-1} \\ & & & & \gamma_{n-1} \end{bmatrix},$$

and the analogous matrices with tildes. Then we have altogether

$$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n, \quad \mathbf{A}^\star\widetilde{\mathbf{Y}}_n = \widetilde{\mathbf{Y}}_{n+1}\underline{\widetilde{\mathbf{T}}}_n \quad (n \le \nu), \tag{2.23}$$

$$\widetilde{\mathbf{Y}}_n^\star\mathbf{B}\mathbf{Y}_n = \mathbf{D}_{\delta;n}, \quad \widetilde{\mathbf{Y}}_n^\star\mathbf{B}\mathbf{A}\mathbf{Y}_n = \mathbf{D}_{\delta;n}\mathbf{T}_n \quad (n \le \nu), \tag{2.24}$$

$$\mathbf{D}_{\delta;n}\mathbf{T}_n = \widetilde{\mathbf{T}}_n^\star\mathbf{D}_{\delta;n} \quad (n \le \nu). \tag{2.25}$$

The first relation in (2.23) means that $\underline{\mathbf{T}}_n$ is the representation in the basis $\{\mathbf{y}_0, \ldots, \mathbf{y}_n\}$ of the restriction of $\mathbf{A}$ to the subspace $\mathcal{K}_n$, which is mapped into $\mathcal{K}_{n+1}$. The subspace $\mathcal{K}_n$ is 'nearly invariant' as its image requires only one additional space dimension. When it turns out that the component of $\mathbf{A}\mathbf{y}_{n-1}$ in this direction is relatively short, then we can expect that the eigenvalues of $\mathbf{T}_n$ are close approximations of eigenvalues of $\mathbf{A}$. This is vaguely the reasoning for applying the BIO algorithm to eigenvalue computations. There are several reasons that make this argument dangerous. First, the basis is not orthogonal, and thus $\mathbf{T}_n$ is linked to $\mathbf{A}$ by an oblique projection only: while in the Hermitian case the eigenvalues of $\mathbf{T}_n$ are *Ritz values*, that is, *Galerkin* or *Rayleigh–Ritz approximations*, they are in the non-Hermitian case only *Petrov–Galerkin* approximations, which are sometimes referred to as *Petrov values*. Second, since neither $\mathbf{A}$ nor $\mathbf{T}_n$ are Hermitian in the case considered here, eigenvalues need not behave nicely under perturbation.

A notorious problem with the Lanczos process is that in finite precision arithmetic round-off affects it strongly. In particular, since only two orthogonality conditions are enforced at every step, while orthogonality with respect to earlier vectors is inherited, a loss of (bi)orthogonality is noticed

[5] By underlining $\mathbf{T}_n$ we want to indicate that we augment this matrix by an additional row. We suggest reading $\underline{\mathbf{T}}_n$ as 'T sub n extended'. The same notation will be used on other occasions.

in practice, which means that $\mathbf{D}_{\delta;n}$ is not diagonal. This is also a serious problem in the Hermitian case; we will return to it briefly in Section 18.

2.3. Normalization; simplification due to symmetry

As discussed before, in the BIO algorithm (Algorithm 1) the coefficients α_n, $\widetilde{\alpha}_n$, β_{n-1}, $\widetilde{\beta}_{n-1}$, γ_n, $\widetilde{\gamma}_n$, and δ_{n+1} are defined uniquely except that in (2.21i) we are free to choose two of the three quantities γ_n, $\widetilde{\gamma}_n$, and δ_{n+1}. Special versions of the algorithm are found by making a particular choice for two of the coefficients γ_n, $\widetilde{\gamma}_n$, and δ_{n+1}, and capitalizing upon the particular choice. The classical choices for theoretical work are $\gamma_n := \widetilde{\gamma}_n := 1$ (Lanczos 1950, Lanczos 1952, Rutishauser 1953, Householder 1964) or $\widetilde{\gamma}_n = \overline{\gamma_n}$ and $\delta_{n+1} := 1$. The latter makes the two vector sequences biorthonormal and yields $\beta_{n-1} = \gamma_{n-1}$; that is, $\mathbf{T}_\nu = \mathbf{T}_\nu^\top$ is real or complex symmetric, *cf.* (2.18). (Consequently, (2.16) and (2.17) then coincide.) For numerical computations the former choice is risky with respect to overflow, and the latter is inappropriate for real nonsymmetric matrices since it may lead to some complex γ_n. Therefore, in the real nonsymmetric case, the two choices

$$\gamma_n := \widetilde{\gamma}_n := \sqrt{|\delta_{\text{temp}}|}\,, \quad \delta_{n+1} := \delta_{\text{temp}}/(\gamma_n\widetilde{\gamma}_n) = \delta_{\text{temp}}/|\delta_{\text{temp}}|, \tag{2.26}$$

$$\gamma_n := \|\mathbf{y}_{\text{temp}}\|\,, \quad \widetilde{\gamma}_n := \|\widetilde{\mathbf{y}}_{\text{temp}}\|\,, \quad \delta_{n+1} := \delta_{\text{temp}}/(\gamma_n\widetilde{\gamma}_n), \tag{2.27}$$

are normally suggested, but there may be a special reason for yet another one. Note that (2.27) requires two additional inner products. These are often justified anyway by the necessity of stability and round-off error control; see Section 18.

Replacing, say, $\gamma_n = 1$ (for all n) by some other choice means replacing $\mathbf{y}_n$ by

$$\widehat{\mathbf{y}}_n := \mathbf{y}_n/\Gamma_n, \quad \text{where } \Gamma_n := \gamma_0\gamma_1\cdots\gamma_{n-1}, \tag{2.28}$$

which in view of (2.23) amounts to replacing $\mathbf{T}_n$ by

$$\widehat{\mathbf{T}}_n := \mathbf{D}_{\Gamma;n}^{-1}\mathbf{T}_n\mathbf{D}_{\Gamma;n}, \quad \text{where } \mathbf{D}_{\Gamma;n} := \operatorname{diag}(\Gamma_0, \Gamma_1, \ldots, \Gamma_{n-1}), \tag{2.29}$$

If γ_n and $\widetilde{\gamma}_n$ are chosen independently of $\mathbf{y}_{\text{temp}}$ and $\widetilde{\mathbf{y}}_{\text{temp}}$, the formulae (2.21e)–(2.21j) can be simplified; *cf.* Algorithm 2 below.

If $\mathbf{A}$ and $\mathbf{B}$ are Hermitian and $\mathbf{B}$ is positive definite, starting with $\widetilde{\mathbf{y}}_0 = \mathbf{y}_0$ and making the natural choice $\widetilde{\gamma}_n := \overline{\gamma_n}$ and $\delta_n > 0$ leads to $\widetilde{\mathbf{T}}_\nu = \overline{\mathbf{T}_\nu} = \mathbf{T}_\nu$ (that is, $\widetilde{\mathbf{T}}_\nu = \mathbf{T}_\nu$ is real) and $\widetilde{\mathbf{y}}_n = \mathbf{y}_n$ (for all n). Thus, the recursion (2.21f) for $\widetilde{\mathbf{y}}_n$ is redundant, the costs are reduced to roughly half, and the Lanczos vectors are orthogonal to each other. Moreover, one can choose $\gamma_n > 0$ (for all n), which then implies that $\beta_{n-1} > 0$ also. Finally, choosing $\delta_n := \delta_0$ (for all n) makes $\mathbf{T}_\nu$ real symmetric. Then the BIO algorithm becomes the symmetric Lanczos algorithm, which is often just called the *Lanczos algorithm.*

If $\mathbf{A}$ is complex symmetric and $\widetilde{\mathbf{y}}_0 = \overline{\mathbf{y}_0}$, then $\widetilde{\mathbf{y}}_n = \overline{\mathbf{y}_n}$ (for all n). Again, the costs reduce to about half. Now, setting $\delta_n := 1$ (for all n) makes $\mathbf{T}_\nu$ complex symmetric, but this can be achieved in general and has nothing to do with $\mathbf{A}$ being complex symmetric. See Freund (1992) for further details on this case.

In Section 6 we will discuss yet other cases where the BIO algorithm simplifies.

3. Termination, breakdowns and convergence

The BIO algorithm *stops* with $\nu := n+1$ when $\delta_{\text{temp}} = 0$, since α_{n+1} would become infinite or indefinite. We call ν here the *index of first breakdown or termination*. Of course, ν is bounded by the maximum dimension of the Krylov spaces, but ν may be smaller for several reasons. The maximum dimension of the subspaces $\mathcal{K}_n$ defined by (2.1) depends not only on $\mathbf{A}$ but also on $\mathbf{y}_0$; it is called the *grade of* $\mathbf{y}_0$ *with respect to* $\mathbf{A}$ and is here denoted by $\bar{\nu}(\mathbf{y}_0, \mathbf{A})$. As is easy to prove, it satisfies

$$\bar{\nu}(\mathbf{y}_0, \mathbf{A}) = \min \{n : \dim \mathcal{K}_n = \dim \mathcal{K}_{n+1}\},$$

and it is at most equal to the degree $\bar{\nu}(\mathbf{A})$ of the minimum polynomial of $\mathbf{A}$. Clearly, the Lanczos process stops with $\mathbf{y}_{\text{temp}} = \widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}$ when $\nu = \bar{\nu}(\mathbf{A})$. If this *full termination* due to $\mathbf{y}_{\text{temp}} = \widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}$ happens before the degree of the minimum polynomial is reached, that is, if $\nu < \bar{\nu}(\mathbf{A})$, we call it an *early full termination*. However, the BIO algorithm can also stop with either $\mathbf{y}_{\text{temp}} = \mathbf{o}$ or $\widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}$, and even with $\langle \widetilde{\mathbf{y}}_{\text{temp}}, \mathbf{y}_{\text{temp}} \rangle_{\mathbf{B}} = 0$ when $\mathbf{y}_{\text{temp}} \neq \mathbf{o}$ and $\widetilde{\mathbf{y}}_{\text{temp}} \neq \mathbf{o}$. Then we say that it *breaks down*, or, more exactly, that we have a *one-sided termination* or a *serious breakdown*, respectively[6].

Lanczos (1950, 1952) was already aware of these various breakdowns. They have since been discussed by many authors; see, in particular, Faddeev and Faddeeva (1964), Gutknecht (1990), Householder (1964), Joubert (1992), Parlett (1992), Parlett et al. (1985), Rutishauser (1953), Saad (1982), and Taylor (1982).

Of course, in floating-point arithmetic, a *near-breakdown* passed without taking special measures may lead to stability problems. Therefore, in practice, breakdown conditions '$= \mathbf{o}$' and '$= 0$' have to be replaced by other conditions that should not depend on scaling and should prevent us from numerical instability. We will return to this question in Section 19.

[6] Parlett (1992) refers to a *benign breakdown* when we have either an early full termination or a one-sided termination.

3.1. Full termination

In the case of *full termination*, that is, when the BIO algorithm terminates with

$$\mathbf{y}_{\text{temp}} = \widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}, \tag{3.1}$$

we can conclude from (2.10) that

$$\mathbf{A}\mathbf{Y}_\nu = \mathbf{Y}_\nu \mathbf{T}_\nu, \quad \mathbf{A}^\star \widetilde{\mathbf{Y}}_\nu = \widetilde{\mathbf{Y}}_\nu \widetilde{\mathbf{T}}_\nu. \tag{3.2}$$

This means that $\mathcal{K}_\nu$ and $\widetilde{\mathcal{K}}_\nu$ are invariant subspaces of dimension ν of $\mathbf{A}$ and $\mathbf{A}^\star$, respectively. The set of eigenvalues of $\mathbf{T}_\nu$ is then a subset of the spectrum of $\mathbf{A}$.

The formula $\mathbf{A}\mathbf{Y}_\nu = \mathbf{Y}_\nu \mathbf{T}_\nu$ points to an often cited objective of the BIO algorithm: the similarity reduction of a given matrix $\mathbf{A}$ to a tridiagonal matrix $\mathbf{T}_\nu$. For the latter the computation of the eigenvalues is much less costly, in particular in the Hermitian case. However, unless $\nu = N$ or the Lanczos process is restarted (possibly several times) with some new pair $\widetilde{\mathbf{y}}_\nu$, $\mathbf{y}_\nu$ satisfying (2.4), it can never determine the geometric multiplicity of an eigenvalue, since it can at best find the factors of the minimal polynomial of $\mathbf{A}$. In theory, to find the whole spectrum, we could continue the algorithm after a full termination with $\nu < N$ by constructing first a new pair $(\mathbf{y}_\nu, \widetilde{\mathbf{y}}_\nu)$ of nonzero vectors that are biorthogonal to the pair of vector sequences constructed so far, *i.e.*, satisfy (2.4); see, for instance, Householder (1964), Lanczos (1950), Lanczos (1952), and Rutishauser (1953). Starting from a trial pair $(\mathbf{y}, \widetilde{\mathbf{y}})$ one would have to construct

$$\mathbf{y}_\nu := \mathbf{y} - \sum_{k=0}^{\nu-1} \mathbf{y}_k \langle \widetilde{\mathbf{y}}_k, \mathbf{y} \rangle_{\mathbf{B}} / \delta_k, \tag{3.3a}$$

$$\widetilde{\mathbf{y}}_\nu := \widetilde{\mathbf{y}} - \sum_{k=0}^{\nu-1} \widetilde{\mathbf{y}}_k \langle \mathbf{y}_k, \widetilde{\mathbf{y}} \rangle_{\mathbf{B}^*} / \overline{\delta_k}, \tag{3.3b}$$

and hope that the two resulting vectors are nonzero. Then, one can set $\gamma_{\nu-1} := \beta_{\nu-1} := 0$, so that after the restart the relations (2.10) hold even beyond this ν. If no breakdown or one-sided termination later occurs, and if any further early full termination is also followed by such a restart, then in theory the algorithm must terminate with $\mathbf{y}_{\text{temp}} = \widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}$ and $\nu = N$. The relations (3.2) then hold with all the matrices being square of order N. The tridiagonal matrix $\mathbf{T}_N$ may have some elements $\gamma_k = \beta_k = 0$ due to the restarts, and the same will then happen to $\widetilde{\mathbf{T}}_N$. Since $\mathbf{Y}_N$ and $\widetilde{\mathbf{Y}}_N$ are nonsingular, $\mathbf{T}_N$ is similar to $\mathbf{A}$, and $\widetilde{\mathbf{T}}_N$ is similar to $\mathbf{A}^\star$. Unfortunately, in practice this all works only for very small N: first, as is seen from (3.3a)–(3.3b), the restart with persisting biorthogonality (2.2) requires all previously computed vectors of the two sequences, but these are

normally not stored; second, completely avoiding loss of (bi)orthogonality during the iteration would require full reorthogonalization with respect to previous Lanczos vectors, which means giving up the greatest advantages of the Lanczos process, the short recurrences.

For the same reasons, finding all the roots of the minimal polynomial with the Lanczos process is in practice normally beyond reach: due to loss of (bi)orthogonality or because $\bar{\nu}(\mathbf{A})$ is too large and the process has to be stopped early, only few of the eigenvalues are found with acceptable accuracy; fortunately, these are often those that are of prime interest.

3.2. One-sided termination

When the BIO algorithm stops due to

$$\text{either } \mathbf{y}_{\text{temp}} = \mathbf{o} \text{ or } \widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}$$

(but not $\mathbf{y}_{\text{temp}} = \widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}$), we call this a *one-sided termination*. In some applications this is welcome, but in others it may still be a serious difficulty. In view of (2.10), it means that either $\mathcal{K}_\nu$ or $\widetilde{\mathcal{K}}_\nu$ is an invariant subspace of dimension ν of $\mathbf{A}$ or $\mathbf{A}^\star$, respectively. For eigenvalue computations, this is very useful information, although sometimes one may need to continue the algorithm with a pair $(\mathbf{y}_\nu, \widetilde{\mathbf{y}}_\nu)$ that is biorthogonal to the pairs constructed so far, in order to find further eigenvalues. Determining the missing vector of this pair will again require the expensive orthogonalization of a trial vector with respect to a ν-dimensional Krylov subspace, that is, either (3.3a) or (3.3b).

In contrast, when we have to solve a linear system, then $\mathbf{y}_{\text{temp}} = \mathbf{o}$ is all we aim at, as we will see in the next section. Unfortunately, when $\widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}$ but $\mathbf{y}_{\text{temp}} \neq \mathbf{o}$, we have a nasty situation where we have to find a replacement for $\widetilde{\mathbf{y}}_{\text{temp}}$ that is orthogonal to $\mathcal{K}_\nu$. In practice, codes either just use some $\widetilde{\mathbf{y}}_{\text{temp}}$ that consists of scaled up round-off errors or restart the BIO algorithm. In either case the convergence slows down. The best precaution against this type of breakdown seems to be choosing as left initial vector $\widetilde{\mathbf{y}}_0$ a random one.

3.3. Serious breakdowns

Let us now discuss the *serious breakdowns* of the BIO algorithm, which we also call *Lanczos breakdowns* to distinguish them from a second type of serious breakdown that can occur additionally in the standard form of the biconjugate gradient method. Hence, assume that the BIO algorithm stops due to

$$\delta_{\text{temp}} = 0, \quad \text{but neither } \mathbf{y}_{\text{temp}} = \mathbf{o} \text{ nor } \widetilde{\mathbf{y}}_{\text{temp}} = \mathbf{o}.$$

In the past, the recommendation was to restart the algorithm from scratch. Nowadays one should implement what is called *look-ahead*. It is the curing of these breakdowns and the corresponding near-breakdowns that is addressed by the *look-ahead Lanczos algorithms* which have attracted so much attention recently. We will discuss the look-ahead BIO algorithm and some of the related theory in Section 19, where we will also give detailed references. In most cases, look-ahead is successful. Oversimplifying matters, we can say that curing a serious breakdown with look-ahead requires that $\langle \widetilde{\mathbf{y}}_\nu, \mathbf{A}^k \mathbf{y}_\nu \rangle_{\mathbf{B}} \neq 0$ for some k, while the breakdown is *incurable* when

$$\langle \widetilde{\mathbf{y}}_\nu, \mathbf{A}^k \mathbf{y}_\nu \rangle_{\mathbf{B}} = 0 \quad (\text{for all } k \geq 0).$$

In theory, the condition $\langle \widetilde{\mathbf{y}}_\nu, \mathbf{A}^k \mathbf{y}_\nu \rangle_{\mathbf{B}} \neq 0$ has to be replaced by a positive lower bound for the smallest singular value of a $k \times k$ matrix, but in practice even this condition is not safe.

The serious breakdown does not occur if $\mathbf{A}$ and $\mathbf{B}$ are Hermitian, $\mathbf{B}$ is positive definite, $\widetilde{\mathbf{y}}_0 = \mathbf{y}_0$, $\widetilde{\gamma}_n = \overline{\gamma_n}$, and $\delta_n > 0$ (for all n), since then $\widetilde{\mathbf{y}}_n = \mathbf{y}_n$ (for all n), and $\langle ., . \rangle_{\mathbf{B}}$ is an inner product. Also, choosing γ_n, $\widetilde{\gamma}_n$, or δ_n differently will not destroy this property. On the other hand, serious breakdowns can still occur for a real symmetric matrix $\mathbf{A}$ if $\widetilde{\mathbf{y}}_0 \neq \mathbf{y}_0$.

Under the standard assumption $\mathbf{B} = \mathbf{I}$ it was shown by Rutishauser (1953) (for another proof see Householder (1964)) that there exist $\mathbf{y}_0$ and $\widetilde{\mathbf{y}}_0$ such that neither a serious breakdown nor a premature termination occurs; that is, such that the process does not end before the degree of the minimal polynomial is attained. Unfortunately, such a pair $(\mathbf{y}_0, \widetilde{\mathbf{y}}_0)$ is in general not known. Joubert (1992) even showed that, in a probabilistic sense, nearly all pairs have this property; that is, the assumption of having neither a premature termination nor a breakdown is a generic property. This is no longer true if the matrix is real and one restricts the initial vectors by requiring $\widetilde{\mathbf{y}}_0 = \mathbf{y}_0 \in \mathbb{R}^N$. Joubert gives an example where serious breakdowns then occur for almost all $\mathbf{y}_0$. Of course, the set of pairs $(\mathbf{y}_0, \widetilde{\mathbf{y}}_0)$ that lead to a near-breakdown never has measure zero. But practice shows that near-breakdowns that have a devastating effect on the process are fairly rare.

3.4. Convergence

Although in theory the BIO algorithm either terminates or breaks down in at most $\min\{\bar{\nu}(\mathbf{y}_0, \mathbf{A}), \bar{\nu}(\widetilde{\mathbf{y}}_0, \mathbf{A}^\star)\}$ steps, this rarely happens in practice, and the process can be continued far beyond N. For small matrices this is sometimes necessary when one wants to find all eigenvalues or to solve a linear system. But the BIO algorithm is usually applied to very large matrices and stopped at some $n \ll N$, so that only (2.10) and (2.11) hold, but not (3.2). The n eigenvalues of $\mathbf{T}_n$ are then considered as approximations of n eigenvalues of $\mathbf{A}$, and typically they tend to approximate eigenvalues that lie near the

border of the spectrum. For eigenvalues of small absolute value, the absolute error is comparable to the one for large eigenvalues; hence, the relative error tends to be large. Therefore, the method is best at finding the dominant eigenvalues. Lanczos (1950, p. 270) found a heuristic explanation for this important phenomenon. Some remarks and references on the convergence of Lanczos-type solvers will be made in the next section when we discuss the basic properties of the BiCG method.

An additional difficulty is that due to the loss of biorthogonality, eigenvalues of $\mathbf{A}$ may reappear in $\mathbf{T}_n$ with too high multiplicity. One refers to these extra eigenvalues as *ghost eigenvalues*. We will come back to this problem in Section 18.

4. The BIORES form of the BICG method

Let us now turn to applying the Lanczos BIO algorithm to the problem of solving linear systems of equations $\mathbf{Ax} = \mathbf{b}$. We first review the basic properties of the conjugate gradient (CG) and the conjugate residual (CR) methods and then describe a first version, BIORES, of the biconjugate gradient (BiCG) method. In the next section we will further cover the MINRES algorithm for the CR method and a first version of the QMR method.

We assume that $\mathbf{A}$ is nonsingular and denote the solution of $\mathbf{Ax} = \mathbf{b}$ by $\mathbf{x}_{\text{ex}}$, its initial approximation by $\mathbf{x}_0$, the nth approximation (or *iterate*) by $\mathbf{x}_n$, and the corresponding *residual* by $\mathbf{r}_n := \mathbf{b} - \mathbf{Ax}_n$. Additionally, we let $\mathbf{y}_0 := \mathbf{r}_0$, or $\mathbf{y}_0 := \mathbf{r}_0/||\mathbf{r}_0||$ if we aim for normalized Lanczos vectors. As in (2.1), $\mathcal{K}_n$ is the nth Krylov space generated by $\mathbf{A}$ from $\mathbf{y}_0$, which now has the direction of the initial residual. From time to time we also refer to the nth *error*, $\mathbf{x}_{\text{ex}} - \mathbf{x}_n$. Note that $\mathbf{r}_n = \mathbf{A}(\mathbf{x}_{\text{ex}} - \mathbf{x}_n)$.

4.1. The conjugate gradient and conjugate residual methods

We first assume that $\mathbf{A}$ and $\mathbf{B}$ are commuting Hermitian positive definite matrices and recall some facts about the *conjugate gradient* (CG) *method* of Hestenes and Stiefel (1952). It is characterized by the property that the nth iterate $\mathbf{x}_n$ minimizes the quadratic function

$$\mathbf{x} \mapsto \langle(\mathbf{x}_{\text{ex}} - \mathbf{x}), \mathbf{A}(\mathbf{x}_{\text{ex}} - \mathbf{x})\rangle_{\mathbf{B}}$$

among all $\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_n$. The standard case is again $\mathbf{B} = \mathbf{I}$. By differentiation one readily verifies that the minimization problem is equivalent to the *Galerkin condition*

$$\langle \mathbf{y}, \mathbf{r}_n\rangle_{\mathbf{B}} = 0 \ \text{(for all } \mathbf{y} \in \mathcal{K}_n), \quad \textit{i.e.}, \quad \mathcal{K}_n \perp_{\mathbf{B}} \mathbf{r}_n. \tag{4.1}$$

Note that

$$\mathbf{r}_n = \mathbf{b} - \mathbf{Ax}_0 + \mathbf{A}(\mathbf{x}_0 - \mathbf{x}_n) \in \mathbf{r}_0 + \mathbf{A}\mathcal{K}_n \subset \mathcal{K}_{n+1}. \tag{4.2}$$

In view of (4.1) and (4.2), $\mathbf{r}_n$ spans the orthogonal complement of $\mathcal{K}_n$ in $\mathcal{K}_{n+1}$. Hence, if we let $\mathbf{y}_0 := \mathbf{r}_0$ and generate the basis $\{\mathbf{y}_k\}_{k=0}^n$ of $\mathcal{K}_{n+1}$ by recursively orthogonalizing $\mathbf{A}\mathbf{y}_m$ with respect to $\mathcal{K}_{m+1}$ $(m = 0, \ldots, n-1)$, then $\mathbf{y}_n$ is proportional to $\mathbf{r}_n$, and by suitable normalization we can achieve $\mathbf{y}_n = \mathbf{r}_n$. This recursive orthogonalization procedure is none other than the symmetric Lanczos process, that is, the BIO algorithm with $\tilde{\mathbf{y}}_0 = \mathbf{y}_0$ and commuting Hermitian matrices $\mathbf{A}$ and $\mathbf{B}$. Note also that by (4.2), $\mathbf{r}_n = p_n(\mathbf{A})\mathbf{y}_0$, where p_n is a polynomial of exact degree n satisfying $p_n(0) = 1$. This property, which is equivalent to $\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K}_n$, means that CG is a Krylov space solver, as defined by (1.1). Of course, up to normalization, the *residual polynomial* p_n is here the Lanczos polynomial of (2.3).

The directions $\mathbf{x}_n - \mathbf{x}_{n-1}$ can be seen to be conjugate to each other (*i.e.*, $\mathbf{A}$-orthogonal with respect to the $\mathbf{B}$-inner product, or, simply, $\mathbf{AB}$-orthogonal), whence CG is a special *conjugate direction* method. In their classical CG method Hestenes and Stiefel (1952) chose $\mathbf{B} = \mathbf{I}$ and thus minimized $\mathbf{x} \mapsto \langle(\mathbf{x}_{\text{ex}} - \mathbf{x}), \mathbf{A}(\mathbf{x}_{\text{ex}} - \mathbf{x})\rangle$, which is the square of the $\mathbf{A}$-norm of the error. (This is a norm since $\mathbf{A}$ is Hermitian positive definite.) With respect to the inner product induced by $\mathbf{A}$, we then have from (4.1) and (4.2)

$$\mathcal{K}_n \perp_{\mathbf{A}} (\mathbf{x}_{\text{ex}} - \mathbf{x}_n), \quad (\mathbf{x}_{\text{ex}} - \mathbf{x}_n) - (\mathbf{x}_{\text{ex}} - \mathbf{x}_0) \in \mathcal{K}_n.$$

This means that $\mathbf{x}_0 - \mathbf{x}_n = (\mathbf{x}_{\text{ex}} - \mathbf{x}_n) - (\mathbf{x}_{\text{ex}} - \mathbf{x}_0)$ is the $\mathbf{A}$-orthogonal projection of the initial error $\mathbf{x}_{\text{ex}} - \mathbf{x}_0$ into $\mathcal{K}_n$, and the error $\mathbf{x}_{\text{ex}} - \mathbf{x}_n$ is the difference between the initial error and its projection. Therefore, the CG method can also be viewed as an *orthogonal projection method* in the error space endowed with the $\mathbf{A}$-norm. Moreover, it can be understood as an orthogonal projection method in the residual space endowed with the $\mathbf{A}^{-1}$-norm.

If $\mathbf{B} = \mathbf{A} = \mathbf{A}^\star$ instead, we have

$$\langle(\mathbf{x}_{\text{ex}} - \mathbf{x}), \mathbf{A}(\mathbf{x}_{\text{ex}} - \mathbf{x})\rangle_{\mathbf{B}} = ||\mathbf{A}(\mathbf{x}_{\text{ex}} - \mathbf{x})||^2 = ||\mathbf{b} - \mathbf{A}\mathbf{x}||^2, \tag{4.3}$$

which shows that the residual norm is now minimized. The $\mathbf{B}$-orthogonality of the residuals means here that they are conjugate. The method is therefore called *conjugate residual* (CR) or *minimum residual* method. Normally, the abbreviation MINRES stands for a particular algorithm due to Paige and Saunders (1975) for this method. We will come back to it in Section 5. Like some other versions of the CR method, MINRES is also applicable to Hermitian indefinite systems; see Ashby, Manteuffel and Saylor (1990), Fletcher (1976).

From (4.1) and (4.2) we can conclude here that $\mathbf{r}_n$ is chosen so that it is orthogonal to $\mathbf{A}\mathcal{K}_n$ and $\mathbf{r}_0 - \mathbf{r}_n$ lies in $\mathbf{A}\mathcal{K}_n$. In other words, with respect to the standard inner product in Euclidean space, $\mathbf{r}_0 - \mathbf{r}_n$ is the orthogonal projection of $\mathbf{r}_0$ onto $\mathbf{A}\mathcal{K}_n$. Therefore, the CR method is an orthogonal projection method in the residual space.

From the fact that the CG-residuals are mutually orthogonal and the CR-residuals are mutually conjugate, it follows in particular that in both cases $\mathbf{r}_\nu = \mathbf{o}$ for some $\nu \leq N$, and thus $\mathbf{x}_\nu = \mathbf{x}_{\text{ex}}$. However, in practice this *finite termination property* is fairly irrelevant, as it is severely spoiled by round-off.

So far we only know how to construct the residuals, but we still need another recurrence for the iterates $\mathbf{x}_n$ themselves. As we will see in a moment, such a recurrence is found by multiplying by $\mathbf{A}^{-1}$ the one for the residuals, that is, the one for the appropriately scaled right Lanczos vectors. This then leads to the three-term version[7], ORES, of the conjugate gradient method (Hestenes 1951). The standard version of the CG method, OMIN, instead uses coupled two-term recurrences also involving the direction vectors, which are multiples of the corrections $\mathbf{x}_{n+1} - \mathbf{x}_n$. In the rest of this section and in Section 8 we want to describe generalizations to the nonsymmetric case.

4.2. Basic properties of the BICG *method*

If $\mathbf{A}$ is non-Hermitian, the construction of an orthogonal basis $\{\mathbf{y}_n\}$ of the Krylov space becomes expensive and memory-intensive, since the recurrences for $\mathbf{y}_n$ generally involve all previous vectors (as first assumed in (2.7)). Therefore, the resulting *Arnoldi* or *full orthogonalization method* (FOM) (Arnoldi 1951, Saad 1981) has to be either restarted periodically or truncated, which means that some of the information that was built up is lost. The same applies to the *generalized conjugate residual* (GCR) *method* (Eisenstat, Elman and Schultz 1983) and its special form, the GMRES algorithm of Saad and Schultz (1986), which extends the MINRES algorithm to the nonsymmetric case.

However, we know how to construct efficiently a pair of biorthogonal sequences $\{\mathbf{y}_n\}$, $\{\tilde{\mathbf{y}}_n\}$, namely by the BIO algorithm of Section 2. By requiring that iterates $\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K}_n$ satisfy the *Petrov–Galerkin condition*

$$\langle \tilde{\mathbf{y}}, \mathbf{r}_n \rangle_{\mathbf{B}} = 0 \ \text{(for all } \tilde{\mathbf{y}} \in \tilde{\mathcal{K}}_n), \quad \textit{i.e.}, \quad \tilde{\mathcal{K}}_n \perp_{\mathbf{B}} \mathbf{r}_n, \tag{4.4}$$

we find the *biconjugate gradient* (BICG) *method.* In contrast to the Galerkin condition (4.1) of the CG method, this one does not belong to a minimization problem in a fixed norm[8].

[7] The acronyms ORES, OMIN, and ODIR were introduced in Ashby et al. (1990) as abbreviations for ORTHORES, ORTHOMIN, and ORTHODIR. We suggest using the short form whenever the basic recurrences of the method are short. Note that our acronyms BIORES, BIOMIN, and BIODIR for the various forms of the BICG method fit into this pattern, as these algorithms also feature short recurrences.

[8] Barth and Manteuffel (1994) showed that BICG and QMR fit into the framework of *variable metric methods*: in exact arithmetic, if the methods do not break down or terminate with $\nu < N$, then the iterates that have been created minimize the error in a

Now $\mathbf{r}_n$ is chosen so that it is orthogonal to $\widetilde{\mathcal{K}}_n$ and so that $\mathbf{r}_0 - \mathbf{r}_n$ lies in $\mathbf{A}\mathcal{K}_n$. This means that $\mathbf{r}_0 - \mathbf{r}_n$ is the projection of $\mathbf{r}_0$ onto $\mathbf{A}\mathcal{K}_n$ along a direction that is orthogonal to another space, namely $\widetilde{\mathcal{K}}_n$, and, hence, is in general oblique with respect to the projection space $\mathbf{A}\mathcal{K}_n$. Therefore, the BiCG method is said to be an *oblique projection method* (Saad 1982, Saad 1996).

Since both the residual $\mathbf{r}_n$ and the right Lanczos vector $\mathbf{y}_n$ satisfy (4.4) and both lie in $\mathcal{K}_{n+1}$, and since we have seen in Section 2 that $\mathbf{y}_n$ is determined up to a scalar factor by these conditions, we can again conclude that the residual must be a scalar multiple of the Lanczos vector and that by appropriate normalization of the latter we could attain $\mathbf{r}_n = \mathbf{y}_n$.

The most straightforward way of taking into account the two conditions $\mathbf{x}_n \in \mathbf{x}_0 + \mathcal{K}_n$ and $\widetilde{\mathcal{K}}_n \perp_{\mathbf{B}} \mathbf{r}_n$ is the following one. Representing $\mathbf{x}_n - \mathbf{x}_0$ in terms of the Lanczos vectors we can write

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Y}_n \mathbf{k}_n, \quad \mathbf{r}_n = \mathbf{r}_0 - \mathbf{A}\mathbf{Y}_n \mathbf{k}_n, \tag{4.5}$$

with some coordinate vector $\mathbf{k}_n$. Using $\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n$, see (2.23), and

$$\mathbf{r}_0 = \mathbf{y}_0 \rho_0 = \mathbf{Y}_{n+1} \underline{\mathbf{e}}_1 \rho_0,$$

with $\underline{\mathbf{e}}_1 := [\ 1 \quad 0 \quad 0 \quad \cdots\ ]^\top \in \mathbb{R}^{n+1}$ and $\rho_0 := ||\mathbf{r}_0||$ (assuming $||\mathbf{y}_0|| = 1$ here), we find that

$$\mathbf{r}_n = \mathbf{Y}_{n+1} \left(\underline{\mathbf{e}}_1 \rho_0 - \underline{\mathbf{T}}_n \mathbf{k}_n \right). \tag{4.6}$$

In view of $\widetilde{\mathbf{Y}}_n^\star \mathbf{B} \mathbf{Y}_{n+1} = [\ \mathbf{D}_{\delta;n} \mid \mathbf{o}\ ]$, the Petrov–Galerkin condition (4.4), which may be written as $\widetilde{\mathbf{Y}}_n^\star \mathbf{B} \mathbf{r}_n = \mathbf{o}$, finally yields the square tridiagonal linear system

$$\mathbf{T}_n \mathbf{k}_n = \mathbf{e}_1 \rho_0, \tag{4.7}$$

where now $\mathbf{e}_1 \in \mathbb{R}^n$. By solving it for $\mathbf{k}_n$ and inserting the solution into (4.5) we could compute $\mathbf{x}_n$. However, this approach, which is sometimes called the Lanczos method for solving linear systems, is very memory-intensive, as one has to store all right Lanczos vectors for evaluating (4.5). Fortunately, there are more efficient versions of the BiCG method that generate not only the residuals (essentially the right Lanczos vectors) but also the iterates with short recurrences. We could try to find such recurrences from the above relations, but we will derive them in a more general and more elegant way.

Unless one encounters a serious breakdown, the BiCG method terminates theoretically with $\mathbf{r}_\nu = \mathbf{o}$ or $\widetilde{\mathbf{y}}_\nu = \mathbf{o}$ for some ν. Therefore, the BiCG method also has the finite termination property, except that it is spoiled not only by round-off but also by the possibility of a breakdown (a serious

norm that depends on the created basis, that is, on $\mathbf{A}$ and $\mathbf{y}_0$. This result also follows easily from one of Hochbruck and Lubich (1997*a*)

one or a left-sided termination). We must emphasize again, however, that it is misleading to motivate the CG method or the BiCG method (in any of their forms) by this *finite termination property*, because this property is irrelevant when large linear systems are solved. What really counts are certain approximation properties that make the residuals (and errors) of the iterates $\mathbf{x}_n$ decrease rapidly. There is a simple, standard error bound that implies at least linear convergence for the CG and CR methods (see, for instance, Kaniel (1966), Saad (1980, 1994, 1996)), but in practice superlinear convergence is observed; there are indeed more sophisticated estimates that explain the superlinearity under certain assumptions on the spectrum (van der Sluis and van der Vorst 1986, 1987, Strakoš 1991, van der Sluis 1992, Hanke 1997). These bounds are no longer valid in the nonsymmetric case, but some of the considerations can be extended to it (van der Vorst and Vuik 1993, Ye 1991). The true mechanism of convergence lies deeper, and seems to remain the same in the nonsymmetric case. For the CR method and its generalization to nonsymmetric systems it has been analysed by Nevanlinna (1993). For the BiCG method convergence seems harder to analyse, however. Recently, a unified approach to error bounds for BiCG, QMR, FOM, and GMRes, as well as comparisons among their residual norms, have been established in Hochbruck and Lubich (1997*a*).

The BiCG method is based on Lanczos (1952) and Fletcher (1976), but, as we will see, there are various algorithms that realize it.

4.3. Recurrences for the BiCG *iterates; the consistency condition*

The recurrence for the iterates $\mathbf{x}_n$ is obtained from the one for the residuals by following a general rule that we will use over and over again. By definition, $\mathbf{x}_n - \mathbf{x}_0 \in \mathcal{K}_n$ for any Krylov space solver, and thus (4.2) holds; here $\mathcal{K}_n$ is still the nth Krylov space generated by $\mathbf{A}$ from $\mathbf{y}_0 = \mathbf{r}_0$. Since $\mathbf{r}_n = p_n(\mathbf{A})\mathbf{y}_0$ with a polynomial p_n of exact degree n, the vectors $\mathbf{r}_0, \ldots, \mathbf{r}_{n-1}$ span $\mathcal{K}_n$ (even when they are linearly dependent, in which case $\mathcal{K}_n = \mathcal{K}_{n-1}$). Therefore, if we let

$$\mathbf{R}_n := [\ \mathbf{r}_0 \quad \mathbf{r}_1 \quad \cdots \quad \mathbf{r}_{n-1}\ ], \quad \mathbf{X}_n := [\ \mathbf{x}_0 \quad \mathbf{x}_1 \quad \cdots \quad \mathbf{x}_{n-1}\ ],$$

and define the extended $(n+1) \times n$ Frobenius (or companion) matrix

$$\underline{\mathbf{F}}_n := \begin{bmatrix} -1 & -1 & \cdots & -1 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix},$$

then we have, in view of $\mathbf{x}_n - \mathbf{x}_0 \in \mathcal{K}_n$,

$$\mathbf{X}_{n+1}\underline{\mathbf{F}}_n = -\mathbf{R}_n\mathbf{U}_n \tag{4.8}$$

with some upper triangular $n \times n$ matrix $\mathbf{U}_n$ and an extra minus sign. Each column sum in $\underline{\mathbf{F}}_n$ is zero, that is, $[\ 1\ \ 1\ \ \cdots\ \ 1\ ]\,\underline{\mathbf{F}}_n = \mathbf{o}^\top$, and therefore, for an arbitrary $\mathbf{b} \in \mathbb{C}^N$, multiplication of $\underline{\mathbf{F}}_n$ from the left by the $N \times (n+1)$ matrix $[\ \mathbf{b}\ \ \mathbf{b}\ \ \cdots\ \ \mathbf{b}\ ]$ yields an $N \times n$ zero matrix. Therefore,

$$\mathbf{R}_{n+1}\underline{\mathbf{F}}_n = ([\ \mathbf{b}\ \ \cdots\ \ \mathbf{b}\ ] - \mathbf{A}\mathbf{X}_{n+1})\,\underline{\mathbf{F}}_n = -\mathbf{A}\mathbf{X}_{n+1}\underline{\mathbf{F}}_n = \mathbf{A}\mathbf{R}_n\mathbf{U}_n. \quad (4.9)$$

Since $\mathbf{r}_m$ and $\mathbf{A}\mathbf{r}_{m-1}$ are both represented by polynomials of exact degree m, the diagonal elements of $\mathbf{U}_n$ cannot vanish. Hence, if we let

$$\underline{\mathbf{H}}_n := \underline{\mathbf{F}}_n\mathbf{U}_n^{-1},$$

we can write (4.8) and (4.9) as

$$\mathbf{R}_n = -\mathbf{X}_{n+1}\underline{\mathbf{H}}_n, \quad \mathbf{A}\mathbf{R}_n = \mathbf{R}_{n+1}\underline{\mathbf{H}}_n, \quad (4.10)$$

where $\underline{\mathbf{H}}_n$ is an $(n+1) \times n$ upper Hessenberg matrix that satisfies[9]

$$\mathbf{e}^\top\underline{\mathbf{H}}_n = \mathbf{o}^\top, \quad \text{where } \mathbf{e}^\top := [\ 1\ \ 1\ \ \cdots\ \ 1\ ], \quad (4.11)$$

as a consequence of $\mathbf{e}^\top\underline{\mathbf{F}}_n = \mathbf{o}^\top$. This is the matrix form of the *consistency condition* for Krylov space solvers. It means that *in each column of* $\underline{\mathbf{H}}_n$ *the elements must sum up to* 0; see, for instance, Gutknecht (1989*b*). This property is inherited from $\underline{\mathbf{F}}_n$. The relations in (4.10) are the matrix representations of the recurrences for computing the iterates and the residuals: $\mathbf{x}_n$ is a linear combination of $\mathbf{r}_{n-1}$ and $\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}$, and $\mathbf{r}_n$ is a linear combination of $\mathbf{A}\mathbf{r}_{n-1}$ and $\mathbf{r}_0, \ldots, \mathbf{r}_{n-1}$. Note that the recurrence coefficients, which are stored in $\underline{\mathbf{H}}_n$, are the same in both formulae.

Another, equivalent form of the consistency condition is the property $p_n(0) = 1$ of the residual polynomials.

We call a Krylov space solver *consistent* if it generates a basis consisting of the residuals (and not of some multiples of them).

4.4. *The* BIORES *algorithm*

In the usual, consistent forms of the BICG method, the Lanczos vectors $\mathbf{y}_n$ are equal to the residuals $\mathbf{r}_n$ and thus the Lanczos polynomials satisfy the consistency condition $p_n(0) = 1$. To apply the above approach, we have to set $\underline{\mathbf{H}}_n := \underline{\mathbf{T}}_n$ and $\mathbf{R}_n := \mathbf{Y}_n$. Therefore, the zero column sum condition requires us to choose $\gamma_n := -\alpha_n - \beta_{n-1}$. However, this can lead to yet another type of breakdown, namely when $\alpha_n + \beta_{n-1} = 0$. Following Bank and Chan (1993) we call this a *pivot breakdown* (for reasons we will describe later, in Section 9), while a breakdown due to $\delta_{\text{temp}} = 0$ in the BIO algorithm is referred to as a *Lanczos breakdown*[10], as before.

[9] The dimension of the vectors $\mathbf{o}$ and $\mathbf{e}$ is always defined by the context.

[10] In Gutknecht (1990) we suggested calling a pivot breakdown a *normalization breakdown*, which is an appropriate name in view of its analogous occurrence in other Krylov space

Recalling that the formulae for the BIO algorithm can be simplified if γ_n is independent of δ_{temp}, and adding the appropriate recurrence for the approximants $\mathbf{x}_n$, we find the following BIORES version of the BICG method.

ALGORITHM 2. (BIORES FORM OF THE BICG METHOD)
For solving $\mathbf{Ax} = \mathbf{b}$, choose an initial approximation $\mathbf{x}_0$, set $\mathbf{y}_0 := \mathbf{b} - \mathbf{Ax}_0$, and choose $\widetilde{\mathbf{y}}_0$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle_{\mathbf{B}} \neq 0$. Then apply Algorithm 1 (BIO) with

$$\gamma_n := -\alpha_n - \beta_{n-1} \tag{4.12}$$

and some $\widetilde{\gamma}_n \neq 0$, so that (2.21e)–(2.21j) simplify to

$$\mathbf{y}_{n+1} := (\mathbf{A}\mathbf{y}_n - \mathbf{y}_n\alpha_n - \mathbf{y}_{n-1}\beta_{n-1})/\gamma_n, \tag{4.13a}$$

$$\widetilde{\mathbf{y}}_{n+1} := (\mathbf{A}^\star\widetilde{\mathbf{y}}_n - \widetilde{\mathbf{y}}_n\widetilde{\alpha}_n - \widetilde{\mathbf{y}}_{n-1}\widetilde{\beta}_{n-1})/\widetilde{\gamma}_n, \tag{4.13b}$$

$$\delta_{n+1} := \langle \widetilde{\mathbf{y}}_{n+1}, \mathbf{y}_{n+1} \rangle_{\mathbf{B}}\,. \tag{4.13c}$$

Additionally, compute the vectors

$$\mathbf{x}_{n+1} := -(\mathbf{y}_n + \mathbf{x}_n\alpha_n + \mathbf{x}_{n-1}\beta_{n-1})/\gamma_n. \tag{4.13d}$$

If $\gamma_n = 0$, the algorithm breaks down ('pivot breakdown'), and we set $\dot{\nu} := n$. If $\mathbf{y}_{n+1} = \mathbf{o}$, it terminates and $\mathbf{x}_{n+1}$ is the solution; if $\mathbf{y}_{n+1} \neq \mathbf{o}$, but $\delta_{n+1} = 0$, the algorithm also breaks down ('Lanczos breakdown' if $\widetilde{\mathbf{y}}_{n+1} \neq \mathbf{o}$, 'left termination' if $\widetilde{\mathbf{y}}_{n+1} = \mathbf{o}$). In these two cases we set $\dot{\nu} := n + 1$.

First we verify the relation between residuals and iterates.

Lemma 4.1 In Algorithm 2 (BIORES) the vector $\mathbf{y}_n$ is the residual of the nth iterate $\mathbf{x}_n$; that is, $\mathbf{b} - \mathbf{Ax}_n = \mathbf{y}_n$ $(n = 0, 1, \ldots, \dot{\nu})$.

Proof. First, $\mathbf{b} - \mathbf{Ax}_0 = \mathbf{y}_0$ by definition of $\mathbf{y}_0$. Assuming $n \geq 1$, $\mathbf{b} - \mathbf{Ax}_n = \mathbf{y}_n$, and $\mathbf{b} - \mathbf{Ax}_{n-1} = \mathbf{y}_{n-1}$, and using (4.13a)–(4.13d), (2.21e), (2.21j), and (4.12), we get

$$\begin{aligned} \mathbf{b} - \mathbf{Ax}_{n+1} &= \mathbf{b} + (\mathbf{Ay}_n + \mathbf{Ax}_n\alpha_n + \mathbf{Ax}_{n-1}\beta_{n-1})/\gamma_n \\ &= \mathbf{b} + (\mathbf{Ay}_n - \mathbf{y}_n\alpha_n - \mathbf{y}_{n-1}\beta_{n-1} + \mathbf{b}(\alpha_n + \beta_{n-1}))/\gamma_n \\ &= \mathbf{y}_{n+1}, \end{aligned}$$

which is what is needed for the induction. When $n = 0$, the same relations hold without the terms involving β_{-1}. □

solvers. Joubert (1992) calls the pivot breakdown a *hard breakdown* since it causes all three standard versions of the BICG method discussed in Jea and Young (1983) to break down, as we will see in Section 9. In his terminology the Lanczos breakdown is a *soft breakdown*. Brezinski, Redivo Zaglia and Sadok (1993) use the terms *true breakdown* and *ghost breakdown*, respectively, while Freund and Nachtigal (1991) refer to *breakdowns of the second kind* and *breakdowns of the first kind*. However, we will see that in the algorithms most often used in practice, it is easier to circumvent a pivot (or hard, or true, or second kind) breakdown than a Lanczos breakdown.

The shorthand notation (2.23)–(2.24) for the BIO algorithm can easily be extended to the BIORES algorithm. Due to the additional possibility of a pivot breakdown, the index of first breakdown or termination is now $\dot{\nu}$ $(\leq \nu)$. We also have to add the matrix representation of the recurrence for the iterates, (4.13d),

$$\mathbf{Y}_n = -\mathbf{X}_{n+1}\underline{\mathbf{T}}_n \quad (n \leq \dot{\nu}). \tag{4.14}$$

Analogously to (4.11), the column sum condition (4.12) can be expressed as

$$\mathbf{e}^\top \underline{\mathbf{T}}_n = \mathbf{o}^\top \quad (n \leq \dot{\nu}). \tag{4.15}$$

4.5. The inconsistent BIORES *algorithm*

We claim that by a small modification introduced in Gutknecht (1990) it is possible to avoid the pivot breakdown that may occur in Algorithm 2 (BIORES).

ALGORITHM 3. (INCONSISTENT BIORES ALGORITHM)
Initially, let $\mathbf{y}_0 := (\mathbf{b} - \mathbf{A}\mathbf{x}_0)/\gamma_{-1}$ with some $\gamma_{-1} \neq 0$, and redefine $\mathbf{x}_0 := \mathbf{x}_0/\gamma_{-1}$. (For example, choose $\gamma_{-1} := ||\mathbf{b} - \mathbf{A}\mathbf{x}_0||$ or $\gamma_{-1} := 1$.) Modify Algorithm 2 (BIORES) by always choosing $\gamma_n \neq 0$ (instead of setting $\gamma_n := -\alpha_n - \beta_{n-1}$). Compute additionally the sequence $\{\dot{\pi}_n\}$ that is defined recursively by

$$\dot{\pi}_0 := 1/\gamma_{-1}, \quad \dot{\pi}_{n+1} := -(\alpha_n \dot{\pi}_n + \beta_{n-1}\dot{\pi}_{n-1})/\gamma_n, \quad n = 0, 1, \ldots, \nu - 1. \tag{4.16}$$

We will see later in Theorem 12.1 that $\dot{\pi}_n$ is the value at 0 of the Lanczos polynomial p_n of (2.3), which up to normalization is also the residual polynomial of $\mathbf{x}_n$. We will also see that p_n, if normalized to be monic, is the characteristic polynomial of the $n \times n$ leading principal submatrix of $\mathbf{T}_\nu$. The problem with BIORES is that this value may become zero, and hence there may not exist a residual polynomial normalized to be 1 at $\zeta = 0$. In other words, inconsistent BIORES works with 'unnormalized residual polynomials' not satisfying the consistency condition. The same idea can be applied to other Krylov space solvers that break down for the same reason. It follows immediately that the pivot breakdown is avoided.

Lemma 4.2 The index of first breakdown or termination ν of the BIO algorithm and the one of the inconsistent BIORES algorithm are identical; the index of first breakdown or termination $\dot{\nu}$ of the consistent BIORES algorithm can, but need not, be smaller.

Moreover, in view of the following result, inconsistent BIORES delivers the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ whenever it does not break down.

Lemma 4.3 In the inconsistent BIORES algorithm, $\mathbf{y}_n$ and $\mathbf{x}_n$ are related by

$$\mathbf{y}_n = \mathbf{b}\dot{\pi}_n - \mathbf{A}\mathbf{x}_n. \tag{4.17}$$

If $\mathbf{y}_\nu\gamma_\nu = 0$, then $\dot{\pi}_\nu \neq 0$, and $\mathbf{x}_{\text{ex}} = \mathbf{x}_\nu/\dot{\pi}_\nu$ is the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Proof. For $n = 0$, (4.17) is correct. Assume it is correct up to the index n. Then by (4.13a)–(4.13d) and (4.16)

$$\begin{aligned}(\mathbf{b}\dot{\pi}_{n+1} - \mathbf{A}\mathbf{x}_{n+1})\gamma_n \\ &= -\mathbf{b}(\alpha_n\dot{\pi}_n + \beta_{n-1}\dot{\pi}_{n-1}) + \mathbf{A}\mathbf{y}_n + \mathbf{A}\mathbf{x}_n\alpha_n + \mathbf{A}\mathbf{x}_{n-1}\beta_{n-1} \\ &= \mathbf{A}\mathbf{y}_n - \mathbf{y}_n\alpha_n - \mathbf{y}_{n-1}\beta_{n-1} = \mathbf{y}_{n+1}\gamma_n.\end{aligned}$$

As mentioned above, $\dot{\pi}_n$ is, up to normalization, the value at 0 of the characteristic polynomial p_n of the $n \times n$ leading principal submatrix of $\mathbf{T}_\nu$. In particular, $\dot{\pi}_\nu$ is a nonzero multiple of the value at 0 of the characteristic polynomial p_ν of $\mathbf{T}_\nu$. We know that when the algorithm terminates due to $\mathbf{y}_{\text{temp}} = \mathbf{y}_\nu\gamma_\nu = \mathbf{o}$, then the eigenvalues of $\mathbf{T}_\nu$ are also eigenvalues of $\mathbf{A}$. Hence, $\dot{\pi}_\nu = 0$ would imply that $\mathbf{A}$ is singular, contrary to our assumption in this chapter. □

Lemma 4.3 indicates that in practice termination should be based on $||\mathbf{y}_n||/|\dot{\pi}_n|$ being small. If we let

$$\dot{\mathbf{p}}_n := [\ \dot{\pi}_0 \quad \cdots \quad \dot{\pi}_{n-1}\ ]^\top,$$

we can formulate the extra recurrence (4.16) of inconsistent BIORES and its residual relation (4.17) as

$$\dot{\mathbf{p}}_{n+1}^\top \underline{\mathbf{T}}_n = \mathbf{o}^\top \quad (n \leq \nu),$$

$$\mathbf{Y}_n = [\ \mathbf{b} \quad \cdots \quad \mathbf{b}\ ]\,\mathrm{diag}(\dot{\pi}_0, \cdots, \dot{\pi}_{n-1}) - \mathbf{A}\mathbf{X}_n \quad (n \leq \nu + 1).$$

From (4.17) we conclude that $\dot{\pi}_n$ and the choice of γ_m $(m \leq n)$ only affect the scaling of $\mathbf{y}_n$ and $\mathbf{x}_n$. It is clear from this formula that whenever $\dot{\pi}_n \neq 0$ for all $n < \nu$, one can rescale $\mathbf{y}_n$, $\mathbf{x}_n$ $(n \leq \nu)$ to get the corresponding vectors of (consistent) BIORES. But once $\dot{\pi}_n = 0$ for some $n < \nu$, this is impossible and BIORES breaks down, that is, $\dot{\nu} < \nu$. In contrast, here one can still go on, and if $\mathbf{y}_\nu = \mathbf{o}$ one finds a solution that is not accessible through Algorithm 2 (using the same initial data). In practice, where vanishing of $\dot{\pi}_n$ is unlikely, but near-vanishing matters, inconsistent BIORES must be considered as a slightly stabilized version of BIORES that eliminates the possibility of overflow or division by zero. In floating-point arithmetic, however, there is no other stability pitfall caused by the particular scaling of (consistent) BIORES or by any other scaling.

5. The QMR solution of a linear system

While the BiCG method yields a Petrov–Galerkin approximation of the solution of a linear system, the *Quasi-Minimal Residual* (QMR) *method* of Freund and Nachtigal (Freund 1992, Freund and Nachtigal 1991, Freund and Nachtigal 1994) produces a solution whose residual has a coordinate vector of minimum length. However, since the basis of the space is – for economy reasons – the one generated by the Lanczos process, and thus is not orthonormal, in general, the residual vector itself is not of minimum length.

Basically, the QMR method takes the right Krylov space basis generated by the Lanczos process and solves a least squares problem in coordinate space in the same way as the MinRes algorithm of Paige and Saunders (1975) and the GMRes algorithm of Saad and Schultz (1986). However, the QMR algorithm in (Freund and Nachtigal 1991) has additional features: its Lanczos part includes an implementation of look-ahead from Freund et al. (1993), and its least squares part allows for weights, which, however, are rarely used and therefore dropped in our presentation[11].

In principle, the QMR philosophy has a wide scope of applications, which goes beyond what has been treated in the literature. In particular, we can apply it to any Krylov space generation procedure producing a relation of the form $\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{H}}_n$ (preferably with column vectors of norm 1) or of certain equivalent forms. We will return to this in Sections 10, 15 and 17.

Since MinRes plays an essential role in QMR, we need to look at it first. We are going to discuss a variation of it that is suitable for QMR, since it is easily adapted to allow for look-ahead.

5.1. *The* MinRes *algorithm*

The MinRes algorithm of Paige and Saunders (1975), as well as QMR and GMRes, start from the representation (4.6) of the residual. MinRes is a particular algorithm for the CR method for Hermitian systems, and thus the aim is to minimize the residual norm. The method makes use of the isometry induced by the coordinate mapping of an inner product space with an orthonormal basis. This isometry is also manifested by the well-known Parseval relation. It implies that instead of minimizing the residual we can minimize its coordinate vector. In fact, by running the symmetric Lanczos algorithm with $\mathbf{B} = \mathbf{I}$ (despite the fact that the inner product matrix $\mathbf{B} = \mathbf{A}$ is used in the minimization problem (4.3) of the CR method) and normalizing the resulting orthogonal basis $\{\mathbf{y}_m\}$ of the Krylov space,

[11] In Tong (1994) the diagonal weight matrix is replaced by a block diagonal one with 2×2 or 3×3 blocks that are chosen suitably. However, the numerical results show little gain in efficiency, if any at all.

we have $\mathbf{Y}_{n+1}^{\star}\mathbf{Y}_{n+1} = \mathbf{I}_{n+1}$, and therefore from (4.6)

$$||\mathbf{r}_n||^2 = ||\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}_n\mathbf{k}_n||^2. \tag{5.1}$$

This is a least squares problem in coordinate space: $\underline{\mathbf{e}}_1\rho_0$ has to be approximated by a linear combination of the columns of the tridiagonal $(n+1)\times n$ matrix $\underline{\mathbf{T}}_n$.

For example, this problem can be solved using the QR or the LQ decomposition of the matrix and, due to the tridiagonality, these decompositions only require n or $n-1$ Givens rotations, respectively[12]. The QR decomposition of an upper Hessenberg matrix of the same size still only requires $n+1$ rotations, and that is why both the GMRES and the QMR algorithms apply QR, while Paige and Saunders used an LQ decomposition. In GMRES $\underline{\mathbf{T}}_n$ is replaced by a Hessenberg matrix, and in the QMR algorithm $\underline{\mathbf{T}}_n$ is tridiagonal except for a few extra nonzero elements above the upper codiagonal if look-ahead is needed; hence, it is a nearly tridiagonal Hessenberg matrix. Although we assume in our presentation of the QMR method in this section that look-ahead does not occur, we choose to work with the QR decomposition, and we modify the original MINRES algorithm accordingly. Our treatment is adapted from Freund and Nachtigal (1991).

Let $\underline{\mathbf{T}}_n = \mathbf{Q}_n\underline{\mathbf{R}}_n^{\mathrm{MR}}$ be a QR decomposition of $\underline{\mathbf{T}}_n$. The last row of the upper triangular $(n+1)\times n$ matrix $\underline{\mathbf{R}}_n^{\mathrm{MR}}$ is zero. If we denote its upper square $n\times n$ submatrix by $\mathbf{R}_n^{\mathrm{MR}}$ (not to be confused with the matrix $\mathbf{R}_n$ of residual vectors) and let

$$\underline{\mathbf{h}}_n := \left[\begin{array}{c}\mathbf{h}_n\\ \hline \widetilde{\eta}_{n+1}\end{array}\right] := \mathbf{Q}_n^{\star}\underline{\mathbf{e}}_1\rho_0, \tag{5.2}$$

we see that

$$\mathbf{k}_n := (\mathbf{R}_n^{\mathrm{MR}})^{-1}\mathbf{h}_n \tag{5.3}$$

is the solution of our least squares problem since

$$\begin{aligned} ||\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}_n\mathbf{k}_n||^2 &= ||\mathbf{Q}_n^{\star}\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{R}}_n^{\mathrm{MR}}\mathbf{k}_n||^2 \\ &= ||\underline{\mathbf{h}}_n - \underline{\mathbf{R}}_n^{\mathrm{MR}}\mathbf{k}_n||^2 \qquad (5.4)\\ &= ||\mathbf{h}_n - \mathbf{R}_n^{\mathrm{MR}}\mathbf{k}_n||^2 + |\widetilde{\eta}_{n+1}|^2 \\ &= |\widetilde{\eta}_{n+1}|^2. \qquad (5.5)\end{aligned}$$

In fact, multiplying the least squares problem (5.1) by the unitary matrix $\mathbf{Q}_n^{\star}$ turns it into one with an upper triangular matrix, see (5.4), where the choice of $\mathbf{k}_n$ no longer influences the defect of the last equation, and thus the problem is solved by choosing $\mathbf{k}_n$ such that the first n equations are fulfilled.

[12] An alternative is to apply Householder transformations; see Walker (1988).

From (5.1) and (5.5) we see in particular that the minimum residual norm is equal to $|\widetilde{\eta}_{n+1}|$ and hence can be found without computing $\mathbf{k}_n$ or the residual. The unitary matrix $\mathbf{Q}_n$ is only determined in its factored form, as the product of n Givens rotations that are chosen to annihilate the subdiagonal elements of the tridiagonal (or Hessenberg) matrix

$$\mathbf{Q}_n := \left[\begin{array}{c|c} \mathbf{Q}_{n-1} & \mathbf{o} \\ \hline \mathbf{o}^\top & 1 \end{array}\right] \mathbf{G}_n \quad \text{with} \quad \mathbf{G}_n := \left[\begin{array}{c|cc} \mathbf{I}_{n-1} & \mathbf{o} & \mathbf{o} \\ \hline \mathbf{o}^\top & c_n & -s_n \\ \mathbf{o}^\top & \overline{s_n} & c_n \end{array}\right], \tag{5.6}$$

where $c_n \geq 0$ and $s_n \in \mathbb{C}^n$ satisfying $c_n^2 + |s_n^2| = 1$ are chosen such that

$$\mathbf{G}_n^\star \begin{bmatrix} \star \\ \vdots \\ \star \\ \mu_n \\ \nu_n \end{bmatrix} = \begin{bmatrix} \star \\ \vdots \\ \star \\ c_n\mu_n + s_n\nu_n \\ 0 \end{bmatrix} \quad \text{with} \quad \begin{bmatrix} \star \\ \vdots \\ \star \\ \mu_n \\ \nu_n \end{bmatrix} := \left[\begin{array}{c|c} \mathbf{Q}_{n-1}^\star & \mathbf{o} \\ \hline \mathbf{o}^\top & 1 \end{array}\right] \underline{\mathbf{T}}_n \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

which means that

$$\begin{aligned} c_n &:= \frac{|\mu_n|}{\sqrt{|\mu_n|^2 + |\nu_n|^2}}, \quad s_n := \overline{c_n \frac{\nu_n}{\mu_n}}, \quad && \text{if } \mu_n \neq 0, \\ c_n &:= 0, && s_n := 1, && \text{if } \mu_n = 0. \end{aligned} \tag{5.7}$$

If $\underline{\mathbf{T}}_n$ is real, c_n and s_n are the cosine and sine of the rotation angle.

The formula for updating $\underline{\mathbf{h}}_n$ is therefore very simple:

$$\left[\begin{array}{c} \mathbf{h}_n \\ \hline \widetilde{\eta}_{n+1} \end{array}\right] = \underline{\mathbf{h}}_n = \mathbf{G}_n^\star \left[\begin{array}{c} \underline{\mathbf{h}}_{n-1} \\ \hline 0 \end{array}\right] = \mathbf{G}_n^\star \left[\begin{array}{c} \mathbf{h}_{n-1} \\ \hline \widetilde{\eta}_n \\ 0 \end{array}\right] = \left[\begin{array}{c} \mathbf{h}_{n-1} \\ \hline c_n\,\widetilde{\eta}_n \\ -\overline{s_n}\,\widetilde{\eta}_n \end{array}\right]. \tag{5.8}$$

In particular, it follows that

$$||\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}_n\mathbf{k}_n|| = |\widetilde{\eta}_{n+1}| = |s_n\widetilde{\eta}_n| = |s_1\, s_2 \cdots s_n|\, ||\mathbf{r}_0||, \tag{5.9}$$

since $\widetilde{\eta}_1 = ||\mathbf{r}_0||$. Even more important is the fact that $\mathbf{h}_n \in \mathbb{C}^n$ emerges from $\mathbf{h}_{n-1} \in \mathbb{C}^{n-1}$ by just appending an additional component $c_n\widetilde{\eta}_n$. By rewriting the first equation in (4.5) using (5.3) as

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Z}_n\mathbf{h}_n, \quad \text{where } \mathbf{Z}_n := [\ \mathbf{z}_0 \ \ \ldots \ \ \mathbf{z}_{n-1}\ ] := \mathbf{Y}_n(\mathbf{R}_n^{\mathrm{MR}})^{-1}$$

contains the QMR direction vectors, we can conclude that

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{z}_{n-1}c_n\widetilde{\eta}_n. \tag{5.10}$$

Finally, since $\mathbf{R}_n^{\mathrm{MR}}$ is a banded upper tridiagonal matrix with bandwidth three, the relation

$$\mathbf{Y}_n = \mathbf{R}_n^{\mathrm{MR}} \mathbf{Z}_n \tag{5.11}$$

can be viewed as the matrix representation of a three-term recurrence for generating the vectors $\{\mathbf{z}_k\}_{k=0}^{n-1}$. (In contrast, in GMRES $\mathbf{R}_n^{\mathrm{MR}}$ is no longer banded, and therefore this recurrence is not short.)

Multiplying (5.10) by $\mathbf{A}$ we could find an analogous recurrence for the residuals, but since it would require an extra matrix-vector product, it is of no interest. There is another, cheaper way of updating the residual. First, inserting $\underline{\mathbf{T}}_n = \mathbf{Q}_n \underline{\mathbf{R}}_n^{\mathrm{MR}}$ and (5.3) into (4.6) and taking (5.2) into account we get

$$\begin{aligned} \mathbf{r}_n &= \mathbf{Y}_{n+1} \left(\underline{\mathbf{e}}_1 \rho_0 - \mathbf{Q}_n \underline{\mathbf{R}}_n^{\mathrm{MR}} (\mathbf{R}_n^{\mathrm{MR}})^{-1} \mathbf{h}_n \right) = \mathbf{Y}_{n+1} \left(\underline{\mathbf{e}}_1 \rho_0 - \mathbf{Q}_n \begin{bmatrix} \mathbf{h}_n \\ \hline 0 \end{bmatrix} \right) \\ &= \mathbf{Y}_{n+1} \mathbf{Q}_n \, \mathbf{l}_{n+1} \widetilde{\eta}_{n+1}, \quad \text{where } \mathbf{l}_{n+1} = [\, 0 \quad \dots \quad 0 \quad 1 \,]^\top \in \mathbb{R}^{n+1} \end{aligned} \tag{5.12}$$

as before. Using (5.6) we conclude further that

$$\begin{aligned} \mathbf{r}_n &= \left[\begin{array}{c|c} \mathbf{Y}_n & \mathbf{y}_n \end{array} \right] \left[\begin{array}{c|c} \mathbf{Q}_{n-1} & \mathbf{o} \\ \hline \mathbf{o}^\top & 1 \end{array} \right] \mathbf{G}_n \begin{bmatrix} \mathbf{o} \\ \hline 1 \end{bmatrix} \widetilde{\eta}_{n+1} \\ &= -\mathbf{Y}_n \mathbf{Q}_{n-1} \mathbf{l}_n s_n \widetilde{\eta}_{n+1} + \mathbf{y}_n c_n \widetilde{\eta}_{n+1}. \end{aligned}$$

Finally, using (5.12) and $\widetilde{\eta}_{n+1} = -\overline{s_n}\, \widetilde{\eta}_n$ (see (5.8)) to simplify the first term on the right-hand side, we get the recursion

$$\mathbf{r}_n = \mathbf{r}_{n-1} |s_n|^2 + \mathbf{y}_n c_n \widetilde{\eta}_{n+1}. \tag{5.13}$$

However, recall that updating the residual is unnecessary for MINRES since its norm is equal to $|\widetilde{\eta}_{n+1}|$. But (5.13) also holds for GMRES, and, since we have not used the fact that $\mathbf{Y}_n$ has orthogonal columns, it will become clear that it remains true for QMR.

5.2. *A first version of the* QMR *method:* BIOQMR

The basic version of the QMR method without look-ahead is now easily explained: the BIO algorithm with normalized Lanczos vectors (that is, with normalization (2.27)) is applied to build up bases of the growing Krylov spaces $\mathcal{K}_n$ and $\widetilde{\mathcal{K}}_n$. As in inconsistent BIORES, the right initial vector $\mathbf{y}_0 := \mathbf{r}_0 / \|\mathbf{r}_0\|$ is the normalized initial residual, while the left one, $\widetilde{\mathbf{y}}_0$, can be chosen arbitrarily. The relations (4.5)–(4.6) remain valid, but (5.1) is no longer true, since the basis $\{\mathbf{y}_k\}_{k=0}^{n-1}$ is no longer orthonormal when $\mathbf{A}$ is not Hermitian or $\widetilde{\mathbf{y}}_0 \neq \mathbf{y}_0$.

Since finding the minimum residual becomes too expensive, Freund and Nachtigal (Freund 1992, Freund and Nachtigal 1991) instead promoted minimizing the coefficient vector of the residual with respect to that basis, the so-called *quasi-residual*

$$\mathbf{q}_n := \underline{\mathbf{e}}_1 \rho_0 - \underline{\mathbf{T}}_n \mathbf{k}_n, \quad \text{satisfying } \mathbf{r}_n = \mathbf{Y}_{n+1} \mathbf{q}_n, \tag{5.14}$$

see (4.6). Minimizing $||\mathbf{q}_n||$ is accomplished exactly as described in the previous subsection, and even the recurrences (5.10) and (5.13) for updating the iterates and residuals, respectively, remain valid.

What differs, however, is that $||\mathbf{r}_n|| = ||\mathbf{q}_n||$ no longer holds, in general. Instead we just have

$$||\mathbf{r}_n|| \leq ||\mathbf{Y}_{n+1}||\, ||\mathbf{q}_n|| \leq \sqrt{n+1}\, |\widetilde{\eta}_{n+1}|,$$

since $\mathbf{Y}_{n+1}$ has columns of length 1, and $||\mathbf{q}_n|| = |\widetilde{\eta}_{n+1}|$ as before; see (5.5). The factor $\sqrt{n+1}$ normally leads to a large overestimate, so that the bound is of limited value. However, the relationship between the residual and the quasi-residual may suggest sparing the work for updating the residual and computing its norm until the norm $|\widetilde{\eta}_{n+1}|$ of the quasi-residual has dropped below a certain tolerance. In the following summary of a (simplified) version of the QMR method we nevertheless assume that the residual is updated. We choose to call it BIOQMR for distinction, to indicate that it is based on the BIO algorithm without look-ahead, whose results are then piped into the QMR least squares process.

ALGORITHM 4. (BIOQMR VERSION OF THE QMR METHOD)
For solving $\mathbf{A}\mathbf{x} = \mathbf{b}$, choose an initial approximation $\mathbf{x}_0 \in \mathbb{C}^N$, let $\mathbf{r}_0 := (\mathbf{b} - \mathbf{A}\mathbf{x}_0)$ and $\mathbf{y}_0 := \mathbf{r}_0/||\mathbf{r}_0||$, choose $\widetilde{\mathbf{y}}_0$ of unit length, and apply Algorithm 1 (BIO) with the option (2.27) producing normalized Lanczos vectors. Within step $n-1$ of the main loop, after generating $\mathbf{y}_n$ and $\widetilde{\mathbf{y}}_n$,

(1) update the QR factorization $\underline{\mathbf{T}}_n = \mathbf{Q}_n \underline{\mathbf{R}}_n^{\mathrm{MR}}$ according to (5.6)–(5.7)
(2) compute the coefficient vector $\mathbf{h}_n$ by appending the component $c_n \widetilde{\eta}_n$ to $\mathbf{h}_{n-1}$, and compute the new last component $\widetilde{\eta}_{n+1} := -\overline{s_n}\, \widetilde{\eta}_n$ of $\underline{\mathbf{h}}_n$
(3) compute $\mathbf{z}_{n-1}$ according to the three-term recurrence implied by (5.11)
(4) compute $\mathbf{x}_n$ and $\mathbf{r}_n$ according to (5.10) and (5.13), respectively
(5) stop if $||\mathbf{r}_n||/||\mathbf{r}_0||$ is sufficiently small.

Note that the extra cost (in excess of those for the BIO algorithm) is very small. On the other hand, the smoothing effect of the QMR method is often very striking: while the Petrov–Galerkin condition imposed in the BICG method sometimes leads to a rather erratic residual norm plot, the norms of the QMR residuals typically decrease nearly monotonically, though not necessarily completely monotonically; see, for instance, the examples in Freund and Nachtigal (1991).

5.3. *The relation between (Petrov–)Galerkin and the (Quasi-)Minimal Residual solutions*

Between the BiCG iterates and residuals and those of the QMR method exist relationships inherited from CG and CR. For the latter two methods, most of them were found by Paige and Saunders (1975) as a byproduct of their derivation of the MinRes algorithm from the symmetric Lanczos process, but more transparent derivations and some new results and interpretations have been found more recently. These relations between Galerkin-based and minimal-residual-based solutions carry over in a straightforward way to the corresponding orthogonalization methods for nonsymmetric systems, the Arnoldi method (or FOM) and the GMRes algorithm for the GCR method, as was shown by Brown (1991).

The transition from CG to CR and from FOM to GCR is also possible by applying to the CG residuals or the Arnoldi residuals, respectively, the minimal residual smoothing process that we will discuss in Section 17. In particular, we will see there why a peak in the FOM residual norm plot leads to a plateau in the one of GCR.

Here we follow first the treatment of Freund and Nachtigal (1991); see also Paige and Saunders (1975, pp. 625–626).

Recall that the Galerkin condition of CG and the Petrov–Galerkin condition of BiCG yield in coordinate space the linear system (4.7),

$$\mathbf{T}_n \mathbf{k}_n^{\mathrm{G}} = \mathbf{e}_1 \rho_0, \tag{5.15}$$

while MinRes and QMR require us to minimize the quasi-residual

$$\mathbf{q}_n := \underline{\mathbf{e}}_1 \rho_0 - \underline{\mathbf{T}}_n \mathbf{k}_n^{\mathrm{MR}} \tag{5.16}$$

of (5.14). Now we have to distinguish the two coordinate vectors, and we will likewise denote the respective iterates and residuals by $\mathbf{x}_n^{\mathrm{G}}$, $\mathbf{r}_n^{\mathrm{G}}$ and $\mathbf{x}_n^{\mathrm{MR}}$, $\mathbf{r}_n^{\mathrm{MR}}$. The results we are going to derive hold for any of the three relations CG–CR, BiCG–QMR, and FOM–GMRes, but some minor modifications in the derivation are needed for the last pair.

The minimization of $||\mathbf{q}_n||^2$ is the least squares problem that we solved in the first subsection by QR decomposition of the $(n+1) \times n$ tridiagonal matrix $\underline{\mathbf{T}}_n$. (The latter could be replaced by the $(n+1) \times n$ upper Hessenberg matrix produced by GMRes or by QMR with look-ahead.) Inserting the update formula (5.6) for $\mathbf{Q}_n$ into $\underline{\mathbf{T}}_n = \mathbf{Q}_n \underline{\mathbf{R}}_n^{\mathrm{MR}}$ and moving the accumulated left factor to the left-hand side of this equation, we get

$$\left[\begin{array}{c|c} \mathbf{Q}_{n-1}^{\star} & \mathbf{o} \\ \hline \mathbf{o}^{\top} & 1 \end{array}\right] \underline{\mathbf{T}}_n = \mathbf{G}_n \underline{\mathbf{R}}_n^{\mathrm{MR}}.$$

Deleting the last row yields

$$\mathbf{Q}_{n-1}^{\star}\mathbf{T}_n = \left[\begin{array}{c|c} \mathbf{I}_{n-1} & \mathbf{o} \\ \hline \mathbf{o}^{\top} & c_n \end{array}\right] \mathbf{R}_n^{\mathrm{MR}} =: \mathbf{R}_n^{\mathrm{G}},$$

where $\mathbf{R}_n^{\mathrm{G}}$ is again upper triangular and, hence, $\mathbf{T}_n = \mathbf{Q}_{n-1}\mathbf{R}_n^{\mathrm{G}}$ is the QR decomposition of the square tridiagonal matrix $\mathbf{T}_n$. Of course, we can solve (5.15) using this decomposition, getting

$$\begin{aligned} \mathbf{k}_n^{\mathrm{G}} &= (\mathbf{R}_n^{\mathrm{G}})^{-1}\,\mathbf{Q}_{n-1}^{\star}\mathbf{e}_1\rho_0 = (\mathbf{R}_n^{\mathrm{MR}})^{-1}\left[\begin{array}{c|c} \mathbf{I}_{n-1} & \mathbf{o} \\ \hline \mathbf{o}^{\top} & c_n^{-1} \end{array}\right]\mathbf{Q}_{n-1}^{\star}\mathbf{e}_1\rho_0 \\ &= (\mathbf{R}_n^{\mathrm{MR}})^{-1}\left(\left[\begin{array}{c|c} \mathbf{I}_{n-1} & \mathbf{o} \\ \hline \mathbf{o}^{\top} & c_n \end{array}\right] + \left[\begin{array}{c|c} \mathbf{O}_{n-1} & \mathbf{o} \\ \hline \mathbf{o}^{\top} & c_n^{-1} - c_n \end{array}\right]\right) \\ &\quad \times \mathbf{Q}_{n-1}^{\star}\mathbf{e}_1\rho_0. \end{aligned} \tag{5.17}$$

On the other hand, from (5.2) and (5.3) we conclude by inserting (5.6) that

$$\begin{aligned} \mathbf{k}_n^{\mathrm{MR}} &= (\mathbf{R}_n^{\mathrm{MR}})^{-1}\mathbf{h}_n = (\mathbf{R}_n^{\mathrm{MR}})^{-1}\left[\begin{array}{c|c} \mathbf{I}_n & \mathbf{o} \end{array}\right]\mathbf{Q}_{n}^{\star}\underline{\mathbf{e}}_1\rho_0 \\ &= (\mathbf{R}_n^{\mathrm{MR}})^{-1}\left[\begin{array}{c|c} \mathbf{I}_n & \mathbf{o} \end{array}\right]\left[\begin{array}{c|cc} \mathbf{I}_{n-1} & \mathbf{o} & \mathbf{o} \\ \hline \mathbf{o}^{\top} & c_n & s_n \\ \mathbf{o}^{\top} & -\overline{s_n} & c_n \end{array}\right]\left[\begin{array}{c|c} \mathbf{Q}_{n-1}^{\star} & \mathbf{o} \\ \hline \mathbf{o}^{\top} & 1 \end{array}\right]\underline{\mathbf{e}}_1\rho_0 \\ &= (\mathbf{R}_n^{\mathrm{MR}})^{-1}\left[\begin{array}{c|c} \mathbf{I}_{n-1} & \mathbf{o} \\ \hline \mathbf{o}^{\top} & c_n \end{array}\right]\mathbf{Q}_{n-1}^{\star}\mathbf{e}_1\rho_0, \end{aligned}$$

which shows that the first of the two terms in (5.17) is just $\mathbf{k}_n^{\mathrm{G}}$. To simplify the other we note that by (5.2) the last component of $\mathbf{Q}_{n-1}^{\star}\mathbf{e}_1\rho_0$ is $\widetilde{\eta}_n$, so that altogether:

$$\mathbf{k}_n^{\mathrm{G}} = \mathbf{k}_n^{\mathrm{MR}} + (\mathbf{R}_n^{\mathrm{MR}})^{-1}\left[\begin{array}{c} \mathbf{o} \\ \hline c_n^{-1} - c_n \end{array}\right]\widetilde{\eta}_n. \tag{5.18}$$

For the iterates, which are in both cases of the form $\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Y}_n\mathbf{k}_n$, we find the relation

$$\mathbf{x}_n^{\mathrm{G}} = \mathbf{x}_n^{\mathrm{MR}} + \mathbf{Y}_n(\mathbf{R}_n^{\mathrm{MR}})^{-1}\left[\begin{array}{c} \mathbf{o} \\ \hline c_n^{-1} - c_n \end{array}\right]\widetilde{\eta}_n = \mathbf{x}_n^{\mathrm{MR}} + \mathbf{z}_{n-1}(c_n^{-1} - c_n)\widetilde{\eta}_n,$$

or, since $c_n^{-1} - c_n = |s_n|^2/c_n$, finally,

$$\mathbf{x}_n^{\mathrm{G}} = \mathbf{x}_n^{\mathrm{MR}} + \mathbf{z}_{n-1}\frac{|s_n|^2\widetilde{\eta}_n}{c_n}. \tag{5.19}$$

This formula allows us to compute the BiCG iterates from quantities produced by the BiOQMR algorithm. But, of course, we could also generate these iterates recursively according to inconsistent BiORes, that is, from (4.13d), (4.16), and $\mathbf{x}_n^{\mathrm{G}} := \mathbf{x}_n/\dot{\pi}_n$; see Lemma 4.3.

Multiplication of (5.19) by $\mathbf{A}$ and subtraction from $\mathbf{b}$ yields an analogue relation for the residuals. However, its direct usage would cost an extra matrix-vector multiplication. Moreover, inserting $\mathbf{z}_{n-1}$ according to (5.10) and making use of (5.13) leads in a few lines to

$$\mathbf{r}_n^{\mathrm{G}} = \mathbf{y}_n \frac{\widetilde{\eta}_{n+1}}{c_n}. \tag{5.20}$$

This is no surprise since we know that the CG and BiCG residuals are multiples of the Lanczos vectors. The analogue also holds for the FOM residuals. What we learn is that, using (5.9) and (5.16), we can express the residual norm as

$$||\mathbf{r}_n^{\mathrm{G}}|| = \frac{|\widetilde{\eta}_{n+1}|}{c_n} = \frac{1}{c_n}\,||\mathbf{q}_n|| = \frac{|s_1 s_2 \cdots s_n|}{c_n}\,||\mathbf{r}_0||. \tag{5.21}$$

As shown by Paige and Saunders (1975, p. 623), formula (5.20) is easily obtained directly: splitting $\mathbf{Y}_{n+1}$ up into its first n and its last column, we get from (4.6)

$$\mathbf{r}_n^{\mathrm{G}} = \mathbf{Y}_n(\mathbf{e}_1\rho_0 - \mathbf{T}_n\mathbf{k}_n^{\mathrm{G}}) - \mathbf{y}_n\gamma_n(\mathbf{l}^{\top}\mathbf{k}_n^{\mathrm{G}}).$$

Here, the first term vanishes, and for the second we see from (5.17) that the last component of $\mathbf{k}_n^{\mathrm{G}}$ is $\mathbf{l}^{\top}\mathbf{k}_n^{\mathrm{G}} = \widetilde{\eta}_n/(\rho_{n,n}c_n)$, where $\rho_{n,n}$ is the (n,n)-element of $\mathbf{R}_n^{\mathrm{MR}}$, which, in view of $\underline{\mathbf{T}}_n = \mathbf{Q}_n\underline{\mathbf{R}}_n^{\mathrm{MR}}$ and (5.6), is linked to γ_n by $\gamma_n = \overline{s_n}\rho_{n,n}$, so that

$$\mathbf{r}_n^{\mathrm{G}} = -\mathbf{y}_n\frac{\gamma_n\widetilde{\eta}_n}{\rho_{n,n}c_n} = -\mathbf{y}_n\frac{\overline{s_n}\,\widetilde{\eta}_n}{c_n} = \mathbf{y}_n\frac{\widetilde{\eta}_{n+1}}{c_n}.$$

This result also means that the quasi-residual $\mathbf{q}_n^{\mathrm{G}}$ that one can associate according to (5.14) with a Galerkin method is given by

$$\mathbf{q}_n^{\mathrm{G}} = \mathbf{l}_{n+1}\frac{\widetilde{\eta}_{n+1}}{c_n}.$$

From (5.20) we conclude further that (5.13) can be rewritten as

$$\mathbf{r}_n^{\mathrm{MR}} = \mathbf{r}_{n-1}^{\mathrm{MR}}|s_n|^2 + \mathbf{r}_n^{\mathrm{G}}c_n^2, \tag{5.22}$$

and, in view of $|s_n|^2 + c_n^2 = 1$, subtraction from $\mathbf{b}$ and premultiplication by $\mathbf{A}^{-1}$ yields

$$\mathbf{x}_n^{\mathrm{MR}} = \mathbf{x}_{n-1}^{\mathrm{MR}}|s_n|^2 + \mathbf{x}_n^{\mathrm{G}}c_n^2. \tag{5.23}$$

Finally, once again using $|s_n|^2 + c_n^2 = 1$ and $\widetilde{\eta}_{n+1} = -\overline{s_n}\,\widetilde{\eta}_n$ (see (5.8)), we

see that

$$\frac{1}{|\widetilde{\eta}_{n+1}|^2} = \frac{1}{|\widetilde{\eta}_n|^2} + \frac{c_n^2}{|\widetilde{\eta}_{n+1}|^2} = \frac{1}{|\widetilde{\eta}_n|^2} + \frac{1}{||\mathbf{r}_n^G||^2}, \tag{5.24}$$

which allows us to find $|\widetilde{\eta}_{n+1}|^2$ recursively without the QR decomposition, just from the residual norms of the Galerkin method. Then, weights in (5.22) and (5.23) are obtained from

$$c_n^2 = \frac{|\widetilde{\eta}_{n+1}|^2}{||\mathbf{r}_n^G||^2}, \quad |s_n|^2 = 1 - c_n^2. \tag{5.25}$$

Recall that $|\widetilde{\eta}_{n+1}| = ||\mathbf{r}_n^{\mathrm{MR}}||$ in the CR and FOM settings, while $|\widetilde{\eta}_{n+1}| = ||\mathbf{q}_n||$ is the norm of the quasi-residual if we apply the above to the BiCG–QMR connection. The relations (5.22)–(5.25) open up an alternative way to compute the QMR (or CR or GCR) iterates and residuals from the corresponding Galerkin residuals, that is, the BiCG (or CG or FOM) residuals. Zhou and Walker (1994), who introduced this approach, call it QMR *smoothing*. We will return to it in Section 17.

The relation (5.22) has its root in the analogue one that holds for the coordinate vectors,

$$\mathbf{k}_n^{\mathrm{MR}} = \begin{bmatrix} \mathbf{k}_{n-1}^{\mathrm{MR}} \\ \hline 0 \end{bmatrix} |s_n|^2 + \mathbf{k}_n^{\mathrm{G}} c_n^2,$$

which was given in Freund (1993, Lemma 4.1).

6. Variations of the Lanczos BiO algorithm

6.1. Further cases where the BiO *and* BiORes *algorithms simplify*

We have mentioned before that in the symmetric case the BiO algorithm simplifies: the left and the right Lanczos vectors coincide, and therefore only one matrix-vector product is needed per step. In fact, this simplification applies in a somewhat more general situation.

For every square matrix $\mathbf{A}$ there exists a nonsingular matrix $\mathbf{S}$ such that $\mathbf{A}^\top = \mathbf{S}\mathbf{A}\mathbf{S}^{-1}$, but, in general, the spectral decomposition of $\mathbf{A}$ is needed to construct $\mathbf{S}$, and thus $\mathbf{S}$ is normally not available. See, for instance, Horn and Johnson (1985, p. 134) for a proof of this result. Rutishauser (1953) and, later, Fletcher (1976) noticed that choosing $\widetilde{\mathbf{y}}_0 = \overline{\mathbf{S}\mathbf{y}_0}$ in the BiO algorithm yields $\widetilde{\mathbf{y}}_n = \overline{\mathbf{S}\mathbf{y}_n}$ $(n = 0, 1, \ldots, \nu - 1)$. (Rutishauser's and Fletcher's remarks are restricted to real matrices, but generalize in the way indicated above to the complex case.) Of course, it then suffices to generate $\{\mathbf{y}_n\}$ by the three-term recurrence, which means that the BiO algorithm becomes transpose-free and only one matrix-vector product involving $\mathbf{A}$ is needed per step. Hence, storage and work are then reduced to roughly

half, except for the additional multiplication by $\mathbf{S}$ for temporarily creating $\widetilde{\mathbf{y}}_n$, which appears in the inner products for α_n and δ_{temp}. Moreover, under these assumptions one-sided termination cannot happen: $\mathbf{y}_n = \mathbf{o}$ if and only if $\widetilde{\mathbf{y}}_n = \mathbf{o}$. However, serious breakdowns are in general still possible.

Fortunately, there are several interesting situations where the matrix $\mathbf{S}$ is known and is simple to multiply with. A trivial case is when $\mathbf{A} = \mathbf{A}^\top$ is symmetric (real or complex), and thus $\mathbf{S} = \mathbf{I}$. Freund (1994) lists several classes of $\mathbf{S}$-*symmetric* and $\mathbf{S}$-*Hermitian* matrices satisfying by definition $\mathbf{A}^\top\mathbf{S} = \mathbf{S}\mathbf{A}$, $\mathbf{S} = \mathbf{S}^\top$ and $\mathbf{A}^\star\mathbf{S} = \mathbf{S}\mathbf{A}$, $\mathbf{S} = \mathbf{S}^\star$, respectively. In particular, every Toeplitz matrix is $\mathbf{S}$-symmetric with $\mathbf{S}$ the antidiagonal unit matrix. Real Hamiltonian matrices multiplied by $i := \sqrt{-1}$ are also $\mathbf{S}$-symmetric. However, note that the conditions $\mathbf{S} = \mathbf{S}^\top$ or $\mathbf{S} = \mathbf{S}^\star$ are not needed for the simplification. Also, the class of $\mathbf{S}$-Hermitian matrices is rather restricted since any such matrix has a real spectrum.

Incidentally, the transformation $\mathbf{S}$ is also crucial in a paper of Jea and Young (1983, Def. 1.1, Thm 4.1).

6.2. The one-sided Lanczos algorithm

It has been pointed out by Saad (see Algorithm 3 in (Saad 1982)), that one can exploit additional freedom in choosing the sequence $\{\widetilde{\mathbf{y}}_n\}$ of left Lanczos vectors without affecting the sequence $\{\mathbf{y}_n\}$ of right Lanczos vectors: we can use for the former any sequence that spans the nested Krylov spaces $\widetilde{\mathcal{K}}_n$ successively. In fact, a closer look at our derivation in Section 2 shows that up to a scalar factor the right Lanczos vectors are fully determined by the orthogonality condition $\widetilde{\mathcal{K}}_n \perp_{\mathbf{B}} \mathbf{y}_n$. Therefore, it does not matter which set of nested bases is used for the left Krylov spaces $\widetilde{\mathcal{K}}_n$. All we need is that

$$\widetilde{\mathbf{y}}_n = \overline{t_n}(A^*)\widetilde{\mathbf{y}}_0$$

with a polynomial t_n of exact degree n or, equivalently, that for $\widetilde{\mathbf{y}}_n$ a recurrence holds that is of type (2.7) with $\widetilde{\tau}_{n,n-1} \neq 0$. Since this means giving up the mutual biorthogonality of the two vector sequences, we have to rederive the formulae for α_n and β_{n-1}. Again taking inner products of the first Lanczos recurrence in (2.19) with $\widetilde{\mathbf{y}}_{n-1}$ and $\widetilde{\mathbf{y}}_n$, we see that the formula (2.20) for β_{n-1} does not change, but the one for α_n changes into one of the following two:

$$\begin{aligned}\alpha_n &:= \langle\widetilde{\mathbf{y}}_n, \mathbf{A}\mathbf{y}_n - \mathbf{y}_{n-1}\beta_{n-1}\rangle_{\mathbf{B}} / \delta_n \\ &= \Big(\langle\widetilde{\mathbf{y}}_n, \mathbf{A}\mathbf{y}_n\rangle_{\mathbf{B}}\,\widetilde{\gamma}_{n-1} - \langle\widetilde{\mathbf{y}}_{n-1}, \mathbf{A}\mathbf{y}_{n-1}\rangle_{\mathbf{B}}\,\beta_{n-1} \\ &\qquad + \beta_{n-1}\delta_{n-1}\overline{\widetilde{\tau}_{n-1,n-1}}\Big) / (\widetilde{\gamma}_{n-1}\delta_n).\end{aligned}$$

These formulae, together with the standard recurrence for the right Lanczos vectors and a nearly arbitrary recurrence for the left Lanczos vectors leads

to Saad's variation of the BIO algorithm that we call here the *one-sided Lanczos algorithm*; in contrast to Saad we prefer the first formula for α_n, which is also used in Gutknecht and Ressel (1996).

ALGORITHM 5. (ONE-SIDED LANCZOS ALGORITHM)
Choose $\mathbf{y}_0, \widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle_{\mathbf{B}} \neq 0$, and set $\beta_{-1} := 0$. For $n = 0, 1, \ldots$ compute

$$\beta_{n-1} := \overline{\widetilde{\gamma}_{n-1}} \delta_n / \delta_{n-1}, \quad (\text{if } n > 0), \tag{6.1a}$$

$$\alpha_n := \langle \widetilde{\mathbf{y}}_n, \mathbf{A}\mathbf{y}_n - \mathbf{y}_{n-1}\beta_{n-1} \rangle_{\mathbf{B}} / \delta_n, \tag{6.1b}$$

$$\mathbf{y}_{\text{temp}} := \mathbf{A}\mathbf{y}_n - \mathbf{y}_n\alpha_n - \mathbf{y}_{n-1}\beta_{n-1}, \tag{6.1c}$$

$$\widetilde{\mathbf{y}}_{\text{temp}} := \mathbf{A}^\star \widetilde{\mathbf{y}}_n - \widetilde{\mathbf{y}}_n \widetilde{\tau}_{n,n} - \cdots - \widetilde{\mathbf{y}}_0 \widetilde{\tau}_{0,n}, \tag{6.1d}$$

$$\delta_{\text{temp}} := \langle \widetilde{\mathbf{y}}_{\text{temp}}, \mathbf{y}_{\text{temp}} \rangle_{\mathbf{B}}; \tag{6.1e}$$

if $\delta_{\text{temp}} = 0$, choose $\gamma_n \neq 0$ and $\widetilde{\gamma}_n \neq 0$, set

$$\nu := n + 1, \quad \mathbf{y}_\nu := \mathbf{y}_{\text{temp}} / \gamma_n, \quad \widetilde{\mathbf{y}}_\nu := \widetilde{\mathbf{y}}_{\text{temp}} / \widetilde{\gamma}_n, \quad \delta_{n+1} := 0,$$

and stop; otherwise, choose $\gamma_n \neq 0$, $\widetilde{\gamma}_n \neq 0$, and δ_{n+1} such that

$$\gamma_n \overline{\widetilde{\gamma}_n} \delta_{n+1} = \delta_{\text{temp}},$$

set

$$\mathbf{y}_{n+1} := \mathbf{y}_{\text{temp}} / \gamma_n, \quad \widetilde{\mathbf{y}}_{n+1} := \widetilde{\mathbf{y}}_{\text{temp}} / \widetilde{\gamma}_n,$$

and proceed with the next step.

Here, the coefficients $\widetilde{\tau}_{k,n}$ have been assumed to be given, but there are situations where one might want to determine them from recently computed right Lanczos vectors. It is easy to adapt this algorithm to the problem of solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ and to specify the resulting *one-sided consistent* BIORES *algorithm* with $\gamma_n := -\alpha_n - \beta_{n-1}$, or the *one-sided inconsistent* BIORES *algorithm*, or to combine it with the QMR approach.

Theoretically, one could use for the left sequence the Krylov vectors $\widetilde{\mathbf{y}}_n := (\mathbf{A}^\star)^n \widetilde{\mathbf{y}}_0$, thus simplifying (6.1d) to $\widetilde{\mathbf{y}}_{\text{temp}} := \mathbf{A}^\star \widetilde{\mathbf{y}}_n$, but in practice these soon become nearly multiples of each other (and of the eigenvector associated with the absolutely largest eigenvalue); therefore, even for moderate n, they are useless as a basis for $\widetilde{\mathcal{K}}_n$, and methods that rely on them do not work for most problems. On the other hand, to use a long recurrence for the left vectors would be a waste. Hence, only two-term or three-term recurrences are a serious option, and in most situations, the normal left Lanczos recurrences will be the best one.

Saad points out that this algorithm reminds us that the orthogonality of the left vectors, that is, $\widetilde{\mathbf{y}} \perp_{\mathbf{B}} \mathcal{K}_n$, is not essential, and that, therefore, in the BIO algorithm there is no reason to improve this orthogonality by reorthogonalization.

However, there are at least two situations where the one-sided Lanczos algorithm is valuable. First, a purely theoretical application is that it can serve as an intermediate step in the derivation of Lanczos-type product methods; see Section 16. Second, for the numerical stability of the Lanczos process it is most important that the space-expanding term in the recursion for the Lanczos vectors, that is, $\mathbf{A}\mathbf{y}$ or $\mathbf{A}^\star\tilde{\mathbf{y}}$, respectively, is not too small compared to the two other terms. If this happens to the right sequence, we have to switch to a look-ahead step (see Section 19). However, if only the left sequence is affected, we can just switch to the one-sided Lanczos algorithm instead.

In fact, for numerical stability, the optimal choice for the left sequence would be the one generated by the Arnoldi process applied to $\mathbf{A}^\star$ with starting vector $\tilde{\mathbf{y}}_0$. However, the cost of this process forbids this: recall that the main advantage of the Lanczos process over the Arnoldi process is the large reduction of memory and computational costs. As a cost-effective compromise we may choose the ORTHORES(2) process instead, which amounts to making $\tilde{\mathbf{y}}_{\text{temp}}$ orthogonal to $\tilde{\mathbf{y}}_n$ and $\tilde{\mathbf{y}}_{n-1}$ (instead of $\mathbf{y}_n$ and $\mathbf{y}_{n-1}$):

$$\tilde{\mathbf{y}}_{\text{temp}} := \mathbf{A}^\star\tilde{\mathbf{y}}_n - \tilde{\mathbf{y}}_n\langle\tilde{\mathbf{y}}_n, \mathbf{A}^\star\tilde{\mathbf{y}}_n\rangle_{\mathbf{B}} - \tilde{\mathbf{y}}_{n-1}\langle\tilde{\mathbf{y}}_{n-1}, \mathbf{A}^\star\tilde{\mathbf{y}}_n\rangle_{\mathbf{B}},$$

except that the last term does not exist when $n = 1$. However, it is better to implement this according to the modified Gram–Schmidt process, and thus compute

$$\begin{aligned}
\tilde{\mathbf{y}}_{\text{temp}} &:= \mathbf{A}^\star\tilde{\mathbf{y}}_n, \\
\tilde{\mathbf{y}}_{\text{temp}} &:= \tilde{\mathbf{y}}_{\text{temp}} - \tilde{\mathbf{y}}_n\langle\tilde{\mathbf{y}}_n, \tilde{\mathbf{y}}_{\text{temp}}\rangle_{\mathbf{B}}, \\
\tilde{\mathbf{y}}_{\text{temp}} &:= \tilde{\mathbf{y}}_{\text{temp}} - \tilde{\mathbf{y}}_{n-1}\langle\tilde{\mathbf{y}}_{n-1}, \tilde{\mathbf{y}}_{\text{temp}}\rangle_{\mathbf{B}}, \quad (\text{if } n > 0).
\end{aligned}$$

6.3. An abstract setting for the Lanczos process

We have introduced the Lanczos process for a real or complex $N \times N$ matrix $\mathbf{A}$. Such a matrix can always be thought of as a linear operator $A : \mathbb{R}^N \to \mathbb{R}^N$ or $A : \mathbb{C}^N \to \mathbb{C}^N$, respectively. For generality, let us concentrate on the complex case: the Euclidean space $\mathbb{C}^N$ is a finite-dimensional inner product space, and we have made use of its inner product $\langle .,. \rangle$ when defining the formal inner product $\langle\tilde{\mathbf{y}}, \mathbf{y}\rangle_{\mathbf{B}} := \langle\tilde{\mathbf{y}}, \mathbf{B}\mathbf{y}\rangle = \tilde{\mathbf{y}}^\star\mathbf{B}\mathbf{y}$, which involves a matrix $\mathbf{B}$ that commutes with $\mathbf{A}$ and is in most applications just the identity $\mathbf{I}$. The norm that comes with $\langle .,. \rangle$ was occasionally used to normalize vectors, but, as we have seen, the Lanczos process can work with unnormalized vectors, and thus the norm is only needed as soon as we want to measure convergence.

It is straightforward to reformulate the Lanczos process for an infinite-dimensional Hilbert space, and there are indeed applications for this setting; see, for instance, Hayes (1954), Kreuzer, Miller and Berger (1981) and Lanczos (1950). However, as pointed out by Parlett (1992) (and further

developed in a private discussion), one can go a big step futher with respect to generality.

Parlett makes the point that the Lanczos algorithm can be defined in a plain vector space. There is no need to normalize the Lanczos vectors because they can be defined by the monic Lanczos polynomials. There is no need for $\mathbf{A}^\star$ or $\mathbf{A}^\top$ provided that one set of Lanczos vectors consists of row vectors and the other consists of column vectors. There is no need for a normed space, let alone an inner product space. The point is that there is no norm or inner product natural to the Lanczos algorithm. Different applications might require different norms. The choice of norm or inner product needs to be justified, not assumed.

Let $\mathcal{V}$ be a linear space over the field $\mathbb{C}$ (for simplicity), and $A : \mathcal{V} \to \mathcal{V}$ a linear operator. The linear functionals defined on $\mathcal{V}$ form another linear space, the *algebraic dual space* $\mathcal{V}^\times$ of $\mathcal{V}$; see, for instance, Kreyszig (1978, Section 2.8-8). A linear operator $A^\times : \mathcal{V}^\times \to \mathcal{V}^\times$ *adjoint* to A can be defined by

$$(A^\times \tilde{y})(y) := \tilde{y}(Ay) \quad (\text{for all } y \in \mathcal{V}, \tilde{y} \in \mathcal{V}^\times).$$

If $\mathcal{V}$ is in addition normed, one considers typically the *normed dual space* $\mathcal{V}'$ of $\mathcal{V}$ consisting only of the linear functionals that are bounded. Then, if A is bounded, the restriction of $A^\times$ to $\mathcal{V}$ becomes a bounded linear operator, the *adjoint* A' of A on $\mathcal{V}'$. It has the same (operator) norm as A; see, for instance, Kreyszig (1978, Section 4.5-2), Rudin (1973, pp. 92–93).

For either of these two situations we can define the Lanczos process if we replace '$\tilde{\mathbf{y}} \in \mathbb{C}^N$ is ($\mathbf{B}$-)orthogonal to $\mathbf{y} \in \mathbb{C}^N$' by '$y \in \mathcal{V}$ is a zero of $\tilde{y} \in \mathcal{V}^\times$ [or: $\mathcal{V}'$]':

$$\langle \tilde{\mathbf{y}}, \mathbf{y} \rangle_{\mathbf{B}} = 0 \quad \rightsquigarrow \quad \tilde{y}(y) = 0$$

In particular, instead of ($\mathbf{B}$-)biorthogonal bases satisfying $\langle \tilde{\mathbf{y}}_m, \mathbf{y}_n \rangle_{\mathbf{B}} = \delta_{m,n}\delta_n$, the Lanczos process then produces *dual bases* satisfying

$$\tilde{y}_m(y_n) = \delta_{m,n}\delta_n.$$

We need, however, to point out a difference between this setting and the one in a complex Hilbert space, in particular $\mathbb{C}^N$. The inner product $\langle \tilde{\mathbf{y}}, \mathbf{y} \rangle$ in $\mathbb{C}^N$ is sesquilinear, while $\tilde{y}(y)$ is bilinear. Therefore, the above defined adjoint operators $A^\times$ and A' are not identical with the Hermitian transpose $\mathbf{A}^\star$ if $\mathcal{V} = \mathbb{C}^N$ (or with the Hilbert space adjoint if $\mathcal{V}$ is a Hilbert space), but rather with the transpose $\mathbf{A}^\top$ of $\mathbf{A}$; see, for instance, Kreyszig (1978, Section 4.5-3), Rudin (1973, pp. 297–298). For this reason, the formulae of our algorithms require some small modifications if translated into the abstract setting of dual spaces.

7. Coupled recurrences: the BIOC algorithm

In his second paper on the subject, Lanczos (1952) suggested under the section heading 'The complete algorithm for minimized iterations' an alternative algorithm for computing the sequences $\{\mathbf{y}_n\}$, $\{\tilde{\mathbf{y}}_n\}$ generated by the BIO algorithm. He also discussed in detail how to apply this algorithm for solving linear systems of equations. While the BIO algorithm for the nonsymmetric Lanczos process described in Section 2 is based on a three-term recurrence we turn now to another algorithm based on a coupled pair of two-term recurrences for the same process. The relationship between the two types of recurrence is the same as that between a linear second-order ordinary differential equation and an equivalent pair of coupled first-order equations: we just introduce an auxiliary quantity. In addition to the pair of biorthogonal (*i.e.*, $\mathbf{B}$-biorthogonal) Krylov space bases, a second pair of biconjugate (*i.e.*, $\mathbf{BA}$-biorthogonal) bases for the same Krylov space is now generated. That is why we introduce here the acronym BIOC for this algorithm.

While the BIO algorithm supplies us with a tridiagonal matrix $\mathbf{T}$ that represents a projection of $\mathbf{A}$, the new algorithm produces the two bidiagonal matrices that are the LU-factors of $\mathbf{T}$. If an LU-factorization (without pivoting) of $\mathbf{T}$ does not exist, the BIOC algorithm breaks down early. This seems to be a disadvantage of the Lanczos process based on coupled two-term recurrences (BIOC algorithm) compared to the one based on three-term recurrences (BIO algorithm). However, practice shows that nevertheless the BIOC algorithm is often numerically preferable, as round-off seems to have less impact.

The usual application of the BIOC algorithm is to solve a linear system of equations $\mathbf{Ax} = \mathbf{b}$, again either by additionally computing iterates that satisfy the Petrov–Galerkin condition of the BICG method, or by solving the least squares problem in coordinate space of the QMR method. Further investigations are necessary to find out if the BIOC algorithm is also advisable for eigenvalue computations, where it has hardly ever been applied until now. Parlett (1995) lists several advantages of the factored form. In particular, if we assume that $\mathbf{T}_n$ and its LU-factors are known to a certain precision, then the factors implicitly determine the entries of $\mathbf{T}_n$ to higher precision. The factors are also the input data of Rutishauser's differential QD algorithm of 1970 (see Rutishauser (1990)), which has recently been enhanced by Fernando and Parlett (1994). Enriched by a suitable shift strategy, it has become the method of choice for the bidiagonal singular value problem and the eigenvalue problem of a real symmetric positive definite tridiagonal matrix. By avoiding explicit shifts, it can be made competitive to the QR algorithm even for the general real symmetric tridiagonal eigenvalue problem. Making its nonsymmetric version sufficiently stable seems

to be a long way ahead, however. It would be less expensive than the normally applied QR algorithm, as it works with bidiagonal matrices, while QR transforms the nonsymmetric tridiagonal into an upper Hessenberg matrix.

7.1. *The* BIOC *algorithm*

We start with the formulation of the BIOC algorithm and a discussion of its main properties.

ALGORITHM 6. (BIOC ALGORITHM)
Choose $\mathbf{y}_0, \widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle_{\mathbf{B}} \neq 0$ and $\delta_0' := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\mathbf{y}_0 \rangle_{\mathbf{B}} \neq 0$, and set $\mathbf{v}_0 := \mathbf{y}_0$, $\widetilde{\mathbf{v}}_0 := \widetilde{\mathbf{y}}_0$. For $n = 0, 1, \ldots$, choose $\gamma_n \neq 0$, $\widetilde{\gamma}_n \neq 0$ and compute

$$\begin{aligned}
\varphi_n &:= \delta_n'/\delta_n, && (7.1a)\\
\widetilde{\varphi}_n &:= \overline{\varphi_n}, && (7.1b)\\
\mathbf{y}_{n+1} &:= (\mathbf{A}\mathbf{v}_n - \mathbf{y}_n\varphi_n)/\gamma_n, && (7.1c)\\
\widetilde{\mathbf{y}}_{n+1} &:= (\mathbf{A}^\star\widetilde{\mathbf{v}}_n - \widetilde{\mathbf{y}}_n\widetilde{\varphi}_n)/\widetilde{\gamma}_n, && (7.1d)\\
\delta_{n+1} &:= \langle \widetilde{\mathbf{y}}_{n+1}, \mathbf{y}_{n+1} \rangle_{\mathbf{B}}\,; && (7.1e)\\
\psi_n &:= \overline{\widetilde{\gamma}_n}\delta_{n+1}/\delta_n', && (7.1f)\\
\widetilde{\psi}_n &:= \overline{\gamma_n\delta_{n+1}/\delta_n'}, && (7.1g)\\
\mathbf{v}_{n+1} &:= \mathbf{y}_{n+1} - \mathbf{v}_n\psi_n, && (7.1h)\\
\widetilde{\mathbf{v}}_{n+1} &:= \widetilde{\mathbf{y}}_{n+1} - \widetilde{\mathbf{v}}_n\widetilde{\psi}_n, && (7.1i)\\
\delta_{n+1}' &:= \langle \widetilde{\mathbf{v}}_{n+1}, \mathbf{A}\mathbf{v}_{n+1} \rangle_{\mathbf{B}}\,. && (7.1j)
\end{aligned}$$

If $\delta_{n+1} = 0$ or $\delta_{n+1}' = 0$, set $\dot{\nu} := n+1$ and stop; otherwise proceed with the next step.

The formulae (7.1c)–(7.1d) and (7.1h)–(7.1i) are known as *coupled two-term recurrences*. We will see below that by eliminating $\mathbf{v}_n$ and $\widetilde{\mathbf{v}}_m$ from them we get back to the three-term recurrences (2.19) of the BIO algorithm.

The basic result for this BIOC algorithm is the following one.

Theorem 7.1 The sequences $\{\mathbf{y}_n\}_{n=0}^{\dot{\nu}}$, $\{\widetilde{\mathbf{y}}_n\}_{n=0}^{\dot{\nu}}$ generated by the BIOC algorithm are biorthogonal, and the sequences $\{\mathbf{v}_n\}_{n=0}^{\dot{\nu}}$ and $\{\widetilde{\mathbf{v}}_n\}_{n=0}^{\dot{\nu}}$ are biconjugate (with respect to $\mathbf{A}$) except that $\langle \widetilde{\mathbf{y}}_{\dot{\nu}}, \mathbf{y}_{\dot{\nu}} \rangle_{\mathbf{B}} = 0$ or $\langle \widetilde{\mathbf{v}}_{\dot{\nu}}, \mathbf{A}\mathbf{v}_{\dot{\nu}} \rangle_{\mathbf{B}} = 0$. That is, for $m, n = 0, 1, \ldots, \dot{\nu}$,

$$\langle \widetilde{\mathbf{y}}_m, \mathbf{y}_n \rangle_{\mathbf{B}} = \begin{cases} 0, & m \neq n, \\ \delta_n, & m = n, \end{cases} \tag{7.2}$$

$$\langle \widetilde{\mathbf{v}}_m, \mathbf{A}\mathbf{v}_n \rangle_{\mathbf{B}} = \begin{cases} 0\,, & m \neq n, \\ \delta_n' = \delta_n\varphi_n\,, & m = n, \end{cases} \tag{7.3}$$

where $\delta_n \neq 0$ and $\delta'_n \neq 0$ for $0 \leq n \leq \dot{\nu} - 1$, but $\delta_{\dot{\nu}} = 0$ or $\delta'_{\dot{\nu}} = 0$. Moreover, for $n = 1, \ldots, \dot{\nu} - 1$ holds in addition to (2.5)

$$\mathbf{v}_n \in \mathcal{K}_{n+1} \backslash \mathcal{K}_n \,, \quad \tilde{\mathbf{v}}_n \in \tilde{\mathcal{K}}_{n+1} \backslash \tilde{\mathcal{K}}_n. \tag{7.4}$$

Proof. We provide an adaptation of Fletcher's proof (Fletcher 1976) to the complex case and to our adjustable normalization. For $m = n = 0$, (7.2)–(7.4) and (2.5) clearly hold. Assume that they hold for $m, n = 0, \ldots, k$ $(< \dot{\nu})$. From (7.1c) and (7.1i) we get

$$\begin{aligned} &\langle \tilde{\mathbf{y}}_m, \mathbf{y}_{k+1} \rangle_{\mathbf{B}} \qquad (7.5) \\ &\quad = \langle \tilde{\mathbf{y}}_m,\, \mathbf{A}\mathbf{v}_k - \mathbf{y}_k \varphi_k \rangle_{\mathbf{B}} \,/\gamma_k \\ &\quad = \left(\langle \tilde{\mathbf{v}}_m,\, \mathbf{A}\mathbf{v}_k \rangle_{\mathbf{B}} + \overline{\tilde{\psi}_{m-1}} \langle \tilde{\mathbf{v}}_{m-1},\, \mathbf{A}\mathbf{v}_k \rangle_{\mathbf{B}} - \langle \tilde{\mathbf{y}}_m,\, \mathbf{y}_k \rangle_{\mathbf{B}}\, \varphi_k \right) / \gamma_k. \qquad (7.6) \end{aligned}$$

Here, for $m \leq k - 1$, all terms are zero by assumption. On the other hand, if $m = k$, (7.1a), (7.1e), and (7.1j) inserted into (7.6) yield $\langle \tilde{\mathbf{y}}_k, \mathbf{y}_{k+1} \rangle_{\mathbf{B}} = (\varphi_k \delta_k + 0 - \varphi_k \delta_k)/\gamma_k = 0$; hence, $\langle \tilde{\mathbf{y}}_m, \mathbf{y}_{k+1} \rangle_{\mathbf{B}} = 0$ for $m \leq k$. By symmetry, $\langle \tilde{\mathbf{y}}_{k+1}, \mathbf{y}_m \rangle_{\mathbf{B}} = 0$ for $m \leq k$ too, and together with (7.1e) it follows that (7.2) holds up to $k + 1$.

Similarly, using (7.1h) and (7.1d) we get

$$\begin{aligned} &\langle \tilde{\mathbf{v}}_m, \mathbf{A}\mathbf{v}_{k+1} \rangle_{\mathbf{B}} \\ &\quad = \langle \tilde{\mathbf{v}}_m,\, \mathbf{A}(\mathbf{y}_{k+1} - \mathbf{v}_k \psi_k) \rangle_{\mathbf{B}} \\ &\quad = \overline{\tilde{\gamma}_m} \langle \tilde{\mathbf{y}}_{m+1},\, \mathbf{y}_{k+1} \rangle_{\mathbf{B}} + \overline{\tilde{\varphi}_m} \langle \tilde{\mathbf{y}}_m,\, \mathbf{y}_{k+1} \rangle_{\mathbf{B}} - \langle \tilde{\mathbf{v}}_m,\, \mathbf{A}\mathbf{v}_k \rangle_{\mathbf{B}}\, \psi_k. \end{aligned}$$

Here, too, for $m \leq k - 1$, all terms are zero by assumption. If $m = k$, we see from (7.1h), (7.1d), (7.2), (7.3), and (7.1f) that $\langle \tilde{\mathbf{v}}_k, \mathbf{A}\mathbf{v}_{k+1} \rangle_{\mathbf{B}} = \overline{\tilde{\gamma}_k} \delta_{k+1} + 0 - \delta'_k \psi_k = 0$. Hence, $\langle \tilde{\mathbf{v}}_m, \mathbf{A}\mathbf{v}_{k+1} \rangle_{\mathbf{B}} = 0$ for $m \leq k$, and by symmetry $\langle \tilde{\mathbf{v}}_{k+1}, \mathbf{A}\mathbf{v}_m \rangle_{\mathbf{B}} = 0$ too. Finally, the equation in (7.3) for $m = n = k + 1$ results from (7.1a).

The formulae (7.1c), (7.1d), (7.1h), (7.1i), (2.5), and (7.4) show clearly that $\mathbf{y}_{k+1}, \mathbf{v}_{k+1} \in \mathcal{K}_{k+2}$ and $\tilde{\mathbf{y}}_{k+1}, \tilde{\mathbf{v}}_{k+1} \in \tilde{\mathcal{K}}_{k+2}$. As in Section 2, $\delta_n \neq 0$ implies that $\mathbf{y}_{k+1} \notin \mathcal{K}_{k+1}$ and $\tilde{\mathbf{y}}_{k+1} \notin \tilde{\mathcal{K}}_{k+1}$. By (7.1h) and (7.1i) it thus follows that (7.4) holds for $n = k + 1$. This completes the induction. □

7.2. Matrix relations

The BIOC algorithm has a matrix interpretation, which quickly reveals the relation to the BIO algorithm. In addition to $\mathbf{Y}_n$ and $\tilde{\mathbf{Y}}_n$ of (2.8) we need the $N \times n$ matrices

$$\mathbf{V}_n := [\ \mathbf{v}_0 \quad \mathbf{v}_1 \quad \cdots \quad \mathbf{v}_{n-1}\ ], \quad \tilde{\mathbf{V}}_n := [\ \tilde{\mathbf{v}}_0 \quad \tilde{\mathbf{v}}_1 \quad \cdots \quad \tilde{\mathbf{v}}_{n-1}\ ],$$

the $n \times n$ matrices

$$\mathbf{L}_n := \begin{bmatrix} \varphi_0 & & & & \\ \gamma_0 & \varphi_1 & & & \\ & \gamma_1 & \varphi_2 & & \\ & & \ddots & \ddots & \\ & & & \gamma_{n-2} & \varphi_{n-1} \end{bmatrix}, \quad \mathbf{U}_n := \begin{bmatrix} 1 & \psi_0 & & & \\ & 1 & \psi_1 & & \\ & & 1 & \ddots & \\ & & & \ddots & \psi_{n-2} \\ & & & & 1 \end{bmatrix},$$

which are lower and upper bidiagonal, respectively, and the *extended* bidiagonal matrices

$$\underline{\mathbf{L}}_n := \left[\begin{array}{c} \mathbf{L}_n \\ \hline \gamma_{n-1}\mathbf{l}_n^\top \end{array}\right] = \begin{bmatrix} \varphi_0 & & & & \\ \gamma_0 & \varphi_1 & & & \\ & \gamma_1 & \varphi_2 & & \\ & & \ddots & \ddots & \\ & & & \gamma_{n-2} & \varphi_{n-1} \\ & & & & \gamma_{n-1} \end{bmatrix}$$

and

$$\mathbf{U}\!\mid_n := \left[\begin{array}{c|c} \mathbf{U}_n & \mathbf{l}_n\psi_{n-1} \end{array}\right] = \begin{bmatrix} 1 & \psi_0 & & & & \\ & 1 & \psi_1 & & & \\ & & 1 & \ddots & & \\ & & & \ddots & \psi_{n-2} & \\ & & & & 1 & \psi_{n-1} \end{bmatrix},$$

with an additional row and column, respectively[13]. Analogously, we define $\widetilde{\mathbf{L}}_n$, $\widetilde{\mathbf{U}}_n$, $\underline{\widetilde{\mathbf{L}}}_n$, and $\widetilde{\mathbf{U}\!\mid}_n$ in the obvious way. Then, according to (7.1h), (7.1i), (7.1c), and (7.1d),

$$\mathbf{Y}_n = \mathbf{V}_n\mathbf{U}_n, \quad \widetilde{\mathbf{Y}}_n = \widetilde{\mathbf{V}}_n\widetilde{\mathbf{U}}_n \quad (n \le \dot{\nu})$$

and

$$\mathbf{A}\mathbf{V}_n = \mathbf{Y}_{n+1}\underline{\mathbf{L}}_n, \quad \mathbf{A}^\star\widetilde{\mathbf{V}}_n = \widetilde{\mathbf{Y}}_{n+1}\underline{\widetilde{\mathbf{L}}}_n \quad (n \le \dot{\nu}).$$

After setting[14]

$$\mathbf{T}_n := \mathbf{L}_n\mathbf{U}_n, \quad \mathbf{T}'_n := \mathbf{U}\!\mid_n\underline{\mathbf{L}}_n \tag{7.7}$$

and

$$\underline{\mathbf{T}}_n := \underline{\mathbf{L}}_n\mathbf{U}_n, \quad \underline{\mathbf{T}}'_n := \mathbf{U}_{n+1}\underline{\mathbf{L}}_n, \tag{7.8}$$

[13] Note that the additional column is indicated by a vertical line. We suggest reading this symbol as 'U sub n extended'.

[14] The prime does *not* mean transposition.

and likewise defining $\widetilde{\mathbf{T}}_n$, $\widetilde{\mathbf{T}}'_n$, $\underline{\widetilde{\mathbf{T}}}_n$, and $\underline{\widetilde{\mathbf{T}}}'_n$, we conclude by eliminating $\mathbf{V}_n$ and $\widetilde{\mathbf{V}}_n$ or $\mathbf{Y}_n$ and $\widetilde{\mathbf{Y}}_n$, respectively, that

$$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n, \quad \mathbf{A}^\star\widetilde{\mathbf{Y}}_n = \widetilde{\mathbf{Y}}_{n+1}\underline{\widetilde{\mathbf{T}}}_n \tag{7.9}$$

and

$$\mathbf{A}\mathbf{V}_n = \mathbf{V}_{n+1}\underline{\mathbf{T}}'_n, \quad \mathbf{A}^\star\widetilde{\mathbf{V}}_n = \widetilde{\mathbf{V}}_{n+1}\underline{\widetilde{\mathbf{T}}}'_n. \tag{7.10}$$

Note that $\mathbf{T}'_n = \mathbf{U}\!\!\!\!\mathbf{l}_n\underline{\mathbf{L}}_n$ and $\mathbf{U}_n\mathbf{L}_n$ differ only by the rank-one matrix $\mathbf{l}_n\gamma_{n-1}\psi_{n-1}\mathbf{l}_n^\top$, which just modifies the element in the lower right corner of $\mathbf{U}_n\mathbf{L}_n$ by adding $\gamma_{n-1}\psi_{n-1}$ to it.

Since the matrix $\mathbf{T}_n$ with the recurrence coefficients α_m, β_m, γ_m of the BIO algorithm, and the matrices $\mathbf{L}_n$ and $\mathbf{U}_n$ with the recurrence coefficients φ_n and ψ_{n-1} of the BIOC algorithm, as well as the tridiagonal matrix $\mathbf{T}'_n$ with elements α'_m, β'_m, γ'_m are related by (7.7), these three sets of parameters are easily converted into each other. After setting $\psi_{-1} := 0$, we obtain

(i) from $\mathbf{T}_{\dot\nu} = \mathbf{L}_{\dot\nu}\mathbf{U}_{\dot\nu}$:

$$\alpha_n = \varphi_n + \gamma_{n-1}\psi_{n-1}, \quad \beta_n = \varphi_n\psi_n, \quad \gamma_n = \widetilde{\gamma}_n,$$

(ii) from $\mathbf{T}'_{\dot\nu} = \mathbf{U}\!\!\!\!\mathbf{l}_{\dot\nu}\underline{\mathbf{L}}_{\dot\nu}$:

$$\alpha'_n = \varphi_n + \gamma_n\psi_n, \quad \beta'_n = \varphi_{n+1}\psi_n, \quad \widetilde{\gamma}_n = \gamma_n.$$

These are essentially the *rhombus rules* of the QD *algorithm* (Rutishauser 1957). Of course, the same formulae also hold with tildes.

Since (7.9) is identical to (2.23), except for the possibility that $\dot\nu < \nu$, we obtain the following result.

Theorem 7.2 If the same starting vectors $\mathbf{y}_0$ and $\widetilde{\mathbf{y}}_0$ and the same scale factors γ_n and $\widetilde{\gamma}_n$ are chosen in the BIO and the BIOC algorithms, then the same biorthogonal vector sequences $\{\mathbf{y}_n\}$ and $\{\widetilde{\mathbf{y}}_n\}$ are produced, except that the BIOC algorithm may break down earlier due to $\delta'_{\dot\nu} = 0$. The bidiagonal matrices $\mathbf{L}_{\dot\nu}$, $\widetilde{\mathbf{L}}_{\dot\nu}$, $\mathbf{U}_{\dot\nu}$, and $\widetilde{\mathbf{U}}_{\dot\nu}$ of the recurrence coefficients of the BIOC algorithm can be obtained by LU decomposition of the tridiagonal matrix $\mathbf{T}_{\dot\nu}$ with the recurrence coefficients of the first $\dot\nu$ steps of the BIO algorithm. The possible earlier breakdown of the BIOC algorithm is due to the possible nonexistence of the LU decomposition (without pivoting) of $\mathbf{T}_\nu$.

This result implies in particular that from the bidiagonal matrices constructed in the BIOC algorithm we can still compute the eigenvalues of $\mathbf{T}_n$, the so-called Petrov values (or, Ritz values in the Hermitian case), as approximations for eigenvalues of $\mathbf{A}$. As mentioned at the beginning of this section, there are a number of reasons why the bidiagonal matrices are preferable; see Parlett (1995).

Theorem 7.2 holds analogously for the ORTHORES and ORTHOMIN versions of the generalized CG and the GCR methods, except that $\mathbf{T}_\nu$ is then upper Hessenberg and $\mathbf{U}_\nu$ is upper triangular, while $\mathbf{L}_\nu$ is still lower bidiagonal; see Gutknecht (1993*a*). There is also a similar result that links ORTHOMIN with ORTHODIR. We will encounter its analogue in Section 9.

7.3. Normalization; simplification due to symmetry

For the BIO algorithm (Algorithm 1) we have chosen a somewhat complicated formulation to make explicit the freedom in choosing two of the three quantities γ_n, $\widetilde{\gamma}_n$, and δ_{n+1}. The same freedom exists in our formulation of BIOC, although we have not made that explicit. In particular, as normalization we can still enforce (2.26) or (2.27). In these two cases, it is necessary to define in a straightforward manner $\mathbf{y}_{\text{temp}}$, $\widetilde{\mathbf{y}}_{\text{temp}}$, and δ_{temp}, as in Algorithm 1. The second choice, (2.27), will lead to Lanczos vectors of length 1. However, if we also wanted to have normalized direction vectors, we would have to introduce additional scale factors in (7.1h) and (7.1i), and to compensate for them in some of the other formulae.

What has been said in Sections 2.1 and 2.6 regarding simplification due to symmetry also carries over to the BIOC algorithm. In particular, if $\mathbf{A}$ is Hermitian, complex symmetric, $\mathbf{S}$-Hermitian, or $\mathbf{S}$-symmetric, then the matrix-vector multiplication by $\mathbf{A}^\star$ can be replaced by multiplication by $\mathbf{S}$.

8. The BIOMIN form of the BICG method

A consistent version of the BICG method based on the BIOC algorithm was presented by Fletcher (1976). He referred to it as the *biconjugate gradient* (BICG) *algorithm*, while later, Jea and Young (1983) called it Lanczos/ORTHOMIN. Here we use the name BIOMIN in order to stress the analogy to the OMIN (Hestenes–Stiefel) version of the conjugate gradient (CG) method (Hestenes and Stiefel 1952) and the differently flavoured analogy to BIORES and BIODIR. The latter is discussed later. The BIOC algorithm is related to the BIOMIN version of BICG and to the OMIN version of CG in the same way as the Lanczos BIO algorithm is related to the BIORES version of BICG and to the ORES version of CG.

We will refer to BIOMIN also as the *standard version* of BICG. We keep using the abbreviation BICG (like CG) as the generic name for the various biconjugate gradient algorithms that are mathematically equivalent except for possible deviations in the breakdown conditions.

8.1. The BIOMIN *algorithm*

When applying the BIO algorithm to solving linear systems in such a way that the right Lanczos vectors became the residuals, we had to stick to a

particular choice of γ_n, namely $\gamma_n := -\alpha_n - \beta_{n-1}$, in order to fulfil the consistency condition for Krylov space solvers. Likewise, γ_n is determined here by the latter condition. In fact, since $\underline{\mathbf{L}}_n = \underline{\mathbf{T}}_n \mathbf{U}_n^{-1}$, the bidiagonal matrix $\underline{\mathbf{L}}_n$ inherits from $\underline{\mathbf{T}}_n$ the property of zero column sums, which, as we recall, was inherited to $\underline{\mathbf{T}}_n$ from $\underline{\mathbf{F}}_n$; that is, we need

$$\mathbf{e}^\top \underline{\mathbf{L}}_n = \mathbf{o}^\top,$$

or, in terms of matrix elements,

$$\gamma_n := -\varphi_n. \tag{8.1}$$

Again, we can then define the iterates $\mathbf{x}_n$ in such a way that $\mathbf{y}_n$ is the nth residual: multiplicaton of (4.14) from the right by $\mathbf{U}_n^{-1}$ yields

$$\mathbf{V}_n = -\mathbf{X}_{n+1} \underline{\mathbf{L}}_n \quad (n \le \dot{\nu}),$$

or,

$$\mathbf{x}_{n+1} := \mathbf{x}_n + \mathbf{v}_n \omega_n$$

if we set

$$\omega_n := \frac{1}{\varphi_n} = -\frac{1}{\gamma_n}. \tag{8.2}$$

Basically we are free to choose $\widetilde{\gamma}_n$, but the normal choice is $\widetilde{\gamma}_n := \overline{\gamma_n}$, which implies that $\widetilde{\psi}_n := \overline{\psi_n}$ and $\psi_n := -\delta_{n+1}/\delta_n$, see (7.1f)–(7.1g). This leads us to the standard BIOMIN form of the BICG method.

ALGORITHM 7. (BIOMIN FORM OF THE BICG METHOD)
For solving $\mathbf{Ax} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0$, set $\mathbf{v}_0 := \mathbf{y}_0 := \mathbf{b} - \mathbf{Ax}_0$, and choose $\widetilde{\mathbf{y}}_0$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{y}_0 \rangle_{\mathbf{B}} \neq 0$ and $\delta_0' := \langle \widetilde{\mathbf{y}}_0, \mathbf{Av}_0 \rangle_{\mathbf{B}} \neq 0$. Then let $\widetilde{\mathbf{v}}_0 := \widetilde{\mathbf{y}}_0$, and apply Algorithm 6 (BIOC) with $\gamma_n := -\varphi_n$ and $\widetilde{\gamma}_n := -\overline{\varphi_n}$, so that after substituting $\omega_n := 1/\varphi_n$ the nth step consists of:

$$\begin{aligned}
\omega_n &:= \delta_n/\delta_n', && (8.3a)\\
\mathbf{y}_{n+1} &:= \mathbf{y}_n - \mathbf{Av}_n \omega_n, && (8.3b)\\
\widetilde{\mathbf{y}}_{n+1} &:= \widetilde{\mathbf{y}}_n - \mathbf{A}^\star \widetilde{\mathbf{v}}_n \overline{\omega_n}, && (8.3c)\\
\mathbf{x}_{n+1} &:= \mathbf{x}_n + \mathbf{v}_n \omega_n, && (8.3d)\\
\delta_{n+1} &:= \langle \widetilde{\mathbf{y}}_{n+1}, \mathbf{y}_{n+1} \rangle_{\mathbf{B}}\,, && (8.3e)\\
\psi_n &:= -\delta_{n+1}/\delta_n, && (8.3f)\\
\mathbf{v}_{n+1} &:= \mathbf{y}_{n+1} - \mathbf{v}_n \psi_n, && (8.3g)\\
\widetilde{\mathbf{v}}_{n+1} &:= \widetilde{\mathbf{y}}_{n+1} - \widetilde{\mathbf{v}}_n \overline{\psi_n}, && (8.3h)\\
\delta_{n+1}' &:= \langle \widetilde{\mathbf{v}}_{n+1}, \mathbf{Av}_{n+1} \rangle_{\mathbf{B}}\,. && (8.3i)
\end{aligned}$$

If $\mathbf{y}_{n+1} = 0$ the process terminates and $\mathbf{x}_{n+1}$ is the solution; if $\delta_{n+1} = 0$ (and hence $\psi_n = 0$) or $\delta_{n+1}' = 0$, but $\mathbf{y}_{n+1} \neq 0$, the algorithm breaks down. In all cases we set $\dot{\nu} := n + 1$.

Assuming $\mathbf{b} - \mathbf{A}\mathbf{x}_n = \mathbf{y}_n$ and using (8.3d) and (8.3b) we in fact get

$$\mathbf{b} - \mathbf{A}\mathbf{x}_{n+1} = \mathbf{b} - \mathbf{A}\mathbf{x}_n - \mathbf{A}\mathbf{v}_n\omega_n = \mathbf{y}_n - \mathbf{A}\mathbf{v}_n\omega_n = \mathbf{y}_{n+1},$$

so that by induction $\mathbf{b} - \mathbf{A}\mathbf{x}_n = \mathbf{y}_n$ for $n = 0, 1, \ldots, \dot{\nu}$. Consequently, if $\mathbf{y}_{\dot{\nu}} = 0$, then $\mathbf{x}_{\dot{\nu}}$ is the solution of the system.

Note that by definition of $\dot{\nu}$ we have

$$\delta_n \neq 0, \quad \psi_{n-1} \neq 0, \quad \varphi_n \neq 0, \quad \omega_n \neq 0, \quad n = (0), 1, \ldots, \dot{\nu} - 1, \tag{8.4a}$$

and one of the following three cases:

$$\delta_{\dot{\nu}} \neq 0, \quad \delta'_{\dot{\nu}} = 0 \quad \Longrightarrow \quad \psi_{\dot{\nu}-1} \neq 0, \quad \varphi_{\dot{\nu}} = 0, \quad \omega_{\dot{\nu}} = \infty, \tag{8.4b}$$

$$\delta_{\dot{\nu}} = 0, \quad \delta'_{\dot{\nu}} \neq 0 \quad \Longrightarrow \quad \psi_{\dot{\nu}-1} = 0, \quad \varphi_{\dot{\nu}} = \infty, \quad \omega_{\dot{\nu}} = 0, \tag{8.4c}$$

$$\delta_{\dot{\nu}} = 0, \quad \delta'_{\dot{\nu}} = 0 \quad \Longrightarrow \quad \psi_{\dot{\nu}-1} = 0, \quad \varphi_{\dot{\nu}}, \ \omega_{\dot{\nu}} \text{ undefined}. \tag{8.4d}$$

In the first case, (8.4b), a pivot breakdown occurs. The second case, (8.4c), is a Lanczos breakdown. Here, we could still compute $\mathbf{y}_{\dot{\nu}+1} = \mathbf{y}_{\dot{\nu}}$, $\widetilde{\mathbf{y}}_{\dot{\nu}+1} = \widetilde{\mathbf{y}}_{\dot{\nu}}$, and $\delta_{\dot{\nu}+1} = 0$; but then $\psi_{\dot{\nu}}$ would be indefinite, and we would have $\mathbf{v}_{\dot{\nu}+1} \in \mathcal{K}_{\dot{\nu}+1}$, $\widetilde{\mathbf{v}}_{\dot{\nu}+1} \in \widetilde{\mathcal{K}}_{\dot{\nu}+1}$ for any value of $\psi_{\dot{\nu}}$. In other words, the Krylov space generation is stopped. Hence, the algorithm *stalls permanently*, with no chance to recover. In the next section we will derive yet another version, BIODIR, of the BICG method, which under a certain assumption can recover in this situation. The last case, (8.4d), is simultaneously a Lanczos and a pivot breakdown.

Formula (8.3d) shows clearly that the approximations $\mathbf{x}_n$ are modified by moving along the *direction vectors* $\mathbf{v}_n$, and it allows us to express $\mathbf{x}_n - \mathbf{x}_0$ as a sum of corrections:

$$\mathbf{x}_n = \mathbf{x}_0 + \sum_{j=0}^{n-1} \mathbf{v}_j\omega_j.$$

This formula was actually the starting point of Lanczos' application of the BIOC algorithm to linear systems (Lanczos 1952, p. 37). When $\mathbf{A}$ is Hermitian positive definite and $\mathbf{B} = \mathbf{I}$, that is, in the classical CG situation, one can say that for finding $\mathbf{x}_{n+1}$ one moves along the straight line determined by the approximation $\mathbf{x}_n$ and the direction $\mathbf{v}_n$ until one reaches the minimum of the quadratic function $\mathbf{x} \mapsto \frac{1}{2}\mathbf{x}^\star\mathbf{A}\mathbf{x} - \mathbf{b}^\star\mathbf{x}$ on this line, or equivalently, the minimum of $\mathbf{x} \mapsto (\mathbf{x}_{\text{ex}} - \mathbf{x})^\star\mathbf{A}(\mathbf{x}_{\text{ex}} - \mathbf{x})$. In fact, as we have mentioned in Section 4, this is then also the minimum among all $\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_n$. At $\mathbf{x}_n$ the gradient (direction of steepest ascent) of this function happens to be $-\mathbf{y}_n$. Thus the gradients (residuals) are orthogonal to each other. This geometric interpretation leads readily to a variety of generalizations of the conjugate gradient method to nonlinear minimization problems; see, for instance, Murray (1972, Chapter 5) and Hestenes (1980).

8.2. The inconsistent BIOMIN *algorithm*

After successfully eliminating the pivot breakdown of the BIORES algorithm by making a minor modification we may wonder whether it is possible to eliminate this type of breakdown here too by introducing an inconsistent version of BIOMIN. It is not difficult to define such a version, but it turns out that the goal is missed. Nevertheless, the inconsistent BIOMIN algorithm is of some interest due to its close relationship to the coupled two-term version of the QMR method.

ALGORITHM 8. (INCONSISTENT BIOMIN FORM OF THE BICG METHOD) Let $\mathbf{y}_0 := (\mathbf{b} - \mathbf{A}\mathbf{x}_0)/\gamma_{-1}$ with some $\gamma_{-1} \neq 0$, redefine $\mathbf{x}_0 := \mathbf{x}_0/\gamma_{-1}$, and set $\dot{\pi}_0 := 1/\gamma_{-1}$ as in Algorithm 3 (inconsistent BIORES). Modify Algorithm 6 (BIOC) by also computing, at the nth step,

$$\mathbf{x}_{n+1} := -(\mathbf{x}_n\varphi_n + \mathbf{v}_n)/\gamma_n$$

and

$$\dot{\pi}_{n+1} := -\dot{\pi}_n\varphi_n/\gamma_n. \tag{8.5}$$

In this algorithm the vectors $\mathbf{y}_n$ and $\mathbf{x}_n$ are again related by (4.17), with the same scale factors $\dot{\pi}_n$. The proof is once again by induction:

$$(\mathbf{b}\dot{\pi}_{n+1} - \mathbf{A}\mathbf{x}_{n+1})\gamma_n = -\mathbf{b}\dot{\pi}_n\varphi_n + \mathbf{A}\mathbf{x}_n\varphi_n + \mathbf{A}\mathbf{v}_n = -(\mathbf{y}_n\varphi_n - \mathbf{A}\mathbf{v}_n) = \mathbf{y}_{n+1}\gamma_n.$$

However, the BIOC algorithm, and thus also the inconsistent BIOMIN algorithm, in any case break down when $\delta'_{n+1} = 0$. Therefore, in exact arithmetic, the latter algorithm does not bring any substantial advantage. However, as we mentioned before, the BIOC algorithm seems to be less affected by round-off than the BIO algorithm. While we do not expect a big difference between consistent and inconsistent BIOMIN in this respect, round-off error control measures are somewhat easier to implement when the Lanczos vectors are normalized.

Concerning the relation to BIORES the following holds.

Lemma 8.1 If the same starting vectors are used, then the three algorithms BIORES (Algorithm 2), BIOMIN (Algorithm 7), and inconsistent BIOMIN (Algorithm 8) are mathematically equivalent, that is, they break down at the same time and they produce the same iterates $\mathbf{x}_n$, except that those of inconsistent BIOMIN are scaled by the factors $\dot{\pi}_n$.

Proof. The relation between the BIO and the BIOC algorithm was established in Theorem 7.2. Compared to the BIO algorithm, BIORES additionally requires that $\gamma_n := -\alpha_n - \beta_{n-1} \neq 0$. But since γ_n is the same for the BIO and the BIOC algorithm, and $\gamma_n := -\varphi_n$ in BIOMIN, this condition is equivalent to $\varphi_n \neq 0$, which has to be observed in the BIOC algorithm any-

way, and which, conversely, is the only additional condition there compared to the BIO algorithm. □

9. The BIODIR form of the BICG method; comparison

9.1. *The* BIC *algorithm*

Formula (8.3d) suggests that the iterates $\mathbf{x}_n$ can be computed by building up the biconjugate sequences $\{\mathbf{v}_n\}$ and $\{\tilde{\mathbf{v}}_n\}$ and using the vectors $\mathbf{y}_n$ and $\tilde{\mathbf{y}}_n$ only for the determination of the step size $1/\varphi_n$, which depends on them via δ_n; see (7.1a). Since the biconjugate sequences can be thought of as being biorthogonal with respect to the formal inner product $\langle\tilde{\mathbf{v}}, \mathbf{v}\rangle_{\mathbf{BA}} := \tilde{\mathbf{v}}^\star\mathbf{BAv}$, we can construct them with the BIO algorithm if we substitute this inner product there. The sequences generated in this way can differ in scaling from those produced by BIOMIN, but we will see in a moment how to make them identical. We call the resulting algorithm the biconjugation or BIC algorithm, and although it is identical to the BIO algorithm except for the change in the formal inner product, we give it here in detail, because it will be a part of the BIODIR form of BICG, and because we want to fix the notation. We distinguish the new recurrence coefficients and the inner products by primes from those of the BIO algorithm.

ALGORITHM 9. (BIC ALGORITHM)
Choose $\mathbf{v}_0, \tilde{\mathbf{v}}_0 \in \mathbb{C}^N$ such that $\delta'_0 := \langle\tilde{\mathbf{v}}_0, \mathbf{Av}_0\rangle_{\mathbf{B}} \neq 0$, and set $\beta'_{-1} := 0$. For $n = 0, 1, \ldots$ compute

$$\alpha'_n := \langle\mathbf{A}^\star\tilde{\mathbf{v}}_n, \mathbf{Av}_n\rangle_{\mathbf{B}}/\delta'_n, \tag{9.1a}$$

$$\tilde{\alpha}'_n := \overline{\alpha'_n}, \tag{9.1b}$$

$$\beta'_{n-1} := \overline{\tilde{\gamma}'_{n-1}}\delta'_n/\delta'_{n-1}, \quad (\text{if } n > 0), \tag{9.1c}$$

$$\tilde{\beta}'_{n-1} := \overline{\gamma'_{n-1}\delta'_n/\delta'_{n-1}} = \overline{\beta'_{n-1}\gamma'_{n-1}}/\tilde{\gamma}'_{n-1}, \quad (\text{if } n > 0), \tag{9.1d}$$

$$\mathbf{v}_{\text{temp}} := \mathbf{Av}_n - \mathbf{v}_n\alpha'_n - \mathbf{v}_{n-1}\beta'_{n-1}, \tag{9.1e}$$

$$\tilde{\mathbf{v}}_{\text{temp}} := \mathbf{A}^\star\tilde{\mathbf{v}}_n - \tilde{\mathbf{v}}_n\tilde{\alpha}'_n - \tilde{\mathbf{v}}_{n-1}\tilde{\beta}'_{n-1}, \tag{9.1f}$$

$$\delta'_{\text{temp}} := \langle\tilde{\mathbf{v}}_{\text{temp}}, \mathbf{Av}_{\text{temp}}\rangle_{\mathbf{B}}\,; \tag{9.1g}$$

if $\delta'_{\text{temp}} = 0$, choose $\gamma'_n \neq 0$ and $\tilde{\gamma}'_n \neq 0$, set

$$\nu' := n + 1, \quad \mathbf{v}_{\nu'} := \mathbf{v}_{\text{temp}}/\gamma'_n, \quad \tilde{\mathbf{v}}_{\nu'} := \tilde{\mathbf{v}}_{\text{temp}}/\tilde{\gamma}'_n, \quad \delta'_{n+1} := 0,$$

and stop; otherwise, choose $\gamma'_n \neq 0$, $\gamma'_n \neq 0$, and δ'_{n+1} such that

$$\gamma'_n\overline{\tilde{\gamma}'_n}\delta'_{n+1} = \delta'_{\text{temp}},$$

set

$$\mathbf{v}_{n+1} := \mathbf{v}_{\text{temp}}/\gamma'_n, \quad \tilde{\mathbf{v}}_{n+1} := \tilde{\mathbf{v}}_{\text{temp}}/\tilde{\gamma}'_n, \tag{9.1h}$$

and proceed with the next step.

It is most natural to choose $\gamma'_n := \|\mathbf{y}_{\text{temp}}\|$, $\widetilde{\gamma}'_n := \|\widetilde{\mathbf{y}}_{\text{temp}}\|$, so that the $\|\mathbf{v}_{n+1}\| = \|\widetilde{\mathbf{v}}_{n+1}\| = 1$, in analogy to (2.27).

Theorem 2.1 now transforms into the following statement.

Corollary 9.1 The sequences $\{\mathbf{v}_n\}_{n=0}^{\nu'}$ and $\{\widetilde{\mathbf{v}}_n\}_{n=0}^{\nu'}$ generated by the BIC algorithm are biconjugate (with respect to $\mathbf{A}$), except that $\langle \widetilde{\mathbf{v}}_{\nu'}, \mathbf{A}\mathbf{v}_{\nu'}\rangle_{\mathbf{B}} = 0$. That is, for $m, n = 0, 1, \ldots, \nu'$,

$$\langle \widetilde{\mathbf{v}}_m, \mathbf{A}\mathbf{v}_n\rangle_{\mathbf{B}} = \begin{cases} 0, & m \neq n, \\ \delta'_n, & m = n, \end{cases} \tag{9.2}$$

where $\delta'_n \neq 0$ for $0 \leq n \leq \nu' - 1$, but $\delta'_{\nu'} = 0$. Moreover, for $n = 1, \ldots, \nu' - 1$, (7.4) is valid for $\mathbf{v}_n$ and $\widetilde{\mathbf{v}}_n$. Conversely, the sequences $\{\mathbf{v}_n\}_{n=0}^{\nu'}$ and $\{\widetilde{\mathbf{v}}_n\}_{n=0}^{\nu'}$ are uniquely determined up to scaling by the condition (9.2) with $\delta'_n \neq 0$ $(n = 0, 1, \ldots, \nu' - 1)$, $\delta'_{\nu'} = 0$, and the assumption $\mathbf{v}_n \in \mathcal{K}_{n+1}$, $\widetilde{\mathbf{v}}_n \in \widetilde{\mathcal{K}}_{n+1}$ $(n = 0, 1, \ldots, \nu')$.

In view of (7.10) it should be clear that the recurrence coefficients α'_n, β'_n, γ'_n and $\widetilde{\alpha}'_n$, $\widetilde{\beta}'_n$, $\widetilde{\gamma}'_n$ in the BIC algorithm are the elements of the matrices $\underline{\mathbf{T}}'_n$ and $\underline{\widetilde{\mathbf{T}}}'_n$ that we introduced in Section 7. In particular, (7.7)–(7.8) hold, and the shorthand notation for recurrences is

$$\mathbf{A}\mathbf{V}_n = \mathbf{V}_{n+1}\underline{\mathbf{T}}'_n, \quad \mathbf{A}^*\widetilde{\mathbf{V}}_n = \widetilde{\mathbf{V}}_{n+1}\underline{\widetilde{\mathbf{T}}}'_n.$$

9.2. The BIODIR *algorithm*

For the application of the BIC algorithm to linear systems of equations an additional recurrence for $\mathbf{x}_n$ is needed again. We keep the freedom of scaling the direction vectors $\mathbf{v}_n$ and $\widetilde{\mathbf{v}}_n$ arbitrarily, but then have to determine the step length appropriately.

ALGORITHM 10. (BIODIR FORM OF THE BICG METHOD)
For solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0$, set $\mathbf{v}_0 := \mathbf{y}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, and apply Algorithm 9 (BIC), additionally computing

$$\omega'_n := \langle \widetilde{\mathbf{v}}_n, \mathbf{y}_n\rangle_{\mathbf{B}} / \delta'_n, \tag{9.3a}$$

$$\mathbf{x}_{n+1} := \mathbf{x}_n + \mathbf{v}_n\omega'_n, \tag{9.3b}$$

$$\mathbf{y}_{n+1} := \mathbf{y}_n - \mathbf{A}\mathbf{v}_n\omega'_n. \tag{9.3c}$$

If $\mathbf{y}_{n+1} = \mathbf{o}$, the process terminates and $\mathbf{x}_{n+1}$ is the solution; if $\delta'_{n+1} = 0$, but $\mathbf{y}_{n+1} \neq \mathbf{o}$, the algorithm breaks down. In both cases we set $\nu' := n + 1$.

Again, $\mathbf{y}_n$ is the nth residual. In fact, by assuming that $\mathbf{b} - \mathbf{A}\mathbf{x}_n = \mathbf{y}_n$ and using (9.3b) and (9.3c) we get

$$\mathbf{b} - \mathbf{A}\mathbf{x}_{n+1} = \mathbf{b} - \mathbf{A}\mathbf{x}_n - \mathbf{A}\mathbf{v}_n\omega'_n = \mathbf{y}_n - \mathbf{A}\mathbf{v}_n\omega'_n = \mathbf{y}_{n+1},$$

as required. The formula (9.3a) for ω'_n enforces $\langle \tilde{\mathbf{v}}_n, \mathbf{y}_{n+1} \rangle_{\mathbf{B}} = 0$. In Theorem 9.2 below, we will verify that BIODIR creates the same iterates as BIOMIN and BIORES, but has different breakdown conditions.

9.3. Comparison of breakdown conditions of BICG *algorithms*

The BIODIR algorithm can only break down due to $\delta'_{\text{temp}} = 0$. When $\omega'_n = 0$, the algorithm can recover: although it *stalls* for one step, that is, $\mathbf{x}_{n+1} = \mathbf{x}_n$ and $\mathbf{y}_{n+1} = \mathbf{y}_n$, a new direction $\mathbf{v}_{n+1}$ is created. Therefore, in general, the method cannot be equivalent to BIORES or BIOMIN since necessarily $\mathbf{y}_{n+1} \neq \mathbf{y}_n$ in both. But if $\omega'_n \neq 0$ $(n = 0, \ldots, \nu - 1)$, it is indeed equivalent to these two methods, as was noted in Jea and Young (1983, p. 411) and is proven next. In the following theorem the various conditions for a first breakdown, a stagnation point, or the termination are summarized and compared.

Theorem 9.2 Assume that the same initial approximation $\mathbf{x}_0$ and the same initial vectors $\mathbf{v}_0 := \mathbf{y}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\tilde{\mathbf{v}}_0 := \tilde{\mathbf{y}}_0$ are used in BIODIR, consistent or inconsistent BIOMIN, and consistent and inconsistent BIORES. Let ν', ν, and $\dot{\nu}$ be the indices of first breakdown or termination of BIODIR, inconsistent BIORES, and the other three algorithms, respectively.

Then $\dot{\nu} = \min\{\nu, \nu'\}$, and the following five conditions for a Lanczos breakdown are equivalent:

(i') $\omega'_m \neq 0$ (for all $m < n$) and $\langle \tilde{\mathbf{v}}_n, \mathbf{y}_n \rangle_{\mathbf{B}} = 0$ in BIODIR
(ii') $\delta_n = 0$ in BIOMIN
(iii') $\delta_n = 0$ in (consistent) BIORES
(iv') $\alpha_m + \beta_{m-1} \neq 0$ (for all $m < n$) and $\delta_n = 0$ in inconsistent BIORES
(v') $n = \nu = \dot{\nu}$.

Condition (i') implies that either $\omega'_n = 0$ or $\delta'_n = 0$; in the latter case, ω'_n is undefined and $n = \nu = \dot{\nu} = \nu'$. In the former, we have stagnation but no breakdown of BIODIR.

Likewise, the next four conditions for a pivot breakdown are equivalent:

(i") $\langle \tilde{\mathbf{v}}_m, \mathbf{y}_m \rangle_{\mathbf{B}} \neq 0$ (for all $m < n$) and $\delta'_n = 0$ in BIODIR
(ii") $\delta'_n = 0$ in BIOMIN
(iii") $\alpha_n + \beta_{n-1} = 0$ in (consistent) BIORES
(iv") $\dot{\pi}_n = 0$ in inconsistent BIORES
(v") $n = \dot{\nu} = \nu'$.

The conditions (ii')–(iv') and (i")–(iii") all cause either the termination or a breakdown of the respective algorithm. However, (iv") does not stop inconsistent BIORES.

The three consistent algorithms produce the same approximants $\mathbf{x}_1, \ldots, \mathbf{x}_{\dot\nu}$ and hence the same residuals $\mathbf{y}_1, \ldots, \mathbf{y}_{\dot\nu}$. If we choose

$$\gamma'_n := \gamma_n \ (= -\varphi_n = -1/\omega_n) \tag{9.4}$$

in BIODIR, then, for $n = 0, \ldots, \dot\nu$, the vectors $\mathbf{v}_n =: \mathbf{v}_n^{\mathrm{DIR}}$ and $\widetilde{\mathbf{v}}_n =: \widetilde{\mathbf{v}}_n^{\mathrm{DIR}}$ produced by BIODIR are the same as the biconjugate vectors $\mathbf{v}_n =: \mathbf{v}_n^{\mathrm{MIN}}$ and $\widetilde{\mathbf{v}}_n =: \widetilde{\mathbf{v}}_n^{\mathrm{MIN}}$ of BIOMIN. This implies that for $0 \le n \le \dot\nu - 1$ the parameters $\alpha'_n, \beta'_{n-1}, \gamma'_n$ of BIODIR are the elements of the matrix $\underline{\mathbf{T}}'_{\dot\nu}$ of (7.8).

In general, without the particular choice (9.4), we have

$$\mathbf{v}_n^{\mathrm{MIN}}\Gamma_n = \mathbf{v}_n^{\mathrm{DIR}}\Gamma'_n, \quad \widetilde{\mathbf{v}}_n^{\mathrm{MIN}}\widetilde{\Gamma}_n = \widetilde{\mathbf{v}}_n^{\mathrm{DIR}}\widetilde{\Gamma}'_n \quad (n \le \dot\nu), \tag{9.5}$$

where

$$\Gamma_n := \gamma_0\gamma_1\cdots\gamma_{n-1}, \quad \Gamma'_n := \gamma'_0\gamma'_1\cdots\gamma'_{n-1}. \tag{9.6}$$

The submatrix $\mathbf{T}'_{\dot\nu}$ of the tridiagonal matrix $\mathbf{T}'_{\nu'}$ with the parameters α'_n, β'_{n-1}, γ'_n of BIODIR and the bidiagonal matrices $\mathbf{L}_{\dot\nu}$ and $\mathbf{U}_{\dot\nu}$ of BIOMIN are then related by

$$\mathbf{D}^{-1}_{\Gamma';\dot\nu}\,\mathbf{T}'_{\dot\nu}\,\mathbf{D}_{\Gamma';\dot\nu} = \mathbf{D}^{-1}_{\Gamma;\dot\nu}\,\mathbf{U}_{\dot\nu}\,\underline{\mathbf{L}}_{\dot\nu}\,\mathbf{D}_{\Gamma;\dot\nu}, \tag{9.7}$$

where

$$\mathbf{D}_{\Gamma;n} := \mathrm{diag}(1, \Gamma_1, \ldots, \Gamma_{n-1}), \quad \mathbf{D}_{\Gamma';n} := \mathrm{diag}(1, \Gamma'_1, \ldots, \Gamma'_{n-1}). \tag{9.8}$$

The step size ω'_n in BIODIR can be expressed as

$$\omega'_n = \omega_n\Gamma'_n/\Gamma_n = -\Gamma'_n/\Gamma_{n+1}, \quad n = 0, 1, \ldots, \dot\nu - 1.$$

Proof. It is clear that up to a possible earlier breakdown of BIOMIN the formulae (9.1a)–(9.1h) of the BIC algorithm produce the same biconjugate sequences $\{\mathbf{v}_n\}_{n=0}^{\dot\nu}$ and $\{\widetilde{\mathbf{v}}_n\}_{n=0}^{\dot\nu}$ as BIOMIN if the $(n+1, n)$-element γ'_n of the matrix $\mathbf{T}'_{\dot\nu}$ satisfying (7.10) is chosen appropriately, namely, so that $\mathbf{T}'_{\dot\nu}$ and $\mathbf{T}_{\dot\nu}$ are related by (7.7). (In fact, according to Theorem 2.1, applied with $\mathbf{B} := \mathbf{BA}$, these sequences are uniquely determined by the biconjugacy condition and the scaling.) Since $\mathbf{T}_{\dot\nu}$ and $\mathbf{T}'_{\dot\nu}$ then have the same subdiagonal elements, we need $\gamma'_n = \gamma_n$ for identical sequences. Choosing γ'_n differently just rescales the two vector sequences. From (7.1c), (7.1d), (7.1h), and (7.1i), it follows easily that (9.5) and (9.6) hold, which, in view of (7.7) and (7.10), lead readily to (9.7).

In the formulae (9.3a)–(9.3c), a scale factor for $\mathbf{v}_n$ inherited by δ'_n yields the inverse factor for ω'_n, which cancels when $\mathbf{y}_{n+1}$ and $\mathbf{x}_{n+1}$ are evaluated. Moreover, a scale factor for $\widetilde{\mathbf{v}}_n$ has no effect. Hence, to prove that the latter two vectors are the same as in consistent BIOMIN it suffices to verify this for a fixed scaling, say for the one induced by $\gamma'_n := \gamma_n = -\varphi_n$, when $\mathbf{v}_n$ and $\widetilde{\mathbf{v}}_n$ are the same as in BIOMIN. For the induction proof we assume

that the pair $\mathbf{y}_n$, $\mathbf{x}_n$ is the same as in BIOMIN. Comparing (9.3b) and (9.3c) with (8.3b) and (8.3d) we see that the pair $\mathbf{y}_{n+1}$, $\mathbf{x}_{n+1}$ is again the same as in BIOMIN if and only if $\omega'_n = \omega_n$. By (7.2), (7.4), and (8.3h), $\langle \widetilde{\mathbf{y}}_n, \mathbf{y}_n \rangle_{\mathbf{B}} = \langle \widetilde{\mathbf{v}}_n, \mathbf{y}_n \rangle_{\mathbf{B}}$. Furthermore, if $\delta'_n \neq 0$, then by (7.2), (8.3a), and (9.3a) we can conclude that indeed

$$\omega_n = \frac{\delta_n}{\delta'_n} = \frac{\langle \widetilde{\mathbf{y}}_n, \mathbf{y}_n \rangle_{\mathbf{B}}}{\delta'_n} = \frac{\langle \widetilde{\mathbf{v}}_n, \mathbf{y}_n \rangle_{\mathbf{B}}}{\delta'_n} = \omega'_n .$$

This line also exhibits that $\omega'_n \neq 0, n = 0, \ldots, \dot{\nu} - 1$. Since $\langle \widetilde{\mathbf{y}}_n, \mathbf{y}_n \rangle_{\mathbf{B}} = \langle \widetilde{\mathbf{v}}_n, \mathbf{y}_n \rangle_{\mathbf{B}}$ is still valid for $n = \dot{\nu}$, the equivalence of (i')–(iii') holds, and since $\delta_{\dot{\nu}} = 0$ is the common breakdown condition of consistent and inconsistent BIORES, (iv) and (v') are equivalent too.

In view of (8.2) and (8.3a) the condition $\varphi_{\dot{\nu}} = 0$ (which by definition of $\dot{\nu}$ implies $\delta_{\dot{\nu}} \neq 0$; see (8.4a)–(8.4d)) is clearly equivalent to $\delta'_{\dot{\nu}} = 0$. Furthermore, since the consistent versions of BIOMIN and BIORES are related by $\varphi_n = -\gamma_n = \alpha_n + \beta_{n-1}$, the equivalence of (i")–(iii") and (v") follows. The fact that, in inconsistent BIORES, $\dot{\pi}_n = 0$ signals a pivot breakdown will follow in Section 12 (Theorem 12.1), where we will prove that $\dot{\pi}_n$ is the value at 0 of the nth Lanczos polynomial. It is also indicated by the infinity of the nth approximant $\mathbf{x}_n/\dot{\pi}_n$ of the solution $\mathbf{x}_{\text{ex}}$. Finally, since all conditions except (i') for BIODIR and (iv") for inconsistent BIORES imply a breakdown or the termination of the respective algorithm, and since no other types of breakdown exist, it follows that $\dot{\nu} \leq \nu'$ and $\dot{\nu} \leq \nu$, and that always at least one equality sign holds. □

What more can we say about the case where $\dot{\nu} < \nu'$, that is, $\omega'_n = 0$ for some $n < \nu'$? As we have seen, BIODIR stalls but does not break down. The sequences $\{\mathbf{v}_m\}_{m=0}^{\nu'-1}$ and $\{\widetilde{\mathbf{v}}_m\}_{m=0}^{\nu'-1}$ are still biconjugate (since they are generated by the BIC algorithm), and $\mathbf{y}_m$ is still the residual of $\mathbf{x}_m$. Using the connection to Padé approximation one can easily show that in this situation $\omega'_n = 0$ can only occur for isolated values of n. The proof was given in Gutknecht (1990, p. 30); the facts it is based on, namely the Lanczos–Padé connection and the block structure theorem, were also presented in Gutknecht (1994*b*). This has the following immediate consequence.

Theorem 9.3 If in the BIODIR algorithm $\omega'_n = 0$ for some n, then $\omega'_{n-1} \neq 0$ (if $n > 0$) and $\omega'_{n+1} \neq 0$ (if $n < \nu' - 1$). The sequence $\{\mathbf{y}_n\}_{n=0}^{\nu'-1}$ of residuals generated by BIODIR satisfies

$$\widetilde{\mathcal{K}}_{n+1} \perp_{\mathbf{B}} \mathbf{y}_{n+1} \quad \text{if} \quad \omega'_n \neq 0,$$

$$\widetilde{\mathcal{K}}_n \perp_{\mathbf{B}} \mathbf{y}_{n+1} = \mathbf{y}_n \quad \text{if} \quad \omega'_n = 0.$$

Table 1. *Matrix relations that describe the recurrences of the* BIO, BIOC, *and* BIC *algorithms. Only those for the right Lanczos and direction vectors are shown. Additionally, on the left, the biorthogonality and biconjugacy conditions are shown, while, on the right, the relations between the matrices of the recurrence coefficients are listed. The lower relation,* $\underline{\mathbf{T}}'_n = \mathbf{U}_{n+1}\underline{\mathbf{L}}_n$, *assumes that* $\gamma'_k = \gamma_k$ *(for all* k*)*

Algorithm	Biorthogonality	Recurrences	Relationships
1 BIO		$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n$	
	$\widetilde{\mathbf{Y}}_n^\star \mathbf{B}\mathbf{Y}_n = \mathbf{D}_{\delta;n}$		$\underline{\mathbf{T}}_n = \underline{\mathbf{L}}_n\mathbf{U}_n$
6 BIOC		$\mathbf{Y}_n = \mathbf{V}_n\mathbf{U}_n \quad \mathbf{A}\mathbf{V}_n = \mathbf{Y}_{n+1}\underline{\mathbf{L}}_n$	
	$\widetilde{\mathbf{V}}_n^\star \mathbf{B}\mathbf{A}\mathbf{V}_n = \mathbf{D}_{\delta';n}$		$\underline{\mathbf{T}}'_n = \mathbf{U}_{n+1}\underline{\mathbf{L}}_n$
9 BIC		$\mathbf{A}\mathbf{V}_n = \mathbf{V}_{n+1}\underline{\mathbf{T}}'_n$	

9.4. An overview of BICG *algorithms*

In Table 1 we first summarize the principle matrix relations of the BIO, BIOC, and BIC algorithms. To get an overview of the five BICG algorithms that we have discussed, we list in Table 2 the various vectors and the corresponding polynomials that come up (except for the iterates $\mathbf{x}_n$, which appear everywhere, of course). Those vectors that are listed in several algorithms are, up to scaling, the same. The table does not give full information about the memory requirement, however, as sometimes previously computed vectors or results of matrix-vector products have to be stored. In the last two columns of the table it is indicated if a Lanczos breakdown ('L') or a pivot breakdown ('P') can occur in the respective algorithm.

In Table 3 we compile the names of the coefficients and the corresponding matrices that belong to each method. However, we do not include those with tildes, as they are closely related to those without tildes.

10. Alternative ways to apply the QMR approach to BICG

As an alternative to the BICG method that is based on a Petrov–Galerkin condition, we discussed in Section 5 the QMR approach for solving a non-Hermitian linear system of equations. Starting from the representations

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Y}_n\mathbf{k}_n, \quad \mathbf{r}_n = \mathbf{r}_0 - \mathbf{A}\mathbf{Y}_n\mathbf{k}_n, \tag{10.1}$$

for the nth approximate solution and its residual, we saw by inserting $\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n$ and $\mathbf{r}_0 = \mathbf{y}_0\rho_0 = \mathbf{Y}_{n+1}\underline{\mathbf{e}}_1\rho_0$ that

$$\mathbf{r}_n = \mathbf{Y}_{n+1}\mathbf{q}_n, \quad \mathbf{q}_n := \underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}_n\mathbf{k}_n. \tag{10.2}$$

Table 2. *Krylov space vectors and corresponding polynomials that appear in our five forms of the* BiCG *method. Different scaling is mirrored by upper indices in the notation for the polynomials, but not in the one for the vectors. The complex conjugate polynomials, which are associated with left vectors, can be multiplied by yet another scale factor, in general. In the columns 'L' and 'P' we indicate if an algorithm is susceptible to Lanczos breakdown or pivot breakdown, respectively*

Algorithm	Vectors	Polynomials	L	P
2 BiORes (consistent)	$\mathbf{y}_n$, $\widetilde{\mathbf{y}}_n$	p_n, $\overline{p_n}$	$\surd$	$\surd$
3 BiORes (inconsistent)	$\mathbf{y}_n$, $\widetilde{\mathbf{y}}_n$	p_n^{INC}, $\overline{p_n^{\mathrm{INC}}}$	$\surd$	
7 BiOMin (standard BiCG)	$\mathbf{y}_n$,$\widetilde{\mathbf{y}}_n$,$\mathbf{v}_n$,$\widetilde{\mathbf{v}}_n$	p_n, $\overline{p_n}$, $\widehat{p}_n$, $\overline{\widehat{p}_n}$	$\surd$	$\surd$
8 BiOMin (inconsistent)	$\mathbf{y}_n$, $\widetilde{\mathbf{y}}_n$, $\mathbf{v}_n$, $\widetilde{\mathbf{v}}_n$	p_n^{INC}, $\overline{p_n^{\mathrm{INC}}}$, $\widehat{p}_n^{\mathrm{INC}}$, $\overline{\widehat{p}_n^{\mathrm{INC}}}$	$\surd$	$\surd$
10 BiODir	$\mathbf{y}_n$, $\mathbf{v}_n$, $\widetilde{\mathbf{v}}_n$	p_n, $\widehat{p}_n^{\mathrm{DIR}}$, $\overline{\widehat{p}_n^{\mathrm{DIR}}}$		$\surd$

Table 3. *Coefficients, inner products, and corresponding matrices that appear in our five forms of the* BiCG *method*

Algorithm	Coefficients	Inner products	Matrices
2 BiORes (consistent)	α_n, β_{n-1}, γ_n	δ_{n+1}	$\underline{\mathbf{T}}_n$, $\mathbf{D}_{\delta;n}$
3 BiORes (inconsistent)	α_n, β_{n-1}, γ_n, $\dot{\pi}_{n+1}$	δ_n	$\mathbf{T}_n$, $\mathbf{D}_{\delta;n}$
7 BiOMin (stand. BiCG)	ω_n, ψ_n	δ_n, δ'_n	$\underline{\mathbf{L}}_n$, $\mathbf{U}_n$, $\mathbf{D}_{\delta;n}$, $\mathbf{D}_{\delta';n}$
8 BiOMin (inconsistent)	ω_n, ψ_n, γ_n,	δ_n, δ'_n	$\underline{\mathbf{L}}_n$, $\mathbf{U}_n$, $\mathbf{D}_{\delta;n}$, $\mathbf{D}_{\delta';n}$
10 BiODir	α'_n, β'_n, γ'_n, ω'_n	δ'_{n+1}	$\mathbf{T}'_n$, $\underline{\mathbf{L}}_n$, $\mathbf{D}_{\delta';n}$

Recall that here the columns of $\mathbf{Y}_{n+1}$ are expected to be normalized. The QMR method then minimizes the quasi-residual $\mathbf{q}_n$ (instead of the residual), and this amounts to solving the least squares problem

$$||\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}_n\mathbf{k}_n||^2 = \min! \tag{10.3}$$

with the $(n+1)\times n$ tridiagonal matrix $\underline{\mathbf{T}}_n$ (which will have some additional fill-in in the upper triangle if look-ahead is needed).

Instead of using the BiO algorithm, can we also apply the BiOC or even the BiC algorithm to find the QMR iterates? Replacing (10.1) by

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n\widehat{\mathbf{k}}_n, \quad \mathbf{r}_n = \mathbf{r}_0 - \mathbf{A}\mathbf{V}_n\widehat{\mathbf{k}}_n,$$

and now substituting $\mathbf{AV}_n = \mathbf{Y}_{n+1}\underline{\mathbf{L}}_n$, we find

$$\mathbf{r}_n = \mathbf{Y}_{n+1}\mathbf{q}_n, \quad \mathbf{q}_n := \underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{L}}_n\widehat{\mathbf{k}}_n.$$

Since $\mathbf{r}_n$ is again written as a linear combination of the columns of $\mathbf{Y}_{n+1}$, the coefficient vector $\mathbf{q}_n$ is the same as in (10.2). The least squares problem

$$||\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{L}}_n\widehat{\mathbf{k}}_n||^2 = \min!$$

that has now to be solved involves a lower bidiagonal matrix, but it is equivalent to the one of (10.3): it could have been obtained from the latter by inserting $\underline{\mathbf{T}}_n = \underline{\mathbf{L}}_n\mathbf{U}_n$ and $\mathbf{U}_n\mathbf{k}_n = \widehat{\mathbf{k}}_n$. Hence, all we have done is a linear change of variables: while the coefficient vector $\widehat{\mathbf{k}}_n$ is different from $\mathbf{k}_n$, the approximate solution $\mathbf{x}_n$ is the same as before. (Of course, exact arithmetic is assumed here.) We suggest calling this form of the QMR method BIOCQMR for distinction. A complete description of it and its implementation, including a version of the BIOC algorithm with look-ahead, was given by Freund and Nachtigal (1994, 1993).

We can also construct a BICQMR form of the QMR approach. Writing

$$\mathbf{x}_n = \mathbf{x}_0 + \mathbf{V}_n\mathbf{k}'_n, \quad \mathbf{r}_n = \mathbf{r}_0 - \mathbf{AV}_n\mathbf{k}'_n,$$

and inserting $\mathbf{AV}_n = \mathbf{V}_{n+1}\underline{\mathbf{T}}'_n$ and $\mathbf{r}_0 = \mathbf{y}_0\rho_0 = \mathbf{v}_0\rho_0 = \mathbf{V}_{n+1}\underline{\mathbf{e}}_1\rho_0$, we obtain

$$\mathbf{r}_n = \mathbf{V}_{n+1}\mathbf{q}'_n, \quad \mathbf{q}'_n := \underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}'_n\mathbf{k}'_n.$$

However, the resulting least squares problem

$$||\underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{T}}'_n\mathbf{k}'_n||^2 = \min!$$

is no longer equivalent to the two previous ones, since $\mathbf{r}_n$ is here represented in a different basis, the columns of $\mathbf{V}_{n+1}$. One must expect that in typical examples these columns are even further away from orthogonal than those of $\mathbf{Y}_{n+1}$ (which are orthogonal in the symmetric case), and that therefore the norm of the true residual is larger than in BIOQMR and BIOCQMR.

11. Preconditioning

By suitable preconditioning the convergence of iterative linear equation solvers is often improved dramatically. In fact, in practice large linear systems of equations resulting from the discretization of partial differential equations are often so hard to treat that iterative methods do not converge at all unless the system is preconditioned, which means that the coefficient matrix is – implicitly or explicitly – replaced by another one with better convergence properties. However, the more effective preconditioners are, the more costly they tend to be, both regarding their one-time computation and the additional cost of evaluation (for instance matrix multiplication) per step.

In general, preconditioning can be viewed as replacing the given system, say,

$$\widehat{\mathbf{A}}\widehat{\mathbf{x}} = \widehat{\mathbf{b}} \quad \text{with initial approximation } \widehat{\mathbf{x}}_0, \tag{11.1}$$

by an equivalent system,

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{with initial approximation } \mathbf{x}_0,$$

where either

$$\begin{aligned} \mathbf{A} &:= \mathbf{C}_L\widehat{\mathbf{A}}\mathbf{C}_R, & \mathbf{b} &:= \mathbf{C}_L\widehat{\mathbf{b}}, \\ \mathbf{x}_0 &:= \mathbf{C}_R^{-1}\widehat{\mathbf{x}}_0, & \widehat{\mathbf{x}} &:= \mathbf{C}_R\mathbf{x}, \end{aligned} \tag{11.2}$$

or

$$\begin{aligned} \mathbf{A} &:= \mathbf{C}_L\widehat{\mathbf{A}}\mathbf{C}_R, & \mathbf{b} &:= \mathbf{C}_L(\widehat{\mathbf{b}} - \widehat{\mathbf{A}}\widehat{\mathbf{x}}_0), \\ \mathbf{x}_0 &:= \mathbf{o}, & \widehat{\mathbf{x}} &:= \widehat{\mathbf{x}}_0 + \mathbf{C}_R\mathbf{x}. \end{aligned} \tag{11.3}$$

The second version combines the preconditioning with a shift of the origin at the beginning of the iteration, as suggested by Freund and Nachtigal (1991). Note that the same shift of origin is also applied in the general concept of iterative refinement, and in Section 18 we will suggest also applying it at later stages.

We call $\mathbf{C}_L$ and $\mathbf{C}_R$ the *left* and the *right preconditioner* (as, for instance, in Ashby et al. (1990)). Many other authors (for instance, Golub and van Loan (1989), Saad (1996), Barrett, Berry, Chan, Demmel, Donato, Dongarra, Eijkhout, Pozo, Romine and van der Vorst (1994)) refer to $\mathbf{M}_L := \mathbf{C}_L^{-1}$ and $\mathbf{M}_R := \mathbf{C}_R^{-1}$ as left and right preconditioners. In fact, some preconditioning techniques, such as the various forms of incomplete LU factorizations (Meijerink and van der Vorst 1977), primarily generate $\mathbf{M}_L$ and $\mathbf{M}_R$, and then evaluate $\widetilde{\mathbf{y}} = \mathbf{A}\mathbf{y}$ by solving the two linear systems $\mathbf{M}_R\mathbf{t} = \mathbf{y}$ and $\mathbf{M}_L\widetilde{\mathbf{y}} = \widehat{\mathbf{A}}\mathbf{t}$. Other preconditioning techniques emerge directly as procedures for computing $\mathbf{t} = \mathbf{C}_R\mathbf{y}$ and $\widetilde{\mathbf{y}} = \mathbf{C}_L\widehat{\mathbf{A}}\mathbf{t}$. Finally, many algorithms can be reformulated so that the left and the right preconditioner can be combined into one matrix multiplication by the product $\mathbf{C}_R\mathbf{C}_L$ or one linear system with matrix $\mathbf{M}_L\mathbf{M}_R$ to solve per step; see, for instance, Saad (1996).

Often, only either a left or a right preconditioner is applied, that is, either $\mathbf{C}_R := \mathbf{I}$ or $\mathbf{C}_L := \mathbf{I}$. In the first case, $\mathbf{x} = \widehat{\mathbf{x}}$, so that the errors of the preconditioned system are the same as those of the original system. In the second case, the residuals remain unchanged. In the general situation, if we set $\mathbf{x}_{\text{ex}} = \mathbf{A}^{-1}\mathbf{b}$ and $\widehat{\mathbf{x}}_{\text{ex}} = \widehat{\mathbf{A}}^{-1}\widehat{\mathbf{b}}$, both (11.2) and (11.3) imply the relations

$$\mathbf{r}_n = \mathbf{C}_L\widehat{\mathbf{r}}_n, \quad \mathbf{x}_{\text{ex}} - \mathbf{x}_n = \mathbf{C}_R^{-1}(\widehat{\mathbf{x}}_{\text{ex}} - \widehat{\mathbf{x}}_n)$$

between the *preconditioned residuals* $\mathbf{r}_n := \mathbf{b} - \mathbf{A}\mathbf{x}_n$ and the *preconditioned errors* $\mathbf{x}_{\text{ex}} - \mathbf{x}_n$ on the one hand and the residuals $\widehat{\mathbf{r}}_n := \widehat{\mathbf{b}} - \widehat{\mathbf{A}}\widehat{\mathbf{x}}_n$ and errors $\widehat{\mathbf{x}}_{\text{ex}} - \widehat{\mathbf{x}}_n$ of the original system (11.1) on the other hand.

12. Lanczos and direction polynomials

Any vector $\mathbf{y}$ in the Krylov space $\mathcal{K}_{n+1} := \text{span}\,(\mathbf{y}_0, \mathbf{A}\mathbf{y}_0, \ldots, \mathbf{A}^n\mathbf{y}_0)$ can be written in the form $\mathbf{y} = p(\mathbf{A})\mathbf{y}_0$, where p is a polynomial of degree at most n, in which the matrix $\mathbf{A}$ is substituted for the argument. The induced mapping is an isomorphism as long as $\mathcal{K}_{n+1} \neq \mathcal{K}_n$, that is, as long as n is smaller than the grade of $\mathbf{y}_0$ with respect to $\mathbf{A}$. The sequence $\{\mathbf{y}_n\}_{n=0}^{\nu-1}$ of right Lanczos vectors generated by the BIO algorithm is in this way associated with the sequence $\{p_n\}_{n=0}^{\nu-1}$ of *Lanczos polynomials*, and from the three-term recurrence (2.19) for the former we get immediately a three-term recurrence formula for the latter. Of course, the analogue is true for the sequence $\{\widetilde{\mathbf{y}}_n\}_{n=0}^{\nu-1}$ of left Lanczos vectors, the operator $\mathbf{A}^\star$, and the corresponding nested sequence of Krylov spaces $\widetilde{\mathcal{K}}_{n+1} := \text{span}\,(\widetilde{\mathbf{y}}_0, \mathbf{A}^\star\widetilde{\mathbf{y}}_0, \ldots, (\mathbf{A}^\star)^n\widetilde{\mathbf{y}}_0)$. But from the recurrences (2.19) and the relations (2.21b) and (2.21d) we see that the coefficients of this second set of polynomials, $\{\widetilde{p}_n\}_{n=0}^{\nu-1}$, are just complex conjugate to those of the Lanczos polynomials if we choose $\widetilde{\gamma}_n = \overline{\gamma_n}$. (Otherwise, in view of (2.28), $\widetilde{p}_n$ would be a scalar multiple of $\overline{p_n}$; for simplicity, we assume $\widetilde{\gamma}_n = \overline{\gamma_n}$ in this section.) Similarly, from the BIOC formulae (7.1a)–(7.1j) it is seen that the vectors $\mathbf{v}_n \in \mathcal{K}_{n+1}$ and $\widetilde{\mathbf{v}}_n \in \widetilde{\mathcal{K}}_{n+1}$ can then be represented by a polynomial $\widehat{p}_n$ of degree n and the one with complex conjugate coefficients, $\overline{\widehat{p}_n}$, respectively. Since $\mathbf{v}_n$ is a direction vector, we call $\widehat{p}_n$ a *direction polynomial*. The coupled two-term recurrences of the BIOC algorithm translate into coupled two-term recurrences for the two polynomial sequences $\{p_n\}$ and $\{\widehat{p}_n\}$. A three-term recurrence for the direction polynomials alone follows by eliminating p_n from the coupled recurrences. Recall that the analogous elimination brought us from the BIOC to the BIC algorithm. The recurrences remain valid up to $n = \nu$, but the correspondence between $\mathcal{P}_n$ and $\mathcal{K}_n$ may no longer be one-to-one for $n = \nu$. When taking the different indices of first breakdown or termination of the various algorithms into account, we obtain altogether the following result.

Theorem 12.1 Let $\{\mathbf{y}_n\}_{n=0}^{\nu}$ and $\{\widetilde{\mathbf{y}}_n\}_{n=0}^{\nu}$ be the biorthogonal vector sequences generated by the BIO algorithm with $\widetilde{\gamma}_n = \gamma_n$, and let $\{\mathbf{v}_n\}_{n=0}^{\nu'}$ and $\{\widetilde{\mathbf{v}}_n\}_{n=0}^{\nu'}$ be the biconjugate vector sequences generated by the BIC algorithm using $\gamma'_n = \gamma_n$ and $\widetilde{\gamma}'_n = \overline{\gamma_n}$. Then there is a pair of sequences of polynomials, $\{p_n\}_{n=0}^{\nu}$ and $\{\widehat{p}_n\}_{n=0}^{\nu'}$, such that

$$\mathbf{y}_n = p_n(\mathbf{A})\mathbf{y}_0, \quad \widetilde{\mathbf{y}}_n = \overline{p_n}(\mathbf{A}^\star)\widetilde{\mathbf{y}}_0, \quad n = 0, 1, \ldots, \nu,$$

and

$$\mathbf{v}_n = \widehat{p}_n(\mathbf{A})\mathbf{v}_0, \quad \widetilde{\mathbf{v}}_n = \overline{\widehat{p}_n}(\mathbf{A}^\star)\widetilde{\mathbf{v}}_0, \quad n = 0, 1, \ldots, \nu'.$$

For $n < \dot{\nu} = \min\{\nu, \nu'\}$ these polynomial sequences satisfy the coupled two-term recurrences

$$\begin{aligned} p_0(\zeta) &:\equiv 1, \\ \widehat{p}_0(\zeta) &:\equiv 1, \\ p_{n+1}(\zeta) &:= [\zeta \widehat{p}_n(\zeta) - \varphi_n p_n(\zeta)]/\gamma_n, \\ \widehat{p}_{n+1}(\zeta) &:= p_{n+1}(\zeta) - \psi_n \widehat{p}_n(\zeta), \quad n = 0, 1, \ldots, \dot{\nu} - 1. \end{aligned} \tag{12.1}$$

They also satisfy individual three-term recurrences, namely

$$\begin{aligned} p_0(\zeta) &:\equiv 1, \\ p_1(\zeta) &:= (\zeta - \alpha_0) p_0(\zeta)/\gamma_0, \\ p_{n+1}(\zeta) &:= [(\zeta - \alpha_n) p_n(\zeta) - \beta_{n-1} p_{n-1}(\zeta)]/\gamma_n, \quad n = 1, \ldots, \nu - 1, \end{aligned} \tag{12.2}$$

and

$$\begin{aligned} \widehat{p}_0(\zeta) &:\equiv 1, \\ \widehat{p}_1(\zeta) &:= (\zeta - \alpha'_0) \widehat{p}_0(\zeta)/\gamma_0, \\ \widehat{p}_{n+1}(\zeta) &:= [(\zeta - \alpha'_n) \widehat{p}_n(\zeta) - \beta'_{n-1} \widehat{p}_{n-1}(\zeta)]/\gamma_n, \quad n = 1, \ldots, \nu' - 1, \end{aligned} \tag{12.3}$$

respectively. Both p_n and $\widehat{p}_n$ have exact degree n, and both have the leading coefficient Γ_n^{-1}, where

$$\Gamma_n := \gamma_0 \gamma_1 \cdots \gamma_{n-1}. \tag{12.4}$$

If $\gamma_n = -\varphi_n$ (for all n) as in (8.1) or, equivalently, if $\gamma_n = -\alpha_n - \beta_{n-1}$ (for all n) as in (4.12), then

$$p_n(0) = 1, \quad n = 0, 1, \ldots, \nu. \tag{12.5}$$

Otherwise, the values $\dot{\pi}_n := p_n(0)$ can be computed recursively according to (4.16) or (8.5).

Proof. The recurrences (4.16) and (8.5) follow from (12.2) and (12.1), respectively, by inserting $\zeta = 0$. It remains to verify the formulae (12.4) for the leading coefficients and (12.5) on the normalization at $\zeta = 0$. Both follow by induction from the recurrences. □

Since the Lanczos vector $\mathbf{y}_n$ is the nth residual of the three consistent versions of the BiCG method that we discussed in Sections 4, 8 and 9, p_n is for each of these algorithms the so-called *residual polynomial*. Property (12.5) is the standard consistency condition for residual polynomials of Krylov space solvers.

For the general Krylov space solver that we briefly considered in Section 4, the recurrence relations (4.10) for the residuals also imply recurrence relations for the residual polynomials, namely, in shorthand notation,

$$\zeta [\, p_0 \ \cdots \ p_{n-1} \,] = [\, p_0 \ \cdots \ p_{n-1} \ \ p_n \,] \underline{\mathbf{H}}_n. \tag{12.6}$$

From this formula it is easy to see that the condition $p_k(0) = 1$ $(k \le n)$ is equivalent to the zero column sum condition (4.11) for $\underline{\mathbf{H}}_n$ and the choice of $p_0(\zeta) \equiv 1$ as the constant polynomial. In analogy to (12.6) the recurrences (12.2) and (12.3) can be written as

$$\zeta [\ p_0 \ \cdots \ p_{n-1}\ ] = [\ p_0 \ \cdots \ p_{n-1} \ p_n\ ] \underline{\mathbf{T}}_n,$$
$$\zeta [\ \widehat{p}_0 \ \cdots \ \widehat{p}_{n-1}\ ] = [\ \widehat{p}_0 \ \cdots \ \widehat{p}_{n-1} \ \widehat{p}_n\ ] \underline{\mathbf{T}}'_n.$$

Let us now define on the linear space $\mathcal{P}$ of all polynomials two linear functionals Φ and Φ'. They are specified by the values they take on the monomial basis:

$$\Phi(\zeta^k) := \mu_k, \quad \Phi'(\zeta^k) := \mu_{k+1} \quad (k \in \mathbb{N}), \tag{12.7}$$

where

$$\mu_k := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}^k \mathbf{y}_0 \rangle_{\mathbf{B}} = \widetilde{\mathbf{y}}_0^\star \mathbf{B} \mathbf{A}^k \mathbf{y}_0. \tag{12.8}$$

Here μ_k is called the kth *moment* associated with $\mathbf{A}$, $\mathbf{y}_0$, and $\widetilde{\mathbf{y}}_0$. In engineering, the set of moments is referred to as *impulse response* or as the set of *Markov parameters* of a system; in the older mathematical literature they are sometimes called the *Schwarz constants* (Rutishauser 1957).

Note that for arbitrary polynomials s and t and corresponding Krylov space vectors $s(\mathbf{A})\mathbf{y}_0$, $t(\mathbf{A}^\star)\widetilde{\mathbf{y}}_0$ we have

$$\begin{aligned} \langle t(\mathbf{A}^\star)\widetilde{\mathbf{y}}_0, s(\mathbf{A})\mathbf{y}_0 \rangle_{\mathbf{B}} &= \Phi(ts), \\ \langle t(\mathbf{A}^\star)\widetilde{\mathbf{y}}_0, \mathbf{A}s(\mathbf{A})\mathbf{y}_0 \rangle_{\mathbf{B}} &= \Phi'(ts) = \Phi(\zeta ts). \end{aligned} \tag{12.9}$$

When we represent s and t by their coefficients,

$$s(\zeta) =: \sum \sigma_k \zeta^k, \quad t(\zeta) =: \sum \tau_k \zeta^k, \tag{12.10}$$

and introduce the infinite coefficient vectors (extended with zeros),

$$\mathbf{s} := [\ \sigma_0 \ \ \sigma_2 \ \ \sigma_3 \ \ \cdots\ ]^\top, \quad \mathbf{t} := [\ \tau_0 \ \ \tau_2 \ \ \tau_3 \ \ \cdots\ ]^\top \tag{12.11}$$

as well as the infinite *moment matrices*

$$\mathbf{M} := \begin{bmatrix} \mu_0 & \mu_1 & \mu_2 & \cdots \\ \mu_1 & \mu_2 & & \\ \mu_2 & & & \\ \vdots & & & \end{bmatrix}, \quad \mathbf{M}' := \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 & \cdots \\ \mu_2 & \mu_3 & & \\ \mu_3 & & & \\ \vdots & & & \end{bmatrix}$$

associated with the functionals Φ and Φ', we see from

$$\Phi(ts) = \sum_k \sum_l \tau_k \sigma_k \Phi(\zeta^{k+l})$$

and from the analogous formula for Φ' that

$$\Phi(ts) = \mathbf{t}^\top \mathbf{M} \mathbf{s}, \quad \Phi'(ts) = \mathbf{t}^\top \mathbf{M}' \mathbf{s}. \tag{12.12}$$

The matrices $\mathbf{M}$ and $\mathbf{M}'$ have Hankel structure: the (k,l)-element only depends on $k+l$. This is a consequence of the fact that the product ζ^{k+l} of ζ^k and ζ^l only depends on $k+l$.

What can we say about the relationship between the functional Φ and the original data of the problem: the matrices $\mathbf{A}$ and $\mathbf{B}$ and the initial vectors $\mathbf{y}_0$ and $\tilde{\mathbf{y}}_0$? (Recall that we are mainly interested in the case $\mathbf{B} = \mathbf{I}$, where we can forget $\mathbf{B}$.)

To explore this relationship, let us for a few lines assume that the matrices $\mathbf{A}$ and $\mathbf{B}$ are diagonalizable. Then, since they commute by assumption, they have a common complete system of eigenvectors (Wilkinson 1965, p. 52): there is a nonsingular $N \times N$ matrix $\mathbf{W}$ of eigenvectors such that

$$\mathbf{AW} = \mathbf{WD}_\lambda, \quad \mathbf{BW} = \mathbf{WD}_\kappa$$

where $\mathbf{D}_\lambda$ and $\mathbf{D}_\kappa$ are diagonal matrices containing the eigenvalues $\lambda_1, \ldots, \lambda_N$ of $\mathbf{A}$ and the eigenvalues $\kappa_1, \ldots, \kappa_N$ of $\mathbf{B}$, respectively. From $\mathbf{A}^\star \mathbf{W}^{-\star} = \mathbf{W}^{-\star}\mathbf{D}_\lambda^\star$ it is clear that the columns of $\mathbf{W}^{-\star}$ are a set of eigenvectors of $\mathbf{A}^\star$. We let

$$\mathbf{W} =: [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N], \quad \mathbf{W}^{-\star} =: [\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2, \ldots, \tilde{\mathbf{w}}_N],$$

and represent $\mathbf{y}_0$ in the basis $\{\mathbf{w}_k\}$, $\tilde{\mathbf{y}}_0$ in the basis $\{\tilde{\mathbf{w}}_k\}$:

$$\mathbf{y}_0 =: \sum_{j=1}^N \mathbf{w}_j \eta_j = \mathbf{W} \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_N \end{bmatrix}, \quad \tilde{\mathbf{y}}_0 =: \sum_{j=1}^N \tilde{\mathbf{w}}_j \tilde{\eta}_j = \mathbf{W}^{-\star} \begin{bmatrix} \tilde{\eta}_1 \\ \vdots \\ \tilde{\eta}_N \end{bmatrix}.$$

Then

$$\mu_k := \langle \tilde{\mathbf{y}}_0, \mathbf{A}^k \mathbf{y}_0 \rangle_{\mathbf{B}} = [\, \overline{\tilde{\eta}_1} \quad \ldots \quad \overline{\tilde{\eta}_N} \,] \mathbf{D}_\kappa \mathbf{D}_\lambda^k \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_N \end{bmatrix} = \sum_{j=1}^N \lambda_j^k \kappa_j \eta_j \overline{\tilde{\eta}_j}.$$

Therefore, μ_k can formally be written as the kth moment of a *discrete measure* $d\mu(\lambda)$ with masses $\kappa_j \eta_j \overline{\tilde{\eta}_j}$ at the points λ_j:

$$\mu_k = \int \lambda^k d\mu(\lambda), \quad \text{where } d\mu(\lambda) := \sum_{j=1}^N \kappa_j \eta_j \overline{\tilde{\eta}_j}\, \delta(\lambda - \lambda_j) d\lambda.$$

(In the last formula, δ is the Dirac function. Of course, if some of the eigenvalues coincide, the corresponding masses have to be added.) More generally,

$$\Phi(s) = \langle \tilde{\mathbf{y}}_0, s(\mathbf{A}) \mathbf{y}_0 \rangle_{\mathbf{B}} = \int s(\lambda) d\mu(\lambda).$$

In the symmetric case ($\mathbf{A} = \mathbf{A}^\star$, $\tilde{\mathbf{y}}_0 = \mathbf{y}_0$), where $\lambda_j \in \mathbb{R}$ and $\tilde{\eta}_j = \eta_j$, $d\mu(\lambda)$ is indeed a discrete positive measure whose support consists of the eigenvalues λ_j that are represented in $\mathbf{y}_0$ (*i.e.*, for which $\eta_j \neq 0$).

By the induced mapping from the Krylov space to polynomials, the biorthogonality (2.2) of the Lanczos vectors and the biconjugacy (7.3) of the sequences $\{\mathbf{v}_n\}$ and $\{\tilde{\mathbf{v}}_n\}$ now take the form

$$\tilde{\mathbf{y}}_0^\star \mathbf{B} p_m(\mathbf{A}) p_n(\mathbf{A}) \mathbf{y}_0 = \delta_{mn}\delta_n, \quad n = 0, 1, \ldots, \nu,$$

and

$$\tilde{\mathbf{y}}_0^\star \mathbf{B} \widehat{p}_m(\mathbf{A}) \mathbf{A} \widehat{p}_n(\mathbf{A}) \mathbf{y}_0 = \delta_{mn}\delta'_n, \quad n = 0, 1, \ldots, \nu',$$

respectively, in view of $\mathbf{AB} = \mathbf{BA}$. With the above definitions, we can translate these conditions into

$$\Phi(p_m p_n) = \delta_{mn}\delta_n, \quad m, n = 0, 1, \ldots, \nu,$$

$$\Phi'(\widehat{p}_m \widehat{p}_n) = \delta_{mn}\delta'_n, \quad m, n = 0, 1, \ldots, \nu'.$$

From the linearity of Φ and Φ' one concludes further that

$$\Phi(\zeta^m p_n) = \begin{cases} 0 & \text{if } 0 \le m < n, \\ \delta_n \Gamma_n & \text{if } m = n, \end{cases} \tag{12.13}$$

and

$$\Phi'(\zeta^m \widehat{p}_n) = \begin{cases} 0 & \text{if } 0 \le m < n, \\ \delta'_n \Gamma_n & \text{if } m = n, \end{cases} \tag{12.14}$$

which is another way to characterize these polynomial sequences. At this point we need to recall the following definition; see, for instance, Gutknecht (1992).

Definition 12.1 If the polynomial p_n of exact degree n satisfies (12.13) for some $\delta_n \neq 0$ and is uniquely determined by these conditions, it is called a *regular formal orthogonal polynomial (FOP)* of the functional Φ.

In other words, (12.13) and (12.14) mean that $\{p_n\}_{n=0}^{\nu}$ is a sequence of regular FOPs of the functional Φ, and that $\{\widehat{p}_n\}_{n=0}^{\nu'}$ is such a sequence for Φ'.

If p_n and $\widehat{p}_n$ are expressed in the monomial basis,

$$p_n(\zeta) =: \sum_{k=0}^{n} \pi_k^{(n)} \zeta^k,$$

and

$$\widehat{p}_n(\zeta) =: \sum_{k=0}^{n} \widehat{\pi}_k^{(n)} \zeta^k,$$

where $\pi_n^{(n)} = \widehat{\pi}_n^{(n)} = \Gamma_n^{-1}$, then the first n equations of (12.13) and (12.14) become homogeneous $n \times (n+1)$ linear systems for the coefficients $\pi_k^{(n)}$ and $\widehat{\pi}_k^{(n)}$, respectively. Since $\pi_n^{(n)}$ and $\widehat{\pi}_n^{(n)}$ are known, we can move them on the

right-hand side:

$$\mathbf{M}_n \begin{bmatrix} \pi_0^{(n)} \\ \pi_1^{(n)} \\ \vdots \\ \pi_{n-1}^{(n)} \end{bmatrix} = - \begin{bmatrix} \mu_n \\ \mu_{n+1} \\ \vdots \\ \mu_{2n-1} \end{bmatrix} \pi_n^{(n)} \text{ with } \mathbf{M}_n := \begin{bmatrix} \mu_0 & \mu_1 & \cdots & \mu_{n-1} \\ \mu_1 & \mu_2 & & \mu_n \\ \vdots & & & \vdots \\ \mu_{n-1} & \mu_n & \cdots & \mu_{2n-2} \end{bmatrix}, \tag{12.15}$$

$$\mathbf{M}'_n \begin{bmatrix} \widehat{\pi}_0^{(n)} \\ \widehat{\pi}_1^{(n)} \\ \vdots \\ \widehat{\pi}_{n-1}^{(n)} \end{bmatrix} = - \begin{bmatrix} \mu_{n+1} \\ \mu_{n+2} \\ \vdots \\ \mu_{2n} \end{bmatrix} \widehat{\pi}_n^{(n)} \text{ with } \mathbf{M}'_n := \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_n \\ \mu_2 & \mu_3 & & \mu_{n+1} \\ \vdots & & & \vdots \\ \mu_n & \mu_{n+1} & \cdots & \mu_{2n-1} \end{bmatrix}.$$

In the engineering literature, linear systems of this form are called *Yule–Walker equations.* The matrices $\mathbf{M}_n$ and $\mathbf{M}'_n$ are the nth leading principal submatrices of the infinite moment matrices $\mathbf{M}$ and $\mathbf{M}'$. Since (12.13) and (12.14) are equivalent to (2.2) and (7.3), respectively, it follows from the uniqueness statements in Theorem 2.1 and Corollary 9.1 that for prescribed leading coefficients $\pi_n^{(n)}$ and $\widehat{\pi}_n^{(n)}$ the solutions of these linear systems are uniquely determined as long as $1 \leq n \leq \nu$ in the first and $1 \leq n \leq \nu'$ in the second system. Clearly, this existence and uniqueness statement is equivalent to the nonsingularity of the matrices $\mathbf{M}_n$ and $\mathbf{M}'_n$ in the respective range of indices. For future use we state part of this result so that it also applies for $n > \nu$ and $n > \nu'$, respectively.

Lemma 12.2 A Lanczos polynomial p_n of exact degree n that is a regular FOP of Φ (that is, which satisfies the orthogonality conditions $\Phi(\zeta^m p_n) = 0$ $(0 \leq m < n)$ and is up to a scalar multiple uniquely determined by them) exists if and only if $\mathbf{M}_n$ is nonsingular.

Likewise, a direction polynomial $\widehat{p}_n$ of exact degree n that is a regular FOP of Φ' (that is, which satisfies the orthogonality conditions $\Phi'(\zeta^m \widehat{p}_n) = 0$ $(0 \leq m < n)$ and is up to a scalar multiple uniquely determined by them) exists if and only if $\mathbf{M}'_n$ is nonsingular.

In particular, the indices of first breakdown or termination, ν and ν', of the BIO algorithm and the BIC algorithm (Algorithms 1 and 9 in Sections 2 and 9), respectively, satisfy

$$\begin{aligned} \nu &= \min \{n : \mathbf{M}_{n+1} \text{ singular}\}, \\ \nu' &= \min \{n : \mathbf{M}'_{n+1} \text{ singular}\}. \end{aligned}$$

Proof. It remains to show that $\mathbf{M}_{\nu+1}$ and $\mathbf{M}'_{\nu'+1}$ are singular. Set $n := \nu$. First, for given $\pi_n^{(n)} \neq 0$, the linear system (12.15) uniquely determines the coefficients of a polynomial p_n that corresponds to a vector $\mathbf{y}_n \in \mathcal{K}_{n+1}$

satisfying $\widetilde{\mathcal{K}}_n \perp_{\mathbf{B}} \mathbf{y}_n$. At the same time, $\overline{p_n}$ is mapped into $\widetilde{\mathbf{y}}_n \in \widetilde{\mathcal{K}}_{n+1}$ satisfying $\widetilde{\mathbf{y}}_n \perp_{\mathbf{B}} \mathcal{K}_n$. We want to prove that, in contrast to the definition of ν, the conditions (2.1)–(2.2) could be fulfilled up to $\nu+1$ if $\mathbf{M}_{\nu+1}$ were nonsingular. It remains to show that $\langle \widetilde{\mathbf{y}}_n, \mathbf{y}_n \rangle_{\mathbf{B}} = 0$ unless $\mathbf{M}_{\nu+1}$ is nonsingular. In fact, if this inner product is 0, then the additional equation $\mu_n \pi_0^{(n)} + \cdots + \mu_{2n-1}\pi_{n-1}^{(n)} + \mu_{2n}\pi_n^{(n)} = 0$ holds, which extends system (12.15) by an extra row at the bottom. Consequently, the coefficients of p_n satisfy $\mathbf{M}_{\nu+1}[\ \pi_0^{(n)} \ \cdots \ \pi_n^{(n)}\ ]^\top = \mathbf{o}$, which implies that $\mathbf{M}_{\nu+1}$ is singular.

Of course, the singularity of $\mathbf{M}'_{\nu'+1}$ is shown the same way. □

If the consistency condition (12.5) holds, then $\pi_0^{(n)} = 1$ and we can move the first column of $\mathbf{M}_n$ to the right-hand side of (12.15), in exchange for the current right-hand side that is moved back to the left. In this way, $\mathbf{M}_n$ is replaced by $\mathbf{M}'_n$, and the system becomes

$$\mathbf{M}'_n \begin{bmatrix} \pi_1^{(n)} \\ \pi_2^{(n)} \\ \vdots \\ \pi_n^{(n)} \end{bmatrix} = - \begin{bmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_{n-1} \end{bmatrix}. \tag{12.16}$$

Lemma 12.3 A residual polynomial p_n of exact degree n that satisfies the consistency condition (12.5) and the orthogonality conditions $\Phi(\zeta^m p_n) = 0$ $(0 \le m < n)$ and is uniquely determined by the latter exists if and only if $\mathbf{M}_n$ and $\mathbf{M}'_n$ are nonsingular.

In particular, if $\dot{\nu}$ denotes the index of first breakdown or termination of the BIORES and the BIOMIN algorithm (Algorithms 2 and 7 in Sections 4 and 8, then

$$\dot{\nu} = \min \{n : \mathbf{M}_{n+1} \text{ or } \mathbf{M}'_{n+1} \text{ singular}\}.$$

Proof. From the previous Lemma we know that an essentially unique p_n satisfying the orthogonality conditions exists if and only if $\mathbf{M}_n$ is nonsingular. In order that p_n can be normalized by $p_n(0) = 1$, we need $\pi_n^{(n)} \neq 0$; then the normalized coefficients satisfy (12.16). If this system had more than one solution, it would have infinitely many solutions with $\pi_n^{(n)} \neq 0$. Renormalizing them, we would find infinitely many solutions of (12.15) with $\pi_n^{(n)} = 1$, in contrast to the nonsingularity of $\mathbf{M}_n$.

Conversely, if both matrices are nonsingular, then the residual polynomial with the stated properties clearly exists and is uniquely determined. □

The above derivation of Lemma 12.2 relies on the existence and uniqueness statements in Theorem 2.1 and Corollary 9.1, which describe the construction of the Krylov space vectors $\mathbf{y}_n$ and $\mathbf{v}_n$ that are the images of the polynomials p_n and $\widehat{p}_n$. There is another, more direct approach to this

lemma. Since $\mathbf{M}_n$ and $\mathbf{M}'_n$ are nonsingular for $1 \le n \le \nu$ and $1 \le n \le \nu'$, respectively, the matrices $\mathbf{M}_\nu$ and $\mathbf{M}'_{\nu'}$ have LDU decompositions (without pivoting). In view of the Hankel structure, $\mathbf{M}_\nu$ and $\mathbf{M}'_{\nu'}$ are real or complex symmetric, so that these LDU decompositions are symmetric ones. It turns out that the upper triangular factors contain in their columns the coefficients of the polynomials p_m and $\widehat{p}_m$, respectively. In fact, let

$$\mathbf{P}_n := \begin{bmatrix} \pi_0^{(0)} & \pi_0^{(1)} & \cdots & \pi_0^{(n)} \\ & \pi_1^{(1)} & & \pi_1^{(n)} \\ & & \ddots & \vdots \\ & & & \pi_n^{(n)} \end{bmatrix}, \quad \widehat{\mathbf{P}}_n := \begin{bmatrix} \widehat{\pi}_0^{(0)} & \widehat{\pi}_0^{(1)} & \cdots & \widehat{\pi}_0^{(n)} \\ & \widehat{\pi}_1^{(1)} & & \widehat{\pi}_1^{(n)} \\ & & \ddots & \vdots \\ & & & \widehat{\pi}_n^{(n)} \end{bmatrix},$$

and, as before, let $\mathbf{Y}_n$ and $\widetilde{\mathbf{Y}}_n$ be the $N \times n$ matrices with columns $\mathbf{y}_m$ and $\widetilde{\mathbf{y}}_m$, $m = 0, 1, \ldots, n-1$, respectively, so that

$$\begin{aligned} \mathbf{Y}_n &= [\ \mathbf{y}_0 \quad \mathbf{A}\mathbf{y}_0 \quad \ldots \quad \mathbf{A}^{n-1}\mathbf{y}_0\ ]\,\mathbf{P}_n, \\ \widetilde{\mathbf{Y}}_n &= [\ \widetilde{\mathbf{y}}_0 \quad \mathbf{A}^\star\widetilde{\mathbf{y}}_0 \quad \ldots \quad (\mathbf{A}^\star)^{n-1}\widetilde{\mathbf{y}}_0\ ]\,\overline{\mathbf{P}_n}. \end{aligned} \tag{12.17}$$

(We still assume that $\widetilde{\gamma}_n = \overline{\gamma_n}$ for all n.) By the definition of the moments,

$$\begin{bmatrix} \widetilde{\mathbf{y}}_0^\star \\ \widetilde{\mathbf{y}}_0^\star\mathbf{A} \\ \vdots \\ \widetilde{\mathbf{y}}_0^\star\mathbf{A}^{n-1} \end{bmatrix} \mathbf{B}\,[\ \mathbf{y}_0 \quad \mathbf{A}\mathbf{y}_0 \quad \ldots \quad \mathbf{A}^{n-1}\mathbf{y}_0\ ] = \mathbf{M}_n.$$

Inserting here (12.17) and the orthogonality property $\widetilde{\mathbf{Y}}_n^\star\mathbf{B}\mathbf{Y}_n = \mathbf{D}_{\delta;n}$ (see (2.24)) yields

$$\mathbf{P}_n^{-\top}\,\mathbf{D}_{\delta;n}\,\mathbf{P}_n^{-1} = \mathbf{M}_n \quad (n \le \nu), \tag{12.18}$$

and likewise we find

$$\widehat{\mathbf{P}}_n^{-\top}\,\mathbf{D}_{\delta';n}\,\widehat{\mathbf{P}}_n^{-1} = \mathbf{M}'_n \quad (n \le \nu). \tag{12.19}$$

These are the claimed (symmetric) LDU decompositions of the nth moment submatrices $\mathbf{M}_n$ and $\mathbf{M}'_n$ if the polynomials are monic, that is, $\gamma_n = \widetilde{\gamma}_n = 1$ (for all n). Otherwise we get a nonstandard symmetric LDU decompositions with prescribed diagonal elements of the triangular factors. $\mathbf{P}_n^{-1}$ and $\widehat{\mathbf{P}}_n^{-1}$ are the upper triangular matrices that contain the coefficients of the monomials when expressed as linear combinations of the Lanczos and the direction polynomials, respectively. Of course, if the decompositions (12.18) and (12.19) exist for $n = \nu$ and $n = \nu'$, respectively, and if the diagonal matrices $\mathbf{D}_{\delta;\nu}$ and $\mathbf{D}_{\delta';n}$ are nonsingular, then these decompositions exist for all n in the range specified. On the other hand, the latter holds if and only if the leading principal submatrices $\mathbf{M}_n$ and $\mathbf{M}'_n$ are all nonsingular. Moreover, we can conclude conversely that the inverses of the triangular

factors contain the coefficients of monic FOPs p_m $(m = 0, 1, \ldots, \nu - 1)$ associated with Φ and of monic FOPs $\widehat{p}_m$ $(m = 0, 1, \ldots, \nu' - 1)$ associated with Φ', respectively. Hence, the singularity of $\mathbf{M}_{\nu+1}$ and $\mathbf{M}'_{\nu'+1}$ as well as the other statements of Lemma 12.2 follow.

13. The Lanczos process for polynomials: the Stieltjes procedure

So far we have considered the Lanczos process as a tool for generating biorthogonal bases for a pair of Krylov spaces of the linear operators $\mathbf{A}$ and $\mathbf{A}^\star$ defined on $\mathbb{C}^N$ (or, more generally, a linear space and its dual space). In the symmetric case (that is, when $\mathbf{A}^\star = \mathbf{A}$ and $\widetilde{\mathbf{y}}_0 = \mathbf{y}_0$) the two bases coincide and are orthogonal. We can apply the same process to the multiplication operator M defined on the space $\mathcal{P}$ of all polynomials with complex coefficients:

$$M : s(\zeta) \mapsto \zeta\, s(\zeta) \quad (s \in \mathcal{P}).$$

We aim for a setting where the dual space is also $\mathcal{P}$. In the case of classical (real) orthogonal polynomials an inner product defined on $\mathcal{P} \times \mathcal{P}$ is given by some bilinear integral operator:

$$\langle t, s\rangle := \int t(\zeta)s(\zeta)d\mu(\zeta) =: \Phi(ts), \quad (t, s) \in \mathcal{P} \times \mathcal{P}. \tag{13.1}$$

Here $d\mu$ is a positive measure whose support is a subset of the real line. Note that the integral only depends on the product ts, and therefore the inner product is a linear functional Φ of this product.

For the Lanczos process applied to polynomials we want to preserve this property, but to relax the assumption on Φ. This means, however, that in the complex case our formal inner product is still based on a bilinear functional, and not on a sesquilinear[15] one. Therefore it is better to forget the integral and just let

$$\langle t, s\rangle := \Phi(ts) \tag{13.2}$$

with an arbitrary complex linear functional Φ defined on $\mathcal{P}$, which may, but need not be, defined by its moments μ_k, the values it takes on the monomials:

$$\Phi(\zeta^k) := \mu_k$$

This exhibits the connection to the Lanczos process for an operator $\mathbf{A}$, where the moments are given by (12.8). When we represent s and t by their coefficient vectors, as in (12.10)–(12.11), the inner product is still given by (12.12).

[15] A *sesquilinear* functional $\langle ., .\rangle$ on $\mathcal{P} \times \mathcal{P}$ is one for which $\langle \alpha t, \beta s\rangle = \overline{\alpha}\beta\langle t, s\rangle$.

Note that as a consequence of (13.2) we have

$$M^\star = M.$$

However, this does not mean that M is a selfadjoint operator, unless (13.2) is really an inner product, which is not true in general, even in the real case, as we do not require that $\langle s, s\rangle = \Phi(s^2) > 0$ if $s \neq 0$.

It is now an easy matter to reformulate the Lanczos process, be it in the BIO or BIOC form, as a recursive process for generating *formal orthogonal polynomials (FOPs)*. For the BIO form, we start with $p_0(\zeta) \equiv 1$; in the nth step, we apply the multiplication operator to p_{n-1} and then 'orthogonalize' the product $\zeta p(\zeta)$ with respect to p_{n-1} and p_{n-2} to get the new member p_n of the sequence. Here 'orthogonalize' refers to the formal inner product (13.2). Again it is seen that the orthogonality with respect to polynomials of lower degree follows automatically. This construction leads exactly to the three-term recurrence (12.2). A similar one that parallels the BIOC algorithm yields the coupled two-term recurrences (12.1).

Lanczos (1952) was well aware of this form of his process, but, at least for classical orthogonal polynomials, it was published long before by Stieltjes (1884). This polynomial version of the Lanczos process is therefore called the *Stieltjes procedure*.

It remains to give formulae for the recurrence coefficients in terms of values of Φ. Since Φ', which appears in the BIOC form, is linked to Φ by

$$\Phi'(s) = \Phi(\zeta s) \quad (s \in \mathcal{P})$$

(see (12.7)), it is easily substituted. As mentioned before, when the aim is the construction of orthogonal polynomials for a measure supported on a subset of the real line, Φ is the integral operator (13.1). In terms of Φ the BIO recurrence coefficients can be expressed as follows:

$$\alpha_n = \Phi(\zeta p_n^2)/\delta_n, \tag{13.3a}$$

$$\beta_{n-1} = \Phi(\zeta p_{n-1} p_n)/\delta_{n-1} = \gamma_{n-1}\delta_n/\delta_{n-1} \quad (\text{if } n > 0), \tag{13.3b}$$

$$\delta_{n+1} = \Phi(p_{n+1}^2). \tag{13.3c}$$

Of course, if the functional Φ is not definite, and thus can assume the value zero at p_{n+1}^2, the Stieltjes procedure can also break down.

For the BIOC version of the Stieltjes procedure we just need

$$\delta_{n+1} = \Phi(p_{n+1}^2), \tag{13.4a}$$

$$\delta'_{n+1} = \Phi(\zeta \widehat{p}_{n+1}^2), \tag{13.4b}$$

since ψ_n and φ_{n+1} are expressed in terms of these values.

14. The biconjugate gradient squared method

One of the disadvantages of the BiCG method is that it also requires matrix-vector multiplications with the transpose matrix $\mathbf{A}^\star$ or $\mathbf{A}^\top$, although the relevant Krylov space containing the residuals is generated only by $\mathbf{A}$. In practice, $\mathbf{A}$ is typically large and sparse, and providing an efficient subroutine for both these products can be a nontrivial task. Moreover, since two Krylov spaces are generated, two matrix-vector products are needed per dimension of the subspaces $\mathcal{K}_n$ that matter.

In 1984 Sonneveld (1989) proposed a new Lanczos-type method that circumvents these two disadvantages and proved to be very successful in practice. He called it the *conjugate gradient squared* (CGS) *algorithm*, although it is aimed at nonsymmetric problems and is not derived from a CG, but from a BiCG algorithm, namely BiOMin. Its nth residual polynomial is the square p_n^2 of the nth residual polynomial p_n of BiCG.

Here Sonneveld's version will be called BiOMinS, but we refer to the underlying approach as the *biconjugate gradient squared* (BiCGS) *method*. As for the BiCG method, there exist several different forms of the BiCGS method: in addition to BiOMinS there are consistent and inconsistent versions of BiOResS and two version of BiODirS. The latter two are not really competitive, and therefore they are not discussed here. Together with the two BiOResS versions they were presented in a separately distributed part (Section 7) of Gutknecht (1990).

All competitive versions of the BiCGS method require two applications of the operator $\mathbf{A}$ at each step; this is comparable to the two matrix-vector multiplications with $\mathbf{A}$ and $\mathbf{A}^\star$ in BiOMin, but in many applications it is an advantage that the multiplication with $\mathbf{A}^\star$ is replaced by one with $\mathbf{A}$. For example, this is true when certain preconditioning techniques are applied, when ordinary differential equations are solved with the help of the Lanczos process (see, for instance, Hochbruck and Lubich (1997*b*)), or, quite generally, when vector and parallel computers are used.

The BiCGS method typically converges nearly twice as fast as the BiCG method. However, the convergence is even less smooth, and in tough problems very erratic: the norm of the residual can suddenly increase again by several orders of magnitude and then drop to the former level after just one or a few steps. Such peaks in the residual norm plot indicate a reduction of the ultimate accuracy of the solution that can be attained, but in Section 18 we will describe a remedy for this loss.

14.1. The BIOMINS *form of the* BICGS *method*

For the derivation of BIOMINS we start from the recurrences (12.1), choosing $\gamma_n := -\varphi_n$ and substituting $\omega_n := 1/\varphi_n$, as in BIOMIN:

$$p_{n+1} := p_n - \omega_n \zeta \widehat{p}_n \,, \tag{14.1}$$

$$\widehat{p}_{n+1} := p_{n+1} - \psi_n \widehat{p}_n \,. \tag{14.2}$$

(In this section we use the sloppy notation $\zeta\widehat{p}_n$ instead of $\zeta\widehat{p}_n(\zeta)$.) Multiplying (14.1) by $\widehat{p}_n$ and (14.2) by p_{n+1} yields

$$p_{n+1}\widehat{p}_n = p_n\widehat{p}_n - \omega_n \zeta \widehat{p}_n^2, \tag{14.3}$$

$$p_{n+1}\widehat{p}_{n+1} = p_{n+1}^2 - \psi_n p_{n+1}\widehat{p}_n. \tag{14.4}$$

Next, squaring both sides of (14.1) and (14.2) and using (14.3) and (14.4), respectively, leads to

$$p_{n+1}^2 = p_n^2 - 2\omega_n\zeta p_n\widehat{p}_n + \omega_n^2\zeta^2\widehat{p}_n^2 \tag{14.5a}$$

$$= p_n^2 - \omega_n\zeta(p_n\widehat{p}_n + p_{n+1}\widehat{p}_n) \tag{14.5b}$$

and

$$\begin{aligned}\widehat{p}_{n+1}^2 &= p_{n+1}^2 - 2\psi_n p_{n+1}\widehat{p}_n + \psi_n^2\widehat{p}_n^2 \\ &= p_{n+1}\widehat{p}_{n+1} - \psi_n p_{n+1}\widehat{p}_n + \psi_n^2\widehat{p}_n^2.\end{aligned} \tag{14.6}$$

The point is that equations (14.3), (14.5b), (14.6), and (14.4) are, in this order, a system of recurrence relations for the four polynomial sequences $\{p_n\widehat{p}_{n-1}\}$, $\{p_n^2\}$, $\{\widehat{p}_n^2\}$, and $\{p_n\widehat{p}_n\}$. From (13.4a)–(13.4b) it follows that the recurrence coefficients can be computed from the values that the functional Φ defined by (12.7) takes at the polynomials $\zeta\widehat{p}_n^2$, p_{n+1}^2, and p_n^2. Here we need to express these values in terms of the new Krylov space vectors

$$\begin{aligned}\mathbf{r}_n &:= p_n^2(\mathbf{A})\mathbf{r}_0, & \widehat{\mathbf{r}}_n &:= \widehat{p}_n^2(\mathbf{A})\mathbf{r}_0, \\ \mathbf{s}_n &:= p_n(\mathbf{A})\widehat{p}_n(\mathbf{A})\mathbf{r}_0, & \mathbf{s}'_n &:= p_{n+1}(\mathbf{A})\widehat{p}_n(\mathbf{A})\mathbf{r}_0\end{aligned} \tag{14.7}$$

and their inner products with an additional vector $\widetilde{\mathbf{y}}_0$, which is now only used for these inner products. In his seminal paper, Sonneveld (1989) chose $\widetilde{\mathbf{y}}_0 := \mathbf{r}_0$, but today it is known that a random vector is likely to yield better convergence; see, for instance, Joubert (1990, 1992).

Once we have written down the recurrences for the four vector sequences of (14.7), we have managed to 'square' a special case of the BIOC algorithm. Like BIOMIN, the BIOMINS algorithm is then based on the fact that one can additionally compute a vector sequence $\{\mathbf{x}_n\}$ with the property that $\mathbf{r}_n$ is the residual at $\mathbf{x}_n$. In summary, we obtain the following standard BIOMIN version of the BICGS method.

ALGORITHM 11. (BIOMINS FORM OF THE BICGS METHOD)
For solving $\mathbf{Ax} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0 \in \mathbb{C}^N$ and set $\mathbf{s}_0 := \mathbf{r}_0 := \widehat{\mathbf{r}}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$. Choose $\widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{r}_0 \rangle_{\mathbf{B}} \neq 0$ and $\delta_0' := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{r}}_0 \rangle_{\mathbf{B}} \neq 0$. Then compute for $n = 0, 1, \ldots$

$$\begin{aligned}
\omega_n &:= \delta_n / \delta_n', && \text{(14.8a)}\\
\mathbf{s}_n' &:= \mathbf{s}_n - \mathbf{A}\widehat{\mathbf{r}}_n \omega_n, && \text{(14.8b)}\\
\mathbf{r}_{n+1} &:= \mathbf{r}_n - \mathbf{A}(\mathbf{s}_n + \mathbf{s}_n')\,\omega_n, && \text{(14.8c)}\\
\mathbf{x}_{n+1} &:= \mathbf{x}_n + (\mathbf{s}_n + \mathbf{s}_n')\,\omega_n, && \text{(14.8d)}\\
\delta_{n+1} &:= \langle \widetilde{\mathbf{y}}_0, \mathbf{r}_{n+1} \rangle_{\mathbf{B}}\,, && \text{(14.8e)}\\
\psi_n &:= -\delta_{n+1}/\delta_n, && \text{(14.8f)}\\
\mathbf{s}_{n+1} &:= \mathbf{r}_{n+1} - \mathbf{s}_n' \psi_n, && \text{(14.8g)}\\
\widehat{\mathbf{r}}_{n+1} &:= \mathbf{s}_{n+1} - \mathbf{s}_n' \psi_n + \widehat{\mathbf{r}}_n \psi_n^2, && \text{(14.8h)}\\
\delta_{n+1}' &:= \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{r}}_{n+1} \rangle_{\mathbf{B}}\,. && \text{(14.8i)}
\end{aligned}$$

If $\mathbf{r}_{n+1} = \mathbf{o}$, the process terminates and $\mathbf{x}_{n+1}$ is the solution; if $\mathbf{r}_{n+1} \neq \mathbf{o}$ but $\delta_{n+1} = 0$ or $\delta_{n+1}' = 0$, the algorithm breaks down. In each case we set $\dot{\nu} := n + 1$.

The recurrences for the vectors $\mathbf{s}_n$, $\mathbf{s}_n'$, $\mathbf{r}_n$, and $\widehat{\mathbf{r}}_n$ are direct translations of equations (14.3)–(14.6), and the formulae for δ_{n+1} and δ_{n+1}' are in view of (12.9) and (14.7) equivalent to (13.4a)–(13.4b). Finally, the recurrence for $\mathbf{x}_n$ is chosen so that $\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$ (for all n): by (14.8d), (14.8c), and by induction we indeed get $\mathbf{r}_{n+1} = \mathbf{r}_n - \mathbf{A}(\mathbf{x}_{n+1} - \mathbf{x}_n) = \mathbf{b} - \mathbf{A}\mathbf{x}_{n+1}$.

From our derivation of this algorithm it is clear that the following holds.

Theorem 14.1 If BIOMIN and BIOMINS are started with the same $\mathbf{x}_0$ and $\widetilde{\mathbf{y}}_0$, the index of first breakdown or termination $\dot{\nu}$, the recurrence coefficients φ_n, ψ_{n-1}, and the inner products δ_n are for both methods the same. The residual polynomials are p_n and p_n^2, respectively.

Although, theoretically, if convergence were defined by $\mathbf{r}_n = \mathbf{o}$ exactly, the two algorithms would converge or break down at the same step, it is evident that in practice, where convergence is defined by a condition like $||\mathbf{r}_n|| \leq \epsilon ||\mathbf{r}_0||$, BIOMINS converges normally faster than BIOMIN, since $|p_n^2(\zeta)| = |p_n(\zeta)|^2 < |p_n(\zeta)|$ if the latter is smaller than 1.

Each step requires two applications of the operator $\mathbf{A}$, that is, two multiplications of $\mathbf{A}$ with a vector, namely $\mathbf{A}\widehat{\mathbf{r}}_n$ and $\mathbf{A}(\mathbf{s}_n + \mathbf{s}_n')$.

Since the coefficients $\omega_n = 1/\varphi_n$ and ψ_{n-1} are the same as in BIOMIN, the bidiagonal matrices $\mathbf{L}_n$ and $\mathbf{U}_n$ are again available, and thus the tridiagonal matrix $\mathbf{T}_n = \mathbf{L}_n \mathbf{U}_n$ may still be used to obtain eigenvalue estimates for $\mathbf{A}$. In fact, deleting (14.8d) from Algorithm 11, we get what one might call a special squared BIOC algorithm, and we could easily modify it to allow for

the freedom of choosing γ_n independently from φ_n, as in our formulation of BIOC.

14.2. BIOS*: the squared* BIO *algorithm*

At this point one may ask whether, in analogy to the other two main forms of the BICG method, BIORES and BIODIR, there are also other forms of the BICGS method. To derive the analogue of BIORES, we first need a squared BIO algorithm based on separate recurrences for the polynomials p_n^2 and $\widehat{p}_n^2$. We start with those for p_n^2: multiplying (12.2) by p_n and squaring (12.2) we obtain, respectively,

$$\gamma_n p_n p_{n+1} = (\zeta - \alpha_n) p_n^2 - \beta_{n-1} p_{n-1} p_n, \tag{14.9}$$

$$\begin{aligned} \gamma_n^2 p_{n+1}^2 &= (\zeta - \alpha_n)^2 p_n^2 - 2(\zeta - \alpha_n)\beta_{n-1} p_{n-1} p_n + \beta_{n-1}^2 p_{n-1}^2 \\ &= (\zeta - \alpha_n)(\gamma_n p_n p_{n+1} - \beta_{n-1} p_{n-1} p_n) + \beta_{n-1}^2 p_{n-1}^2 , \end{aligned} \tag{14.10}$$

where (14.9) has already been used to simplify (14.10). These two relations allow us to generate the two sequences $\{p_{n-1}p_n\}$ and $\{p_n^2\}$ recursively. The coefficients α_n, β_{n-1} are given by (13.3a)–(13.3c). So far, the parameters γ_n can be chosen freely ($\neq 0$), and this freedom persists if one aims at an inconsistent version of BIORESS. Later, we will want to choose $\gamma_n := -\alpha_n - \beta_{n-1}$ in the case of consistent BIORESS, since this condition is equivalent to $p_n(0) = 1$, which implies $p_n^2(0) = 1$, the consistency condition for the residual polynomial p_n^2. (The freedom of choosing the sign of $p_n(0)$ would not help to avoid any type of breakdown.)

In summary, we see that a method for generating

$$\mathbf{r}_n := p_n^2(\mathbf{A})\mathbf{r}_0, \quad \mathbf{r}'_n := p_n(\mathbf{A})p_{n+1}(\mathbf{A})\mathbf{r}_0 \tag{14.11}$$

can be based on (14.9), (14.10), and (13.3a)–(13.3c). This is the *squared biorthogonalization* or BIOS *algorithm.*

ALGORITHM 12. (BIOS ALGORITHM)
Choose $\mathbf{r}_0, \tilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle \tilde{\mathbf{y}}_0, \mathbf{r}_0 \rangle_{\mathbf{B}} \neq 0$, and set $\mathbf{r}'_{-1} := \mathbf{o} \in \mathbb{C}^N$, $\beta_{-1} := 0$. Choosing arbitrary scale factors $\gamma_n \neq 0$, compute, for $n = 0, 1, \ldots$,

$$\alpha_n := \langle \tilde{\mathbf{y}}_0, \mathbf{A}\mathbf{r}_n \rangle_{\mathbf{B}} / \delta_n, \tag{14.12a}$$

$$\beta_{n-1} := \gamma_{n-1}\delta_n / \delta_{n-1} \quad (\text{if } n > 0), \tag{14.12b}$$

$$\mathbf{r}'_n := [\mathbf{A}\mathbf{r}_n - \mathbf{r}_n\alpha_n - \mathbf{r}'_{n-1}\beta_{n-1}]/\gamma_n, \tag{14.12c}$$

$$\begin{aligned} \mathbf{r}_{n+1} := [&\mathbf{A}(\mathbf{r}'_n\gamma_n - \mathbf{r}'_{n-1}\beta_{n-1}) - (\mathbf{r}'_n\gamma_n - \mathbf{r}'_{n-1}\beta_{n-1})\alpha_n \\ &+ \mathbf{r}_{n-1}\beta_{n-1}^2]/\gamma_n^2, \end{aligned} \tag{14.12d}$$

$$\delta_{n+1} := \langle \tilde{\mathbf{y}}_0, \mathbf{r}_{n+1} \rangle_{\mathbf{B}} \tag{14.12e}$$

until $\delta_{n+1} = 0$, when we set $\nu := n + 1$.

Note that γ_n can be chosen to make $\mathbf{r}_n$ of unit length.

This algorithm was proposed independently both by Chronopoulos and Ma (1989) and by Gutknecht (1990); later it was rediscovered by Chan, de Pillis and van der Vorst (1991).

14.3. The BIORESS *forms of the* BICGS *method*

After we have 'squared' the BIO algorithm so that nested Krylov spaces of the double dimension are generated, it remains to find a way to compute the sequence $\{\mathbf{x}_n\}$ of approximants with the property that

$$\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$$

in the case of consistent BIORESS, or, more generally,

$$\mathbf{r}_n = \mathbf{b}\dot{\pi}_n^2 - \mathbf{A}\mathbf{x}_n, \tag{14.13}$$

where $\dot{\pi}_n := p_n(0)$ as before. This approach to solving a linear system of equations follows exactly the general scheme discussed in Section 4. Assuming that it works, we conclude from (14.12d) and (4.16) that

$$\begin{aligned}
&\mathbf{A}\mathbf{x}_{n+1}\gamma_n^2 \qquad\qquad (14.14)\\
&\quad = (\mathbf{b}\dot{\pi}_{n+1}^2 - \mathbf{r}_{n+1})\gamma_n^2\\
&\quad = \mathbf{b}(\beta_{n-1}^2\dot{\pi}_{n-1}^2 - \alpha_n\gamma_n\dot{\pi}_n\dot{\pi}_{n+1} + \alpha_n\beta_{n-1}\dot{\pi}_n\dot{\pi}_{n-1})\\
&\qquad - \mathbf{A}(\mathbf{r}'_n\gamma_n - \mathbf{r}'_{n-1}\beta_{n-1}) + (\mathbf{r}'_n\gamma_n - \mathbf{r}'_{n-1}\beta_{n-1})\alpha_n - \mathbf{r}_{n-1}\beta_{n-1}^2\\
&\quad = \mathbf{A}\mathbf{x}_{n-1}\beta_{n-1}^2 - \mathbf{A}\mathbf{x}'_n\alpha_n\gamma_n + \mathbf{A}\mathbf{x}'_{n-1}\alpha_n\beta_{n-1}\\
&\qquad - \mathbf{A}(\mathbf{r}'_n\gamma_n - \mathbf{r}'_{n-1}\beta_{n-1}), \qquad\qquad (14.15)
\end{aligned}$$

where

$$\mathbf{x}'_n := \mathbf{A}^{-1}(\mathbf{b}\dot{\pi}_n\dot{\pi}_{n+1} - \mathbf{r}'_n)\,,$$

that is,

$$\mathbf{r}'_{n-1} = \mathbf{b}\dot{\pi}_{n-1}\dot{\pi}_n - \mathbf{A}\mathbf{x}'_{n-1}\,. \tag{14.16}$$

Multiplying (14.15) by $\mathbf{A}^{-1}$ yields a recursive formula for $\mathbf{x}_{n+1}$. Similarly, using (14.12c) and (4.16) we get

$$\begin{aligned}
\mathbf{A}\mathbf{x}'_n\gamma_n &= \mathbf{b}\dot{\pi}_n\dot{\pi}_{n+1}\gamma_n - \mathbf{r}'_n\gamma_n\\
&= -\mathbf{b}(\dot{\pi}_n^2\alpha_n + \dot{\pi}_{n-1}\dot{\pi}_n\beta_{n-1}) - \mathbf{A}\mathbf{r}_n + \mathbf{r}_n\alpha_n + \mathbf{r}'_{n-1}\beta_{n-1}\\
&= -\mathbf{A}\mathbf{x}_n\alpha_n - \mathbf{A}\mathbf{x}'_{n-1}\beta_{n-1} - \mathbf{A}\mathbf{r}_n.
\end{aligned}$$

If we set $\dot{\pi}_{-1} := 0$, $\dot{\pi}_0 := 1$, then (14.13) and (14.16) hold for $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{r}'_{-1} := \mathbf{x}'_{-1} := \mathbf{o}$, and the recurrence can be started with these initial values.

ALGORITHM 13. (INCONSISTENT BIORESS FORM OF BICGS)
For solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0 \in \mathbb{C}^N$, set $\mathbf{r}_0 := (\mathbf{b} - \mathbf{A}\mathbf{x}_0)/\gamma_{-1}$ with some $\gamma_{-1} \neq 0$ (for instance, such that $||\mathbf{r}_0|| = 1$), and

redefine $\mathbf{x}_0 := \mathbf{x}_0/\gamma_{-1}$. Furthermore, let $\mathbf{r}'_{-1} := \mathbf{x}'_{-1} := \mathbf{o} \in \mathbb{C}^N$, $\beta_{-1} := 0$, $\dot{\pi}_{-1} := 0$, $\dot{\pi}_0 := 1/\gamma_{-1}$, and choose $\tilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle \tilde{\mathbf{y}}_0, \mathbf{r}_0 \rangle_{\mathbf{B}} \neq 0$. Then apply Algorithm 12, additionally computing

$$\dot{\pi}_{n+1} := -(\alpha_n \dot{\pi}_n + \beta_{n-1} \dot{\pi}_{n-1})/\gamma_n, \tag{14.17a}$$

$$\mathbf{x}'_n := -(\mathbf{x}_n \alpha_n + \mathbf{x}'_{n-1} \beta_{n-1} + \mathbf{r}_n)/\gamma_n, \tag{14.17b}$$

$$\mathbf{x}_{n+1} := [\mathbf{x}_{n-1}\beta_{n-1}^2 - \mathbf{x}'_n \alpha_n \gamma_n + \mathbf{x}'_{n-1}\alpha_n \beta_{n-1} - (\mathbf{r}'_n \gamma_n - \mathbf{r}'_{n-1}\beta_{n-1})]/\gamma_n^2. \tag{14.17c}$$

If $\mathbf{r}_{n+1}\gamma_n^2 = \mathbf{o}$ and $\dot{\pi}_{n+1} \neq 0$, the algorithm terminates and $\mathbf{x}_{\text{ex}} := \mathbf{x}_{n+1}/\dot{\pi}_{n+1}^2$ is the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$; likewise, if $\mathbf{r}'_n \gamma_n = \mathbf{o}$, $\dot{\pi}_n \neq 0$, and $\dot{\pi}_{n+1} \neq 0$, the algorithm terminates and $\mathbf{x}_{\text{ex}} := \mathbf{x}'_n/(\dot{\pi}_n \dot{\pi}_{n+1})$ is the solution; if $\mathbf{r}_{n+1} \neq \mathbf{o}$ but $\delta_{n+1} = 0$, the algorithm stops due to a Lanczos breakdown. In each case we set $\nu := n + 1$.

ALGORITHM 14. (CONSISTENT BIORESS FORM OF BICGS)
Modify Algorithm 13 by choosing $\gamma_{-1} := 1$ and $\gamma_n := -\alpha_n - \beta_{n-1}$ $(n \geq 0)$, so that $\dot{\pi}_n = 1$ $(n \geq 0)$. If $\gamma_n = 0$ for some n, the algorithm stops due to a pivot breakdown, and we set $\dot{\nu} := n$. Otherwise, $\dot{\nu} := \nu$.

In both versions of BIORESS each step again requires two applications of the operator $\mathbf{A}$, namely for $\mathbf{A}\mathbf{r}_n$ and $\mathbf{A}(\mathbf{r}'_n\gamma_n - \mathbf{r}'_{n-1}\beta_{n-1})$. But note that these algorithms also produce two iterates and the corresponding residuals per step. The 'normal' BICGS iterates are $\mathbf{x}_n$, but in practice the intermediate iterates $\mathbf{x}'_n$ are often better. In fact, in Fokkema, Sleijpen and van der Vorst (1996), a *shifted* CGS *algorithm* is proposed whose residual polynomials are $(1 - \mu\zeta)p_{n-1}(\zeta)p_n(\zeta)$, where μ is a preselected value. This algorithm converges somewhat more smoothly than BICGS. With BIORESS we actually get a similar kind of iterates in addition to the usual ones. However, the three-term recurrence may spoil the accuracy more than the two-term recurrence of shifted CGS.

According to the derivation of these two algorithms the recurrence coefficients and the breakdown conditions are the same as those of the respective version of BIORES. Hence, in view of Theorem 9.2, the following result is straightforward.

Theorem 14.2 If consistent BIORES and consistent BIORESS are started with the same $\mathbf{x}_0$ and $\tilde{\mathbf{y}}_0$, then the index of first breakdown or termination $\dot{\nu}$, the recurrence coefficients α_n, β_{n-1} (and thus $\gamma_n := -\alpha_n - \beta_{n-1}$), and the inner products δ_n are the same for both algorithms. Moreover, consistent BIORESS and BIOMINS produce the same iterates $\mathbf{x}_n$ and thus also the same residuals $\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$ and the same residual polynomials p_n^2.

Likewise, if inconsistent BIORES and inconsistent BIORESS are started with the same $\mathbf{x}_0$ and $\tilde{\mathbf{y}}_0$, and if the same constants γ_n are used in both

Table 4. *Krylov space vectors and corresponding polynomials that appear in our five forms of the* BiCGS *method. Different scaling is mirrored by upper indices in the notation for the polynomials, but not in the one for the vectors. In the columns 'L' and 'P' it is indicated if an algorithm is susceptible to Lanczos breakdown or pivot breakdown, respectively*

Algorithm	Vectors	Polynomials	L	P
11 BIOMINS (CGS)	$\mathbf{r}_n$,$\mathbf{s}_n$,$\mathbf{s}'_n$,$\widehat{\mathbf{r}}_n$	p_n^2, $p_n\widehat{p}_n$, $p_{n+1}\widehat{p}_n$, $(\widehat{p}_n)^2$	√	√
13 BIORESS (cons.)	$\mathbf{r}_n$, $\mathbf{r}'_n$	p_n^2, $p_{n+1}p_n$	√	√
14 BIORESS (incons.)	$\mathbf{r}_n$, $\mathbf{r}'_n$	$(p_n^{\mathrm{INC}})^2$, $p_{n+1}^{\mathrm{INC}}p_n^{\mathrm{INC}}$	√	
BIODIRS$_1$	$\mathbf{r}_n$, $\widehat{\mathbf{r}}_n$, $\widehat{\mathbf{w}}_n$	p_n^2, $(\widehat{p}_n^{\mathrm{DIR}})^2$, $\widehat{p}_{n+1}^{\mathrm{DIR}}\widehat{p}_n^{\mathrm{DIR}}$	√	√
BIODIRS$_2$	$\mathbf{r}_n$, $\mathbf{s}_n$, $\widehat{\mathbf{r}}_n$, $\widehat{\mathbf{w}}_n$	p_n^2, $p_n\widehat{p}_n^{\mathrm{DIR}}$, $(\widehat{p}_n^{\mathrm{DIR}})^2$, $\widehat{p}_{n+1}^{\mathrm{DIR}}\widehat{p}_n^{\mathrm{DIR}}$		√

algorithms, then the index of first breakdown or termination ν, the recurrence coefficients α_n, β_{n-1}, and the inner products δ_n are the same for both algorithms. The iterates $\mathbf{x}_n$ and the vectors $\mathbf{r}_n$ are related by (4.17) and (14.13), respectively.

It follows in particular that BIORESS yields as a by-product the same tridiagonal submatrices $\mathbf{T}_n$, and thus, optionally, the same eigenvalue estimates as BIORES.

14.4. An overview of BiCGS *algorithms*

For the reader's convenience we list in Table 4 the various vectors and the corresponding polynomials that come up in the three BiCGS algorithms that we have discussed and the two, BIODIRS$_1$ and BIODIRS$_2$ from Section 7 of Gutknecht (1990), that we only alluded to. The vectors that are listed in several algorithms are, up to scaling, identical. However, neither the iterates $\mathbf{x}_n$ nor the auxiliary vectors $\mathbf{x}'_n$ that appear in BIORESS and in the two forms of BIODIRS, respectively, are contained in the list. If we wanted to judge the memory requirements, we would also have to take into account the storage of the results of matrix-vector products and the sometimes required storage of previously computed vectors. Also indicated in Table 4 are the breakdown possibilities.

Let us repeat that since the coefficients computed in the squared methods are the same as those of the respective unsquared method, one still implicitly generates the matrices $\underline{\mathbf{T}}_n$, $\underline{\mathbf{L}}_n$ and $\mathbf{U}_n$, or $\underline{\mathbf{T}}'_n$, respectively. Therefore, theoretically, the squared methods can be used for eigenvalue computations. However, it does not seem so easy to mimic ideas like selective reorthogonal-

ization or to find other ways to enhance the numerical stability. Therefore, in practice, it may be difficult to obtain reliable eigenvalue information from these methods.

15. The transpose-free QMR algorithm

In Section 14 we have seen that 'squaring' the BiCG method leads to a very effective method, BiCGS, which, however, typically exhibits a somewhat erratic convergence behaviour. An obvious remedy would be to 'square' the QMR method instead, since it converges as fast as BiCG, but more smoothly. However, it is not so obvious how this can be achieved. There are various answers to this question, but only one turns out to be convincing: Freund (1993) found a way to apply the QMR approach to bases built up from Krylov space vectors that correspond to squares and products of Lanczos and direction polynomials. The details are given below. This *transpose-free* QMR (TFQMR) *algorithm*, as he called it, is roughly equally fast but much more smoothly converging than the BiCGS method, and the cost per step is only slightly higher. Since smoother convergence often helps to reduce round-off, one may expect that there are examples where the algorithm outperforms BiOMinS in speed, but the examples in Freund's paper do not yet confirm this. However, Freund has examples where TFQMR clearly outperforms the BiCGStab algorithm of Section 16.

In Freund and Szeto (1991) an alternative strategy was followed: the *quasi-minimal residual squared* (QMRS) *algorithm* generates residuals whose residual polynomials are the squares of the QMR residual polynomials. Consequently, the convergence is fast and smooth. However, this method requires three matrix-vector products per step in contrast to two in TFQMR and BiOMinS, thus increasing the work per step by roughly 50%.

While for both these methods the residual lies in $\mathcal{K}_{2n}$ after n iterations, this is not true for the *transpose-free implementation of the* QMR *method* proposed by Chan et al. (1991). Here the idea is simply to run the BiOS algorithm for determining the Lanczos coefficients α_n, β_{n-1}, and γ_n, and then to construct the QMR iterates (or alternatively, the BiCG iterates, if a transpose-free BiCG algorithm is sought) by additionally executing the recurrences of the QMR (or the BiORes) algorithm, except for those that generate $\widetilde{\mathcal{K}}_n$. Clearly, such an approach requires considerable extra work, in particular three matrix-vector products per step instead of two. Nevertheless, the convergence speed is at best equal to the one of the BiCG method, not the one of the BiCGS method. Moreover, the Lanczos coefficients found by the BiOS algorithm, although in theory identical to those of inconsistent BiORes, turn out to be more contaminated by round-off. Therefore, convergence is in practice often worse than for QMR and BiCG, respectively.

Algorithms using this approach cannot compete with other transpose-free methods.

Let us now turn to the preferred approach, the TFQMR algorithm. Multiplying (14.1) by p_n and p_{n-1}, we get the two relations

$$\begin{aligned} p_n p_{n+1} &:= p_n^2 - \omega_n \zeta p_n \widehat{p}_n, \\ p_{n+1}^2 &:= p_n p_{n+1} - \omega_n \zeta p_{n+1} \widehat{p}_n. \end{aligned}$$

Recalling the definitions (see (14.7) and (14.11))

$$\begin{aligned} \mathbf{r}_n &:= p_n^2(\mathbf{A})\mathbf{r}_0, & \mathbf{r}'_n &:= p_{n+1}(\mathbf{A})p_n(\mathbf{A})\mathbf{r}_0, \\ \mathbf{s}_n &:= p_n(\mathbf{A})\widehat{p}_n(\mathbf{A})\mathbf{r}_0, & \mathbf{s}'_n &:= p_{n+1}(\mathbf{A})\widehat{p}_n(\mathbf{A})\mathbf{r}_0, \end{aligned}$$

we can translate this into the recurrences

$$\mathbf{r}'_n := \mathbf{r}_n - \mathbf{A}\mathbf{s}_n\omega_n, \tag{15.1}$$

$$\mathbf{r}_{n+1} := \mathbf{r}'_n - \mathbf{A}\mathbf{s}'_n\omega_n, \tag{15.2}$$

which, together with (14.8b), (14.8c), and

$$\mathbf{A}\widehat{\mathbf{r}}_{n+1} := \mathbf{A}\mathbf{s}_{n+1} - \mathbf{A}\mathbf{s}'_n\psi_n + \mathbf{A}\widehat{\mathbf{r}}_n\psi_n^2,$$

allow us to build up the Krylov space. Note that (14.8h) generates $\mathbf{A}\widehat{\mathbf{r}}_n$ recursively, so that there is no need to determine $\widehat{\mathbf{r}}_n$ itself. Only the two matrix-vector products $\mathbf{A}\mathbf{s}_n$ and $\mathbf{A}\mathbf{s}'_n$ are required per step, and they boost the dimension of the space by two.

Defining the matrices

$$\begin{aligned} \mathbf{R}_{2n} &:= [\ \mathbf{r}_0 \ \ \mathbf{r}'_0 \ \ \mathbf{r}_1 \ \ \dots \ \ \mathbf{r}_{n-1} \ \ \mathbf{r}'_{n-1}\ ], \\ \mathbf{S}_{2n} &:= [\ \mathbf{s}_0 \ \ \mathbf{s}'_0 \ \ \mathbf{s}_1 \ \ \dots \ \ \mathbf{s}_{n-1} \ \ \mathbf{s}'_{n-1}\ ], \end{aligned}$$

each containing a basis for $\mathcal{K}_{2n}$, and extending this definition to odd indices by dropping the last component, we can write (15.1)–(15.2) as

$$\mathbf{A}\mathbf{S}_m = \mathbf{R}_{m+1}\underline{\mathbf{L}}_m^{[1]}\mathbf{D}_{\omega|\omega;m}^{-1}, \tag{15.3}$$

where

$$\underline{\mathbf{L}}_m^{[1]} := \begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & -1 & \end{bmatrix}$$

is $(m+1) \times m$ lower bidiagonal and $\mathbf{D}_{\omega|\omega;m}$ is the $m \times m$ diagonal matrix

$$\mathbf{D}_{\omega|\omega;m} := \operatorname{diag}(\omega_0, \omega_0, \omega_1, \omega_1, \omega_2, \dots).$$

Note that m can either be even, $m = 2n$, or odd, $m = 2n+1$.

Once (15.3) is found, the usual QMR approach applies. We represent the mth iterate as

$$\mathbf{x}_m^{\mathrm{TFQMR}} = \mathbf{x}_0 + \mathbf{S}_m \mathbf{k}_m,$$

so that $\mathbf{r}_m^{\mathrm{TFQMR}} = \mathbf{r}_0 - \mathbf{A}\mathbf{S}_m\mathbf{k}_m$ holds for the residual. Inserting (15.3) yields

$$\mathbf{r}_m^{\mathrm{TFQMR}} = \mathbf{R}_{m+1}\left(\underline{\mathbf{e}}_1 - \underline{\mathbf{L}}_m^{[1]}\mathbf{D}_{\omega|\omega;m}^{-1}\mathbf{k}_m\right) = \mathbf{R}_{m+1}\mathbf{D}_{\mathbf{r}|\mathbf{r}';m+1}^{-1}\mathbf{q}_m,$$

with the diagonal matrix

$$\mathbf{D}_{\mathbf{r}|\mathbf{r}';m+1} := \mathrm{diag}(||\mathbf{r}_0||, ||\mathbf{r}_0'||, ||\mathbf{r}_1||, ||\mathbf{r}_1'||, \ldots)$$

used to normalize the columns of $\mathbf{R}_{m+1}$ and the quasi-residual

$$\mathbf{q}_m := \underline{\mathbf{e}}_1\rho_0 - \underline{\mathbf{L}}_m^{\mathrm{TFQMR}}\mathbf{k}_m \quad \text{with } \underline{\mathbf{L}}_m^{\mathrm{TFQMR}} := \mathbf{D}_{\mathbf{r}|\mathbf{r}';m+1}\underline{\mathbf{L}}_m^{[1]}\mathbf{D}_{\omega|\omega;m}^{-1}.$$

This $(m+1)\times m$ least squares problem is solved by a QR decomposition based on Givens rotations as before. Once again, the quasi-residual norms are found for free, and the iterates can be updated as in Section 5.

Writing $\mathbf{t}$ instead of $\mathbf{A}\hat{\mathbf{r}}$ we can formulate the TFQMR algorithm as follows.

ALGORITHM 15. (TFQMR ALGORITHM)
For solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0 \in \mathbb{C}^N$, set $\mathbf{s}_0 := \mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{t}_0 := \mathbf{A}\mathbf{r}_0$, and choose $\tilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\delta_0 := \langle\tilde{\mathbf{y}}_0, \mathbf{r}_0\rangle_{\mathbf{B}} \neq 0$ and $\delta_0' := \langle\tilde{\mathbf{y}}_0, \mathbf{A}\hat{\mathbf{r}}_0\rangle_{\mathbf{B}} \neq 0$. Then compute for $n = 0, 1, \ldots$

$$\begin{aligned}
\omega_n &:= \delta_n/\delta_n', && (15.4a)\\
\mathbf{s}_n' &:= \mathbf{s}_n - \mathbf{t}_n\omega_n, && (15.4b)\\
\mathbf{r}_n' &:= \mathbf{r}_n - \mathbf{A}\mathbf{s}_n\omega_n, && (15.4c)\\
\mathbf{r}_{n+1} &:= \mathbf{r}_n' - \mathbf{A}\mathbf{s}_n'\omega_n, && (15.4d)\\
\delta_{n+1} &:= \langle\tilde{\mathbf{y}}_0, \mathbf{r}_{n+1}\rangle_{\mathbf{B}}\,, && (15.4e)\\
\psi_n &:= -\delta_{n+1}/\delta_n, && (15.4f)\\
\mathbf{s}_{n+1} &:= \mathbf{r}_{n+1} - \mathbf{s}_n'\psi_n, && (15.4g)\\
\mathbf{t}_{n+1} &:= \mathbf{A}\mathbf{s}_{n+1} - \mathbf{A}\mathbf{s}_n'\psi_n + \mathbf{t}_n\psi_n^2, && (15.4h)\\
\delta_{n+1}' &:= \langle\tilde{\mathbf{y}}_0, \mathbf{t}_{n+1}\rangle_{\mathbf{B}}\,. && (15.4i)
\end{aligned}$$

Within this loop, for $m := 2n+1$ and $2n+2$, additionally

(1) update the QR factorization $\underline{\mathbf{L}}_m^{\mathrm{TFQMR}} = \mathbf{Q}_m\underline{\mathbf{R}}_n^{\mathrm{TFQMR}}$, analogous to (5.6)–(5.7) (with $\underline{\mathbf{T}}_n$ replaced by $\underline{\mathbf{L}}_m^{\mathrm{TFQMR}}$)
(2) according to (5.8), compute the coefficient vector $\mathbf{h}_m$ by appending the component $c_m\tilde{\eta}_m$ to $\mathbf{h}_{m-1}$, and compute the new last component $\tilde{\eta}_{m+1} := -\overline{s_m}\,\tilde{\eta}_m$ of $\underline{\mathbf{h}}_m$
(3) compute $\mathbf{z}_{m-1}$ according to the two-term recurrence implied by $\mathbf{S}_m = \mathbf{R}_m^{\mathrm{TFQMR}}\mathbf{Z}_m$

(4) compute $\mathbf{x}'_n$ and $\mathbf{x}_{n+1}$ according to

$$\mathbf{x}'_n := \mathbf{x}_n + \mathbf{z}_{2n} c_{2n+1} \widetilde{\eta}_{2n+1}, \quad \mathbf{x}_{n+1} := \mathbf{x}'_n + \mathbf{z}_{2n+1} c_{2n+2} \widetilde{\eta}_{2n+2},$$

respectively,

(5) if the norm of the quasi-residual, $\widetilde{\eta}_m$, is below a certain bound, check the norm of the true residual; stop if it is also small enough.

Like BIORESS, the TFQMR algorithm has the benefit that it delivers two iterates and corresponding residual norms per step. We will encounter the same property again in the following section on Lanczos-type product methods.

By comparison with Algorithm 11, note that by inserting the assignment

$$\mathbf{x}^{\mathrm{G}}_{n+1} := \mathbf{x}^{\mathrm{G}}_n + (\mathbf{s}_n + \mathbf{s}'_n)\, \omega_n$$

after (15.4d), we could additionally produce the BICGS iterates $\mathbf{x}^{\mathrm{G}}_n$ at almost no cost.

16. Lanczos-type product methods

As we mentioned, Sonneveld's BICGS method has the disadvantage that convergence is often interrupted by a sudden large increase of the residual norm, followed by an equally fast decrease to the previous order of magnitude. Although such spikes normally do not prevent convergence, they may reduce the speed of convergence and, in particular, the ultimate accuracy of the solution. Actually, it is sometimes rather the maximum norm of the iterates (not the residuals) that counts; see Section 18. Most users of iterative methods prefer a smoother, if not monotone, residual norm plot.

By improving an unpublished idea of Sonneveld, van der Vorst (1992) was the first to find a modification of the BICGS method with smoother convergence. In retrospect, his BICGSTAB algorithm can be understood as the application of the BICGS approach to a coupled two-term version of the one-sided Lanczos process of Section 6 instead of BIOMIN. In other words, we make use of the freedom to choose the left polynomials t_n different from the Lanczos polynomials. The residual polynomials of the resulting method are no longer the squares p_n^2 but the products $p_n t_n$, where t_n is an arbitrary polynomial of exact degree n satisfying $t_n(0) = 1$. We therefore call the class of such methods *Lanczos-type product methods* (*LTPMs*).

In BICGSTAB the polynomials t_n are determined indirectly by local one-dimensional residual minimization, and in BICGSTAB2 (Gutknecht 1993*c*) we have extended this approach to local two-dimensional minimization, which is more appropriate in view of the typically complex spectrum of non-Hermitian matrices. In the same paper we presented the formulae for using an arbitrary set of polynomials t_n satisfying a three-term recurrence, such as, for instance, suitably shifted and scaled Chebyshev polynomials as

they are used in the Chebyshev method for solving linear systems. Other authors have also contributed to the class of LTPMs; see in particular Brezinski and Redivo Zaglia (1995), Fokkema et al. (1996), Zhang (1997). There are several algorithms that compete for being the most efficient one for a broad variety of examples: among these are BiCGStab2 (Gutknecht 1993*c*), BiCGStab(ℓ) (Sleijpen and Fokkema 1993), and BiCG×MR2 (Gutknecht 1994*c*), which are treated as examples below.

We concentrate here on consistent LTPMs based on the coupled two-term recurrences for the Lanczos polynomials and a three-term recurrence for the second set of polynomials, but we will indicate modifications needed for other classes. Consistent and inconsistent LTPMs based on three-term recurrences for both sets are treated in Gutknecht and Ressel (1996), where the application of look-ahead to these methods was also achieved. Eijkhout (1994) also derived a three-term version of BiCGStab, but his way of finding the Lanczos recurrence coefficients is unnecessarily complicated. Brezinski and Redivo Zaglia (1995) suggested a different way of doing look-ahead; see Section 19 for comments.

Algorithms that combine the BiCGS method or an LTPM with smoothing processes attain convergence speeds equal to the best LTPMs and an even smoother residual norm plot. In particular, the BiCGS method and LTPMs can be combined with quasi-residual minimization. In the former case one finds Freund's TFQMR algorithm (Freund 1993) of Section 15. A QMR-smoothed BiCGStab algorithm was introduced in Chan, Gallopoulos, Simoncini, Szeto and Tong (1994), while QMR minimization for general LTPMs is described by Ressel and Gutknecht (1996). A very effective alternative is the minimum residual (MR) smoothing process, which can be applied to any Krylov space solver and will be described in Section 17.

Most of the methods discussed in this section only make sense if $\mathbf{B}$ is Hermitian positive definite. Since $\mathbf{B}$ is also required to commute with $\mathbf{A}$, there are hardly any interesting examples other than $\mathbf{B} = \mathbf{I}$. We therefore assume here that this holds, that is, we drop $\mathbf{B}$.

16.1. LTPMs based on coupled two-term recurrences

When choosing the polynomials t_n of an LTPM we mainly aim at two properties: (i) fast and smooth convergence of the resulting solver; (ii) low memory requirements, which, in particular, means short recurrences. In BiCGStab the recurrence is two-term, but this is a strong restriction for a basis of polynomials of increasing degree. In contrast, three-term recurrences are satisfied by a broad class of such bases including all sets of classical orthogonal polynomials. We assume the recurrence to be of the form

$$t_{l+1}(\zeta) := (\xi_l + \eta_l \zeta)\, t_l(\zeta) + (1 - \xi_l)\, t_{l-1}(\zeta), \tag{16.1}$$

with $t_{-1}(\zeta) :\equiv 0$, $t_0(\zeta) :\equiv 1$ and $\xi_0 = 1$. Note that this form conserves the consistency condition: $t_l(0) = 1$ (for all n). The formulae for the resulting LTPM were given in Gutknecht (1993*c*), but we choose here a different exposition that is analogous to the one in Gutknecht and Ressel (1996) and Ressel and Gutknecht (1996) for LTPMs based on the Lanczos three-term recurrences.

We define two doubly indexed sequences of *product vectors*:

$$\begin{aligned} \mathbf{w}_n^l &:= t_l(\mathbf{A})\mathbf{y}_n = t_l(\mathbf{A})p_n(\mathbf{A})\mathbf{y}_0, \\ \widehat{\mathbf{w}}_n^l &:= t_l(\mathbf{A})\mathbf{v}_n = t_l(\mathbf{A})\widehat{p}_n(\mathbf{A})\mathbf{y}_0. \end{aligned}$$

Here, $\mathbf{y}_n$ and $\mathbf{v}_n$ are the right Lanczos and direction vectors of BIOMIN; they will not appear in the final algorithm. Additionally, we introduce a doubly indexed sequence of *product iterates* $\mathbf{x}_n^l$ with the properties

$$\mathbf{b} - \mathbf{A}\mathbf{x}_n^l = \mathbf{w}_n^l \quad \text{and} \quad \mathbf{x}_n^l \in \mathbf{x}_0 + \mathcal{K}_{n+l}. \tag{16.2}$$

The diagonal sequences $\{\mathbf{x}_n^n\}$ and $\{\mathbf{w}_n^n\}$ contain the iterates and residuals we really aim at. Some other product iterates and product vectors will appear in the recurrences and can occasionally also be considered as useful approximations to the solution vector $\mathbf{x}_{ex}$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$ and the corresponding residual, respectively.

Let us arrange the product vectors into a $\mathbf{w}$-*table* and a $\widehat{\mathbf{w}}$-*table*. These two tables are very helpful for explaining the underlying mechanism of LTPMs. To fix matters, we let the n-axis point downwards and the l-axis point to the right. Then the iteration will essentially proceed from the upper left corner (with the initial vectors $\mathbf{w}_0^0 := \widehat{\mathbf{w}}_0^0 := \mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$) to the lower right. The entries that are actually needed in competitive algorithms lie on or close to a diagonal of the tables, that is, $|n - l|$ is small for them. For moving down the tables, we apply the consistent coupled two-term Lanczos recurrences (14.1)–(14.2). Multiplying them by $t_l(\zeta)$ and translating into Krylov space notation we obtain

$$\mathbf{w}_{n+1}^l := \mathbf{w}_n^l - \mathbf{A}\widehat{\mathbf{w}}_n^l\omega_n, \tag{16.3a}$$

$$\widehat{\mathbf{w}}_{n+1}^l := \mathbf{w}_{n+1}^l - \widehat{\mathbf{w}}_n^l\psi_n. \tag{16.3b}$$

On the other hand, for moving to the right we multiply (16.1) by p_n and $\widehat{p}_n$, respectively. Translating into Krylov space notation we then get

$$\mathbf{w}_n^{l+1} := \mathbf{A}\mathbf{w}_n^l\eta_l + \mathbf{w}_n^l\xi_l + \mathbf{w}_n^{l-1}(1 - \xi_l), \tag{16.3c}$$

$$\widehat{\mathbf{w}}_n^{l+1} := \mathbf{A}\widehat{\mathbf{w}}_n^l\eta_l + \widehat{\mathbf{w}}_n^l\xi_l + \widehat{\mathbf{w}}_n^{l-1}(1 - \xi_l). \tag{16.3d}$$

Of course, since the Lanczos two-term recurrences are coupled, elements of both tables appear in (16.3a) and (16.3b), while (16.3c) and (16.3d) each affect only one table. Constructing recursively the diagonals of the two tables requires us to apply these formulae cyclically for a certain sequence

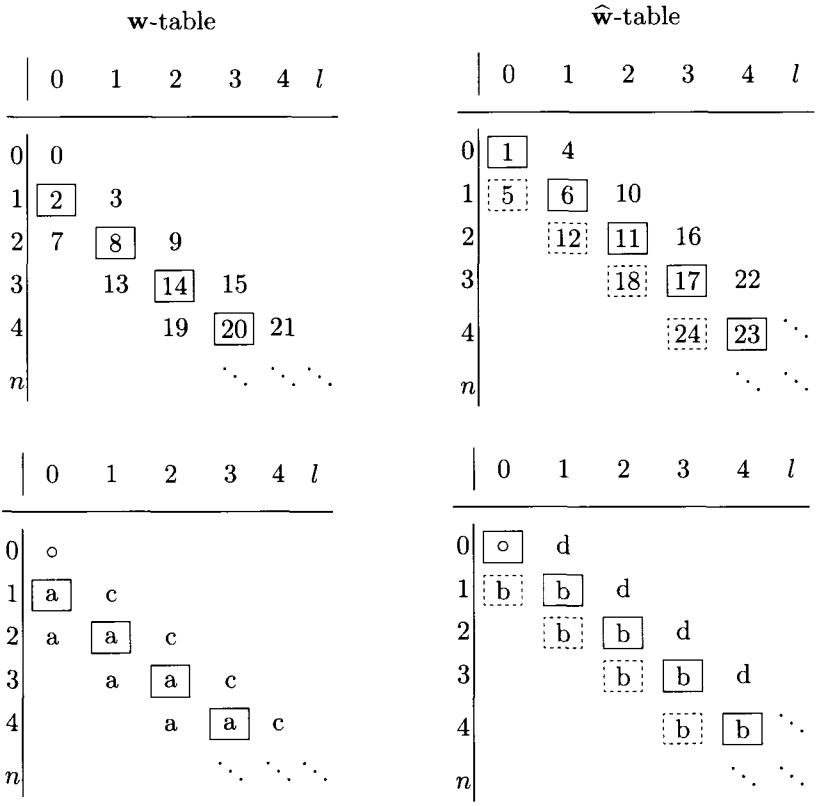

Fig. 1. The $\mathbf{w}$-table and the $\widehat{\mathbf{w}}$-table of an LTPM based on coupled two-term Lanczos recurrences and a three-term recurrence for the polynomials t_l. The numbers in the upper tables specify the order in which the elements $\mathbf{w}_n^l$ and $\widehat{\mathbf{w}}_n^l$ of the two tables are computed. The letters in the lower tables indicate which formula among (16.3a)–(16.3d) to use for computing the corresponding entry. An entry in a solid box indicates that its product with $\mathbf{A}$ also has to be computed. For the entries in dashed boxes the product with $\mathbf{A}$ can be computed indirectly, without executing a matrix-vector product

of (l, n)-pairs. The order in which the formulae are applied is indicated by the numbers in the upper half of Figure 1, while the letters in the lower half specify the formula applied.

Inserting (16.2) into formulae (16.3a) and (16.3c), subtracting $\mathbf{b}$ on each side of them, and multiplying by the inverse of $\mathbf{A}$ we readily get corresponding recurrences for the product iterates:

$$\begin{aligned} \mathbf{x}_{n+1}^l &:= \mathbf{x}_n^l + \widehat{\mathbf{w}}_n^l \omega_n, \\ \mathbf{x}_n^{l+1} &:= -\mathbf{w}_n^l \eta_l + \mathbf{x}_n^l \xi_l + \mathbf{x}_n^{l-1}(1 - \xi_l). \end{aligned}$$

Of course, we could arrange them in an $\mathbf{x}$-table and display it along with the $\widehat{\mathbf{w}}$-table, very much like Figure 1. Note that $\mathbf{x}_0^0 := \mathbf{x}_0$.

It remains to specify formulae for determining the coefficients ω_n and ψ_n. Recall that in the Lanczos process we have $\widetilde{\mathcal{K}}_n \perp \mathbf{y}_n$ and $\widetilde{\mathcal{K}}_n \perp \mathbf{A}\mathbf{v}_n$. Because $\{t_l(\mathbf{A})\widetilde{\mathbf{y}}_0\}_{l=0}^{n-1}$ is a basis of $\widetilde{\mathcal{K}}_n$ (as long as n does not exceed the grade of $\widetilde{\mathbf{y}}_0$ with respect to $\mathbf{A}^\star$), we conclude that

$$\langle \widetilde{\mathbf{y}}_0, \mathbf{w}_n^l \rangle = 0, \quad \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_n^l \rangle = 0, \quad \text{if } l < n. \tag{16.5}$$

In particular, if we set $l := n-1$ and then replace n by $n+1$, we have

$$\langle \widetilde{\mathbf{y}}_0, \mathbf{w}_{n+1}^n \rangle = 0, \quad \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_{n+1}^n \rangle = 0.$$

Taking this orthogonality into account in the recursions (16.3a)–(16.3b) for $l = n$ and defining

$$\widetilde{\delta}_n := \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_n^n \rangle, \quad \widetilde{\delta}'_n := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_n^n \rangle$$

leads us to

$$\omega_n := \frac{\langle \widetilde{\mathbf{y}}_0, \mathbf{w}_n^n \rangle}{\langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_n^n \rangle} = \frac{\widetilde{\delta}_n}{\widetilde{\delta}'_n},$$

and

$$\psi_n := \frac{\langle \widetilde{\mathbf{y}}_0, \mathbf{A}\mathbf{w}_{n+1}^n \rangle}{\langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_n^n \rangle} = \frac{\widetilde{\delta}_{n+1}}{\widetilde{\delta}'_n \eta_n}.$$

For the last equation we have expressed $\mathbf{A}\mathbf{w}_{n+1}^n$ according to (16.3c) and taken (16.5) into account to see that $\langle \widetilde{\mathbf{y}}_0, \mathbf{A}\mathbf{w}_{n+1}^n \rangle = \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_{n+1}^{n+1} \rangle / \eta_n$.

Summarizing, we find the following *generic consistent* $(3, 2\times 2)$*-type LTPM*. Its type $(3, 2 \times 2)$ indicates that we apply a three-term recurrence for the polynomials t_l and two coupled two-term recurrences for the Lanczos and the direction polynomials. In Ressel and Gutknecht (1996) the analogous $(3,3)$-type LTPM is called BIO×THREE.

ALGORITHM 16. (GENERIC CONSISTENT $(3, 2 \times 2)$-TYPE LTPM)
For solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0^0 \in \mathbb{C}^N$ and set $\mathbf{w}_0^0 := \widehat{\mathbf{w}}_0^0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0^0$. Choose $\widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\widetilde{\delta}_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_0^0 \rangle \neq 0$ and $\widetilde{\delta}'_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_0^0 \rangle \neq 0$. Then compute for $n = 0, 1, \ldots$

$$\omega_n := \widetilde{\delta}_n / \widetilde{\delta}'_n, \tag{16.6a}$$

$$\mathbf{w}_{n+1}^{n-1} := \mathbf{w}_n^{n-1} - \mathbf{A}\widehat{\mathbf{w}}_n^{n-1}\omega_n, \quad \text{if } n > 0, \tag{16.6b}$$

$$\mathbf{x}_{n+1}^{n-1} := \mathbf{x}_n^{n-1} + \widehat{\mathbf{w}}_n^{n-1}\omega_n, \quad \text{if } n > 0, \tag{16.6c}$$

$$\mathbf{w}_{n+1}^{n} := \mathbf{w}_n^{n} - \mathbf{A}\widehat{\mathbf{w}}_n^{n}\omega_n, \tag{16.6d}$$

$$\mathbf{x}_{n+1}^{n} := \mathbf{x}_n^{n} + \widehat{\mathbf{w}}_n^{n}\omega_n, \tag{16.6e}$$

$$\mathbf{w}_{n+1}^{n+1} := \mathbf{A}\mathbf{w}_{n+1}^{n}\eta_n + \mathbf{w}_{n+1}^{n}\xi_n + \mathbf{w}_{n+1}^{n-1}(1 - \xi_n), \tag{16.6f}$$

$$\mathbf{x}_{n+1}^{n+1} := -\mathbf{w}_{n+1}^{n}\eta_n + \mathbf{x}_{n+1}^{n}\xi_n + \mathbf{x}_{n+1}^{n-1}(1-\xi_n), \tag{16.6g}$$

$$\widetilde{\delta}_{n+1} := \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_{n+1}^{n+1} \rangle, \tag{16.6h}$$

$$\widehat{\mathbf{w}}_{n}^{n+1} := \mathbf{A}\widehat{\mathbf{w}}_{n}^{n}\eta_n + \widehat{\mathbf{w}}_{n}^{n}\xi_n + \widehat{\mathbf{w}}_{n}^{n-1}(1-\xi_n), \tag{16.6i}$$

$$\psi_n := \widetilde{\delta}_{n+1}/(\widetilde{\delta}'_n\eta_n), \tag{16.6j}$$

$$\widehat{\mathbf{w}}_{n+1}^{n} := \mathbf{w}_{n+1}^{n} - \widehat{\mathbf{w}}_{n}^{n}\psi_n, \tag{16.6k}$$

$$\mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n} := \mathbf{A}\mathbf{w}_{n+1}^{n} - \mathbf{A}\widehat{\mathbf{w}}_{n}^{n}\psi_n, \tag{16.6l}$$

$$\widehat{\mathbf{w}}_{n+1}^{n+1} := \mathbf{w}_{n+1}^{n+1} - \widehat{\mathbf{w}}_{n}^{n+1}\psi_n, \tag{16.6m}$$

$$\widetilde{\delta}'_{n+1} := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n+1} \rangle, \tag{16.6n}$$

If $\mathbf{w}_{n+1}^{l} = \mathbf{o}$ for $l = n-1$, n, or $n+1$, then the algorithm terminates and $\mathbf{x}_{n+1}^{l}$ is the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$. However, if none of these residuals vanishes, but $\widetilde{\delta}_{n+1} = 0$ or $\widetilde{\delta}'_{n+1} = 0$, the algorithm breaks down.

Note that $\mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n}$ is obtained without using a matrix-vector product, hence only two of them are needed per step: $\mathbf{A}\mathbf{w}_{n+1}^{n}$ and $\mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n+1}$. In the above form, the algorithm requires considerable storage (although of course we assume that entries on the same (co-)diagonal of any of our four tables are stored at the same memory location). However, some of the vectors can be spared, namely those that are used only once later and do not depend on a quantity that is overwritten before the vector is used. An extra benefit of the given algorithm is that we not only obtain one approximate solution per step but three, since each of the vectors $\mathbf{x}_{n}^{l}$ can be considered as one and its residual is available. However, checking the length of such a residual requires an inner product. On the other hand, in more sophisticated versions of the algorithm some of these inner products may get computed anyway in order to determine if it is worth recomputing the vector by applying reorthogonalization; see Section 18.

If the polynomials t_n satisfy a two-term recurrence instead of the three-term recurrence, that is, if $\xi_l = 1$ (for all l), then we do not need the quantities on the lowest codiagonal of each table. This is the most important advantage of BiCGStab over other LTPMs.

16.2. The BiCGStab *algorithm*

In van der Vorst's BiCGStab method (van der Vorst 1992), the polynomials t_l are built up in factored form,

$$t_l(\zeta) = (1-\chi_1\zeta)(1-\chi_2\zeta)\cdots(1-\chi_l\zeta),$$

and, hence, they satisfy a two-term recurrence:

$$t_{l+1}(\zeta) = (1-\chi_{l+1}\zeta)\, t_l(\zeta).$$

This means that in the three-term recurrence of (16.1) we have

$$\xi_l = 1, \quad \eta_l = -\chi_{l+1}.$$

Note that such a two-term recurrence necessarily means that the zeros remain fixed, except that an additional one is added at each step.

In the nth step, the reciprocal χ_{n+1} of the $(n+1)$th zero is chosen such that $\mathbf{w}_{n+1}^{n+1}$ has minimum length:

$$||\mathbf{w}_{n+1}^{n+1}|| := \min_{\chi} ||\mathbf{w}_{n+1}^{n} - \mathbf{A}\mathbf{w}_{n+1}^{n}\chi||. \tag{16.7}$$

This one-dimensional minimization problem is solved by making $\mathbf{w}_{n+1}^{n+1}$ orthogonal to $\mathbf{A}\mathbf{w}_{n+1}^{n}$, which means that

$$\chi_{n+1} := \frac{\langle \mathbf{A}\mathbf{w}_{n+1}^{n}, \mathbf{w}_{n+1}^{n}\rangle}{||\mathbf{A}\mathbf{w}_{n+1}^{n}||^2}. \tag{16.8}$$

This remains correct in the case of complex data; see Gutknecht (1993*c*).

Due to the two-term recurrence, there is no need to compute $\mathbf{w}_{n+1}^{n-1}$, $\widehat{\mathbf{w}}_{n+1}^{n}$, and $\mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n}$, so that the corresponding lines in Algorithm 16 can be dropped. Moreover, in order to minimize the memory requirements we insert the formulae for $\mathbf{x}_{n+1}^{n}$ and $\widehat{\mathbf{w}}_{n}^{n+1}$ at the points where these vectors are actually used.

ALGORITHM 17. (BICGSTAB ALGORITHM)
For solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0^0 \in \mathbb{C}^N$ and set $\mathbf{w}_0^0 := \widehat{\mathbf{w}}_0^0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0^0$. Choose $\widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\widetilde{\delta}_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_0^0\rangle \neq 0$ and $\widetilde{\delta}_0' := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_0^0\rangle \neq 0$. Then compute for $n = 0, 1, \ldots$

$$\omega_n := \widetilde{\delta}_n/\widetilde{\delta}_n', \tag{16.9a}$$

$$\mathbf{w}_{n+1}^{n} := \mathbf{w}_n^n - \mathbf{A}\widehat{\mathbf{w}}_n^n\omega_n, \tag{16.9b}$$

$$\chi_{n+1} := \langle \mathbf{A}\mathbf{w}_{n+1}^{n}, \mathbf{w}_{n+1}^{n}\rangle/||\mathbf{A}\mathbf{w}_{n+1}^{n}||^2, \tag{16.9c}$$

$$\mathbf{w}_{n+1}^{n+1} := \mathbf{w}_{n+1}^{n} - \mathbf{A}\mathbf{w}_{n+1}^{n}\chi_{n+1}, \tag{16.9d}$$

$$\mathbf{x}_{n+1}^{n+1} := \mathbf{x}_n^n + \widehat{\mathbf{w}}_n^n\omega_n + \mathbf{w}_{n+1}^{n}\chi_{n+1}, \tag{16.9e}$$

$$\widetilde{\delta}_{n+1} := \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_{n+1}^{n+1}\rangle, \tag{16.9f}$$

$$\psi_n := -\widetilde{\delta}_{n+1}/(\widetilde{\delta}_n'\chi_{n+1}), \tag{16.9g}$$

$$\widehat{\mathbf{w}}_{n+1}^{n+1} := \mathbf{w}_{n+1}^{n+1} - (\widehat{\mathbf{w}}_n^n - \mathbf{A}\widehat{\mathbf{w}}_n^n\chi_{n+1})\,\psi_n, \tag{16.9h}$$

$$\widetilde{\delta}_{n+1}' := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n+1}\rangle, \tag{16.9i}$$

If $\mathbf{w}_{n+1}^{n+1} = \mathbf{o}$ or $\mathbf{w}_{n+1}^{n} = \mathbf{o}$, the algorithm terminates and, respectively, $\mathbf{x}_{n+1}^{n+1}$ or $\mathbf{x}_{n+1}^{n}$ (defined by $\mathbf{x}_{n+1}^{n} := \mathbf{x}_n^n + \widehat{\mathbf{w}}_n^n\omega_n$) is the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$. Otherwise, if $\widetilde{\delta}_{n+1} = 0$ or $\widetilde{\delta}_{n+1}' = 0$, the algorithm breaks down.

Note that this algorithm, like TFQMR, provides two residuals and corresponding iterates per step, although, as suggested in our formulation, we need to compute only one iterate per step and can determine the other only once its residual satisfies the convergence criterion.

The breakdown conditions $\widetilde{\delta}_{n+1} = 0$ and $\widetilde{\delta}'_{n+1} = 0$ seem to indicate a Lanczos or pivot breakdown, respectively. However, BICGSTAB is also subject to a somewhat hidden danger of breakdown. In fact, due to the different leading coefficients of p_n and t_n, the inner products δ_{n+1}, δ'_{n+1} of the Lanczos process and $\widetilde{\delta}_{n+1}$, $\widetilde{\delta}'_{n+1}$ of BICGSTAB are related by

$$\widetilde{\delta}_{n+1} = \delta_{n+1} \frac{\chi_1 \chi_2 \cdots \chi_{n+1}}{\omega_0 \omega_1 \cdots \omega_n}, \quad \widetilde{\delta}'_{n+1} = \delta'_{n+1} \frac{\chi_1 \chi_2 \cdots \chi_{n+1}}{\omega_0 \omega_1 \cdots \omega_n}.$$

Consequently, not only the Lanczos breakdown ($\delta_{n+1} = 0$) and the pivot breakdown ($\delta'_{n+1} = 0$) take their toll, but also the *minimization breakdown* $\chi_{n+1} = 0$. But the latter also surfaces as a vanishing of $\widetilde{\delta}_{n+1}$ (and $\widetilde{\delta}'_{n+1}$). It is no surprise that a vanishing χ_{n+1} causes a disaster, since the Krylov space is then no longer expanded by (16.9d).

Unfortunately, there are applications where χ_n tends to be small, namely if $\mathbf{A}$ has eigenvalues with small real part, but non-small imaginary part. An example are matrices resulting from the discretization of convection-dominated convection–diffusion equations.

BICGSTAB has the additional disadvantage that, for real matrices and right-hand sides (and real initial vectors), the zeros $1/\chi_k$ of t_l are necessarily always real, and therefore they cannot efficiently help to damp error components associated with eigenvalues of $\mathbf{A}$ with large imaginary part.

16.3. The BICGSTAB2 *algorithm*

Both above-mentioned disadvantages of BICGSTAB can be removed by replacing the local one-dimensional minimization by a two-dimensional one in every other step. In between, in the odd steps, we may still do a one-dimensional minimization, but it will have no effect on what follows, at least not in exact arithmetic, as long as we advance in the Krylov space. This is the idea behind BICGSTAB2 (Gutknecht 1993*c*). Here, if required to be consistent, t_l satisfies recurrences of the form

$$\begin{aligned} t_{l+1}(\zeta) &:= (1 - \chi_{l+1}\zeta)\, t_l(\zeta), \quad \text{if } l \text{ is even}, \\ t_{l+1}(\zeta) &:= (\xi_l + \eta_l\zeta)\, t_l(\zeta) + (1 - \xi_l)\, t_{l-1}(\zeta), \quad \text{if } l \text{ is odd}. \end{aligned}$$

The pair (ξ_n, η_n) is again chosen to minimize $\mathbf{w}_{n+1}^{n+1}$, and in view of the relevant recurrence (16.3c) this means that the pair is the minimizer of

$$||\mathbf{w}_{n+1}^{n+1}|| = \min_{\xi,\eta} ||\mathbf{A}\mathbf{w}_{n+1}^{n}\eta + \mathbf{w}_{n+1}^{n}\xi + \mathbf{w}_{n+1}^{n-1}(1 - \xi)||. \tag{16.11}$$

This is a standard least squares problem for two unknowns. It could be solved via the 2×2 system of normal equations, but since this system can be ill-conditioned, it may be preferable to use a QR decomposition. In theory, the system cannot be rank-deficient except when the Krylov space is exhausted: rank-deficiency means here that $\mathbf{A}\mathbf{w}_{n+1}^n \in \mathcal{K}_{2n+3}$ and $\mathbf{w}_{n+1}^n - \mathbf{w}_{n+1}^{n-1} \in \mathcal{K}_{2n+2}$ are linearly dependent.

In Gutknecht (1993*c*) we chose χ_n such that in the odd steps the one-dimensional minimization problem (16.7) is solved. At the cost of one additional inner product[16], this choice has the advantage of also producing at odd steps a new, normally better iterate that may satisfy the convergence criterion. On the other hand, the result of the next, even step does not depend on χ_n as long as $\chi_n \neq 0$, except for a possible effect on round-off.

Sleijpen and Fokkema (1993) pointed out that round-off may indeed be a problem if $|\chi_{n+1}|$ is small, which is likely to happen for some n if $\mathbf{A}$ has eigenvalues close to the imaginary axis. In contrast to BICGSTAB, which will then have the same round-off problem, we can get around it here. In this case, for example, we could just redefine χ_{n+1} as 1 (or some other suitably chosen value). We could even go further, forget the one-dimensional minimization problem completely, and just let $\chi_{n+1} := 1$ for all n. This has the additional advantage that one inner product can be saved. The choice of 1 as value of χ_{n+1} (and, thus, of an additional zero at 1 of t_n if n is odd) can be justified by the fact that well preconditioned matrices often have a cluster of eigenvalues around 1. However, numerical experiments indicate that $\chi_{n+1} := 1$ is not always the best choice.

In fact, choosing χ_{n+1} requires us to compromise between several objectives (see Section 18 for more details):

(i) finding small residuals, or at least, avoiding large intermediate iterates $\mathbf{x}_n^n$ and $\mathbf{x}_n^{n-1}$

(ii) avoiding round-off errors in the computation of the inner products $\widetilde{\delta}_{n+1}$ and $\widetilde{\delta}'_{n+1}$

(iii) avoiding stagnation of the Krylov space generation.

An analysis (Sleijpen and van der Vorst 1995*a*) shows that the last two objectives go hand in hand and suggests choosing χ_{n+1} as minimizer of

$$\frac{||\mathbf{w}_{n+1}^{n+1}||}{|\chi|} = \frac{||\mathbf{w}_{n+1}^n - \mathbf{A}\mathbf{w}_{n+1}^n \chi||}{|\chi|}$$

instead of (16.7), which optimizes the first objective. This requires us to

[16] The other one is needed anyway for solving (16.11).

make $\mathbf{A}\mathbf{w}_{n+1}^n - \mathbf{w}_{n+1}^n/\chi_{n+1}$ orthogonal to $\mathbf{w}_{n+1}^n$, that is, to choose

$$\chi_{n+1} := \frac{||\mathbf{w}_{n+1}^n||^2}{\langle \mathbf{w}_{n+1}^n, \mathbf{A}\mathbf{w}_{n+1}^n \rangle}. \tag{16.12}$$

Compromising suitably between (16.8) and (16.12), taking the value of

$$\widehat{\delta}_{n+1} := \frac{\langle \mathbf{w}_{n+1}^n, \mathbf{A}\mathbf{w}_{n+1}^n \rangle}{||\mathbf{w}_{n+1}^n||\ ||\mathbf{A}\mathbf{w}_{n+1}^n||}$$

into account, ultimately leads to our recommendation (Gutknecht and Ressel 1997) to let in BICGSTAB2

$$\chi_{n+1} := \begin{cases} \dfrac{||\mathbf{w}_{n+1}^n||}{||\mathbf{A}\mathbf{w}_{n+1}^n||} & \text{if } \widehat{\delta}_{n+1} \le \frac{1}{2}\sqrt{2}, \\ \dfrac{\langle \mathbf{A}\mathbf{w}_{n+1}^n, \mathbf{w}_{n+1}^n \rangle}{||\mathbf{A}\mathbf{w}_{n+1}^n||^2} & \text{if } \widehat{\delta}_{n+1} > \frac{1}{2}\sqrt{2}. \end{cases} \tag{16.13}$$

This is a variation of a proposal of Sleijpen and van der Vorst (1995*a*) for choosing χ_{n+1} in BICGSTAB such that the effect of rounding errors in case of stagnation is minimized (Sleijpen and van der Vorst 1995*a*, Section 3.4).

In their BICGSTAB(2) algorithm, which is mathematically equivalent to BICGSTAB2, Sleijpen and Fokkema (1993) gave up the consistency condition of t_l for odd l and simply based the iteration on

$$t_{l+1}(\zeta) := \zeta\, t_l(\zeta), \quad \text{if } l \text{ is even}, \tag{16.14a}$$

$$\begin{aligned} t_{l+1}(\zeta) &:= (1 - \chi_l\zeta - \chi_{l+1}\zeta^2)t_{l-1}(\zeta) \\ &= t_{l-1}(\zeta) - \chi_l t_l(\zeta) - \chi_{l+1}\zeta t_l(\zeta) \quad \text{if } l \text{ is odd}, \end{aligned} \tag{16.14b}$$

where χ_l and χ_{l+1} are now the parameters of the two-dimensional minimization problem for $\mathbf{w}_{n+1}^{n+1}$.

This saves a few operations, but has the disadvantage that the odd steps do not produce a residual and a corresponding iterate. Therefore, only every four matrix-vector products does one have a chance to realize that the algorithm has converged. On the other hand, once the problems caused by small values of $|\chi_{n+1}|$ and ill-conditioned least squares problems have been eliminated in BICGSTAB2, we can expect that it is numerically as stable as BICGSTAB(2), in particular since (16.14b) seems to be more susceptible to a growing gap between recursively computed and true residuals, a difference barely noticed in practice, however; see Section 18 for more details. Alternative versions of BICGSTAB(2) introduced in Sleijpen, van der Vorst and Fokkema (1994) also avoid these two drawbacks.

BICGSTAB2 is easy to generalize by allowing any sequence of one-, two-, or even higher-dimensional minimizations. Let us define $\mathcal{S}_\ell$ as the set of indices $n+1$ where we perform an ℓ-dimensional minimization, and let $\mathcal{S}_0$

denote the set of indices where no minimization is performed. Since $n+1$ starts at 1 in our algorithms, we then have, for example, $\mathcal{S}_1 = \{1,3,5,\ldots\}$ and $\mathcal{S}_2 = \{2,4,6,\ldots\}$ for standard BiCGStab2, but $\mathcal{S}_0 = \{1,3,5,\ldots\}$ and $\mathcal{S}_2 = \{2,4,6,\ldots\}$ for BiCGStab(2). In the variant of BiCGStab2 obtained by observing (16.13), which in contrast to BiCGStab(2) is still consistent at each n, it is determined on the fly whether an odd index belongs to $\mathcal{S}_0$ or $\mathcal{S}_1$. But even if it does not, the residual will not normally grow so much, so that the loss of ultimate accuracy is kept within limits.

With this notation, BiCGStab2 and its consistent variants can be formulated as follows.

ALGORITHM 18. (GENERALIZED BiCGStab2 ALGORITHM)
For solving $\mathbf{Ax} = \mathbf{b}$ choose an initial approximation $\mathbf{x}_0^0 \in \mathbb{C}^N$ and set $\mathbf{w}_0^0 := \widehat{\mathbf{w}}_0^0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0^0$. Choose $\widetilde{\mathbf{y}}_0 \in \mathbb{C}^N$ such that $\widetilde{\delta}_0 := \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_0^0 \rangle \neq 0$ and $\widetilde{\delta}_0' := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_0^0 \rangle \neq 0$. Then, for $n = 0, 1, \ldots$, compute

$$\omega_n := \widetilde{\delta}_n / \widetilde{\delta}_n', \tag{16.15a}$$

$$\mathbf{w}_{n+1}^{n-1} := \mathbf{w}_n^{n-1} - \mathbf{A}\widehat{\mathbf{w}}_n^{n-1}\omega_n, \quad \text{if } n+1 \in \mathcal{S}_2, \tag{16.15b}$$

$$\mathbf{x}_{n+1}^{n-1} := \mathbf{x}_n^{n-1} + \widehat{\mathbf{w}}_n^{n-1}\omega_n, \quad \text{if } n+1 \in \mathcal{S}_2, \tag{16.15c}$$

$$\mathbf{w}_{n+1}^{n} := \mathbf{w}_n^{n} - \mathbf{A}\widehat{\mathbf{w}}_n^{n}\omega_n, \tag{16.15d}$$

$$\mathbf{x}_{n+1}^{n} := \mathbf{x}_n^{n} + \widehat{\mathbf{w}}_n^{n}\omega_n. \tag{16.15e}$$

If $n+1 \notin \mathcal{S}_2$, define χ_{n+1}, for example, by (16.13) and let

$$\mathbf{w}_{n+1}^{n+1} := \mathbf{w}_{n+1}^{n} - \mathbf{A}\mathbf{w}_{n+1}^{n}\chi_{n+1}, \tag{16.15f}$$

$$\mathbf{x}_{n+1}^{n+1} := \mathbf{x}_{n+1}^{n} + \mathbf{w}_{n+1}^{n}\chi_{n+1}, \tag{16.15g}$$

$$\widetilde{\delta}_{n+1} := \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_{n+1}^{n+1} \rangle, \tag{16.15h}$$

$$\widehat{\mathbf{w}}_n^{n+1} := \widehat{\mathbf{w}}_n^{n-1} - \mathbf{A}\widehat{\mathbf{w}}_n^{n}\chi_{n+1}, \tag{16.15i}$$

$$\psi_n := -\widetilde{\delta}_{n+1} / (\widetilde{\delta}_n' \chi_{n+1}), \tag{16.15j}$$

$$\widehat{\mathbf{w}}_{n+1}^{n} := \mathbf{w}_{n+1}^{n} - \widehat{\mathbf{w}}_n^{n}\psi_n, \tag{16.15k}$$

$$\mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n} := \mathbf{A}\mathbf{w}_{n+1}^{n} - \mathbf{A}\widehat{\mathbf{w}}_n^{n}\psi_n, \quad \text{if } n+2 \in \mathcal{S}_2, \tag{16.15l}$$

$$\widehat{\mathbf{w}}_{n+1}^{n+1} := \mathbf{w}_{n+1}^{n+1} - \widehat{\mathbf{w}}_n^{n+1}\psi_n, \tag{16.15m}$$

else (that is, if $n+1 \in \mathcal{S}_2$) determine ξ_n and η_n as solutions of (16.11) and proceed with

$$\mathbf{w}_{n+1}^{n+1} := \mathbf{A}\mathbf{w}_{n+1}^{n}\eta_n + \mathbf{w}_{n+1}^{n}\xi_n + \mathbf{w}_{n+1}^{n-1}(1-\xi_n), \tag{16.15n}$$

$$\mathbf{x}_{n+1}^{n+1} := -\mathbf{w}_{n+1}^{n}\eta_n + \mathbf{x}_{n+1}^{n}\xi_n + \mathbf{x}_{n+1}^{n-1}(1-\xi_n), \tag{16.15o}$$

$$\widehat{\mathbf{w}}_{n}^{n+1} := \mathbf{A}\widehat{\mathbf{w}}_n^{n}\eta_n + \widehat{\mathbf{w}}_n^{n}\xi_n + \widehat{\mathbf{w}}_n^{n-1}(1-\xi_n), \tag{16.15p}$$

$$\widetilde{\delta}_{n+1} := \langle \widetilde{\mathbf{y}}_0, \mathbf{w}_{n+1}^{n+1} \rangle, \tag{16.15q}$$

$$\psi_n := \widetilde{\delta}_{n+1}/(\widetilde{\delta}'_n \eta_n), \tag{16.15r}$$

$$\widehat{\mathbf{w}}_{n+1}^{n} := \mathbf{w}_{n+1}^{n} - \widehat{\mathbf{w}}_{n}^{n} \psi_n, \tag{16.15s}$$

$$\mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n} := \mathbf{A}\mathbf{w}_{n+1}^{n} - \mathbf{A}\widehat{\mathbf{w}}_{n}^{n} \psi_n, \tag{16.15t}$$

$$\widehat{\mathbf{w}}_{n+1}^{n+1} := \mathbf{w}_{n+1}^{n+1} - \widehat{\mathbf{w}}_{n}^{n+1} \psi_n. \tag{16.15u}$$

Finally, set

$$\widetilde{\delta}'_{n+1} := \langle \widetilde{\mathbf{y}}_0, \mathbf{A}\widehat{\mathbf{w}}_{n+1}^{n+1} \rangle. \tag{16.15v}$$

If $\mathbf{w}_{n+1}^{l} = \mathbf{o}$ for $l = n-1$, n, or $n+1$, then the algorithm terminates and $\mathbf{x}_{n+1}^{l}$ is the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$. However, if none of these residuals vanishes, but $\widetilde{\delta}_{n+1} = 0$ or $\widetilde{\delta}'_{n+1} = 0$, the algorithm breaks down.

Note that as in BICGSTAB storing $\mathbf{x}_{n+1}^{n}$ and $\mathbf{x}_{n+1}^{n-1}$ could be avoided, and that $\widehat{\mathbf{w}}_{n+1}^{n}$ could be stored over $\mathbf{w}_{n+1}^{n}$.

16.4. The BICG×MR2 *and* BICG×CHEBY *methods*

In the framework of Algorithm 18 we may in particular let $\mathcal{S}_2 = \{2, 3, 4, \ldots\}$, which means that we do a two-dimensional minimization at every step except the first one (where $n+1 = 1$). This yields a method we call BICG×MR2 or, if we want to indicate that we mean the version based on coupled two-term recurrences, the BIOC×MR2 algorithm. It was proposed in Gutknecht (1994*c*) and also independently by Cao (1997*b*) and by Zhang (1997); a look-ahead variant based on the three-term look-ahead Lanczos recurrences is given in Gutknecht and Ressel (1996), while the combination with the QMR approach is explored in Ressel and Gutknecht (1996). This method has the advantage of avoiding the sometimes doubtful one-dimensional minimization step and yet producing an iterate and its residual at every step. Experiments show that it typically converges slightly faster than BICGSTAB2, which often converges markedly faster than BICGSTAB. The latter can be considered as a BICG×MR1 method.

Compared to BICGSTAB2, the character of the polynomials t_n changes drastically in BICG×MR2: while in the former method just two new zeros are added in every other step, but all the other zeros remain fixed, here all the zeros get modified in each step, as is the case with orthogonal polynomials, for which the zeros interlace.

A further alternative is to look at product methods where the second set of polynomials, $\{t_l\}$, consists of the residual polynomials of some other Krylov space solver. For example, one might choose them as suitably shifted and scaled Chebyshev polynomials, as mentioned as a possibility in van der Vorst (1992) and Gutknecht (1993*c*). This requires the construction of an

ellipse containing the eigenvalues of $\mathbf{A}$. A technique for this task was devised by Manteuffel (1977). The recurrences of the resulting BICG×CHEBY *algorithm* are the same as those for BICG×MR2, but the coefficients η_n and ξ_n are known in advance; they are determined by the foci of the ellipse.

16.5. The BICGSTAB(ℓ) *algorithm*

Sleijpen and Fokkema (1993) went one step further in the generalization of BICGSTAB and BICGSTAB2: with BICGSTAB(ℓ) (where ℓ is a positive integer) they introduced a method that performs an ℓ-dimensional minimization every ℓ steps. In other words, the polynomials t_n are built up by appending polynomial factors of degree ℓ. The original paper suggested generating the Krylov space in between by simply multiplying $\mathbf{w}_{k\ell}^{k\ell}$ successively by $\mathbf{A}$ as in (16.14a). We have mentioned already that this may cause a departure of the recursively computed from the true residual. Therefore, two other realizations of this method were proposed and compared in Sleijpen et al. (1994): a 'stabilized matrix' version, which is, for $\ell > 4$, considerably more costly, and an 'orthogonal matrix' version, which sometimes seems to converge more slowly; see Tables 1 and 5 in Sleijpen et al. (1994). Further tests ultimately led to the enhanced implementation of BICGSTAB(ℓ) of Fokkema (1996*a*, 1996*b*).

There are examples where $\ell = 4$ is markedly superior to $\ell = 2$, and there are even cases where $\ell = 8$ works well, while neither $\ell < 8$ nor any other method tried converged. But in most examples the convergence rate seems to be about the same. There are two reasons to expect better convergence: first, the residual reduction due to ℓ-dimensional minimization is clearly stronger than for ℓ times of one-dimensional minimization; second, it can be seen that the Lanczos process is less affected by round-off if one works with larger ℓ.

16.6. Further LTPMs

As is clear from the definition of LTPMs, there exist infinitely many such methods, even when we allow at most three terms in the recursion for t_n, and quite a few of them may seem to make sense for one reason or another. But our interest is of course restricted to those that are competitive.

Moreover, as mentioned, LTPMs come in various versions depending on the recursions used. We have chosen here to describe what we call the $(3, 2\times 2)$-type LTPM, which applies a three-term recurrence for the polynomials t_l and two coupled two-term recurrences for the Lanczos and the direction polynomials. We could equally well use the three-term recurrence for the Lanczos polynomials, as in Gutknecht and Ressel (1996) and Ressel and Gutknecht (1996), thus getting $(3, 3)$-type LTPMs. Then there is no $\hat{\mathbf{w}}$-table, but the $\mathbf{w}$-table has in the generic case bandwidth 4. On the other

hand, one can also turn to coupled two-term recurrences for both polynomial sequences, that is, $(2\times 2, 2\times 2)$-type LTPMs, as suggested by Fokkema et al. (1996) as well as Zhang (1997). This has the advantage that the standard BIOMINS version of the BICGS method fits into the pattern, but, on the other hand, methods like BICGSTAB2 and BICG×MR2 are less easy to reformulate. To display these versions in a way that is analogous to our Figure 1, we would have to introduce four tables for four different types of product vectors, and in each table the bandwidth would be 2.

Among the recently proposed LTPMs is the already mentioned *shifted* CGS *algorithm* of Fokkema et al. (1996) whose residual polynomials are $(1-\mu\zeta)p_{n-1}(\zeta)p_n(\zeta)$, where μ is a fixed chosen value, for instance the inverse of an estimate for the largest eigenvalue for $\mathbf{A}$. The authors' examples indicate that its residual norm histories have typically less dramatic peaks than BICGS. The same is true for the CGS2 *method* introduced in the same paper: there, both polynomials are Lanczos polynomials of the same degree, but they correspond to two different left initial vectors. A number of further LTPMs were suggested by Brezinski and Redivo Zaglia (1995), but examples for demonstrating their usefulness are missing.

17. Smoothing processes

The erratic convergence behaviour of the basic Lanczos-type solvers, the BICG and BICGS methods, often gives rise to criticism. Plots of the residual norm are the most often used tool when algorithms are compared and, therefore, a method sells well if it converges quickly and smoothly. Thus, it is not surprising that there is an interest in smoothing processes that modify the BICG or BICGS iterates so that the residual norm plot becomes smoother. However, we should say that smoothing is also dubious. In fact, what really counts in practice is that a method should find the solution (up to a certain error) as quickly as possible and, since the error cannot be checked, the residual is monitored instead. The smoothness of the convergence does not matter from that point of view. Nevertheless, smoothing processes are of some interest since they can speed up the convergence slightly. On the other hand, they hide some useful information: a peak in the residual norm plot indicates a temporary stagnation of convergence, while in the smoothed residual plot we cannot distinguish temporary from permanent stagnation.

A question that has been resolved concerns the relationship between the convergence behaviour of CG and CR, as well as FOM and GMRES. This relationship approximately carries over to BICG and QMR, and makes us understand why peaks in the residual norm plot of BICG are matched by plateaux in the one of QMR. This result is closely related to our previous dis-

cussion of the relationship between (Petrov–)Galerkin and (quasi-)minimal residual methods, in Section 5.

17.1. Trivial minimal residual smoothing

If a monotone residual norm plot is all we aim at, then there is a trivial recipe. We let[17]

$$\widetilde{\mathbf{x}}_n := \begin{cases} \widetilde{\mathbf{x}}_{n-1}, \\ \mathbf{x}_n, \end{cases} \qquad \widetilde{\mathbf{r}}_n := \begin{cases} \widetilde{\mathbf{r}}_{n-1}, & \text{if } \|\widetilde{\mathbf{r}}_{n-1}\| < \|\mathbf{r}_n\|, \\ \mathbf{r}_n, & \text{if } \|\widetilde{\mathbf{r}}_{n-1}\| \geq \|\mathbf{r}_n\|. \end{cases}$$

This clearly implies that the residuals $\widetilde{\mathbf{r}}_n$ of the 'smoothed' iterates satisfy

$$\|\widetilde{\mathbf{r}}_n\| = \min\{\|\widetilde{\mathbf{r}}_{n-1}\|, \|\mathbf{r}_n\|\} = \min\{\|\mathbf{r}_0\|, \|\mathbf{r}_1\|, \ldots, \|\mathbf{r}_n\|\}.$$

We call this *trivial minimal residual* (TMR) *smoothing*. We do not consider this as a serious proposal, but mention it, because it is sometimes applied when numerical results are presented. It reflects the position mentioned above, that all that really matters is to fulfil a prescribed bound for the residual norm as quickly as possible. This is a legitimate reason for applying TMR smoothing when publishing results, but the code of conduct requires that authors declare it.

Note that TMR neither increases nor reduces round-off.

17.2. Minimal residual smoothing

Given any pair of sequences of iterates and residuals, for example those produced by the BiCG method, Schönauer (1987) proposed to replace them by the smoothed sequences

$$\widetilde{\mathbf{x}}_n := \widetilde{\mathbf{x}}_{n-1}(1-\theta_n) + \mathbf{x}_n\theta_n, \quad \widetilde{\mathbf{r}}_n := \widetilde{\mathbf{r}}_{n-1}(1-\theta_n) + \mathbf{r}_n\theta_n, \tag{17.1}$$

where θ_n is chosen to make the residual as small as possible, which requires that

$$\widetilde{\mathbf{r}}_{n-1} - \mathbf{r}_n \perp \widetilde{\mathbf{r}}_n, \tag{17.2}$$

or

$$\theta_n := \frac{\langle \widetilde{\mathbf{r}}_{n-1} - \mathbf{r}_n, \widetilde{\mathbf{r}}_{n-1}\rangle}{\|\widetilde{\mathbf{r}}_{n-1} - \mathbf{r}_n\|^2}. \tag{17.3}$$

From the relations (17.1) and (17.2) we conclude by Pythagoras' theorem that

$$\|\widetilde{\mathbf{r}}_n\|^2 = \|\widetilde{\mathbf{r}}_{n-1}\|^2 - \|\widetilde{\mathbf{r}}_{n-1} - \mathbf{r}_n\|^2|\theta_n|^2. \tag{17.4}$$

The idea was further developed and investigated by Weiss (1990, 1994),

[17] Before, we used tildes for the left Lanczos vectors and the left direction vectors, but we did not define $\widetilde{\mathbf{x}}_n$ and $\widetilde{\mathbf{r}}_n$. These vectors are now the smoothed iterates and residuals.

and then taken up by Gutknecht (1993*a*) as well as Zhou and Walker (1994); see also Walker (1995). Following Zhou and Walker, we call it *minimal residual* (MR) *smoothing*. Note that we perform a *local*, one-dimensional minimization, as we did in BICGSTAB. Of course, we could generalize this approach to a local MR(ℓ) smoothing, that is, an ℓ-dimensional minimization additionally involving $\widetilde{\mathbf{r}}_{n-2}, \ldots, \widetilde{\mathbf{r}}_{n-\ell}$. However, numerical tests show that this is hardly worth the extra work (Zhou and Walker 1994).

Obviously, MR smoothing is some kind of a recursive weighted mean process, but, in general, the weights need not be positive. From the definition it is clear, however, that the resulting residual norm plot is monotonically decreasing. Therefore, this is a very effective smoothing process. Note that the given sequence is piped through the process without generating any feedback.

The main theoretical result of Weiss' thesis is that applying MR smoothing to the FOM iterates yields the GCR (or, GMRES) iterates: the orthogonal residuals of the former become conjugate and minimal residuals of the latter method. For a short proof see Gutknecht (1993*a*, p. 49). *A fortiori*, applying the MR smoothing to the CG iterates yields the CR iterates. Hence, in these two cases the transformation must be identical to the relations (5.22) and (5.23), and we conclude that

$$\theta_n = c_n^2 \ \ (\geq 0), \quad 1 - \theta_n = |s_n|^2 \ \ (\geq 0). \tag{17.5}$$

Actually, in these cases of orthogonal residual methods, where $\mathbf{r}_n \perp \mathcal{K}_n$, (17.3) and (17.4) simplify since we have $\mathbf{r}_n \perp \widetilde{\mathbf{r}}_{n-1} \in \mathcal{K}_n$ and thus, again by Pythagoras, $||\widetilde{\mathbf{r}}_{n-1} - \mathbf{r}_n||^2 = ||\widetilde{\mathbf{r}}_{n-1}||^2 + ||\mathbf{r}_n||^2$, so that

$$\theta_n = \frac{||\widetilde{\mathbf{r}}_{n-1}||^2}{||\widetilde{\mathbf{r}}_{n-1}||^2 + ||\mathbf{r}_n||^2} \quad \in (0, 1].$$

Inserting this into (17.4) and taking the reciprocal yields

$$\frac{1}{||\widetilde{\mathbf{r}}_n||^2} = \frac{||\widetilde{\mathbf{r}}_{n-1}||^2 + ||\mathbf{r}_n||^2}{||\widetilde{\mathbf{r}}_{n-1}||^2 ||\mathbf{r}_n||^2} = \frac{1}{||\widetilde{\mathbf{r}}_{n-1}||^2} + \frac{1}{||\mathbf{r}_n||^2} = \sum_{k=0}^{n} \frac{1}{||\mathbf{r}_k||^2}. \tag{17.6}$$

Here, for the last equality, we applied induction in order to represent the norm of the smoothed residual in terms of the original residual norms. Conversely, solving for $||\mathbf{r}_n||^2$ leads to

$$||\mathbf{r}_n||^2 = \frac{||\widetilde{\mathbf{r}}_n||^2}{1 - ||\widetilde{\mathbf{r}}_n||^2 / ||\widetilde{\mathbf{r}}_{n-1}||^2} \quad (n \geq 1). \tag{17.7}$$

We will comment on these formulae later, but want to remind the reader at this point that they only hold if the given residuals $\mathbf{r}_n$ are mutually orthogonal.

In Gutknecht (1993*a*) we pointed out that there is also an algorithm to do the inverse of MR smoothing. One motivation for using it can be that in

Galerkin methods the errors $\mathbf{x}_n - \mathbf{x}_{\text{ex}}$ have a different spectral decomposition and are often smaller than in methods that minimize the residual.

17.3. Quasi-minimal residual smoothing

When we apply MR smoothing to the BiCG iterates, the resulting smoothed iterates differ from those of the QMR method. But in Section 5 we have seen that nevertheless the BiCG iterates $\mathbf{x}_n := \mathbf{x}_n^{\text{G}}$ and their residuals $\mathbf{r}_n := \mathbf{r}_n^{\text{G}}$ are related to the QMR iterates $\widetilde{\mathbf{x}}_n := \mathbf{x}_n^{\text{MR}}$ by the relations (5.22) and (5.23), which are of the form (17.1) with θ_n satisfying (17.5). Now θ_n is no longer determined by one-dimensional minimization in the residual space, which leads to the choice (17.3), but given by (17.5), where, as we have seen in (5.25), c_n^2 and $|s_n|^2$ can be expressed in terms of the norms of the BiCG residual, $||\mathbf{r}_n^G||$, and the quasi-residual, $||\mathbf{q}_n|| = |\widetilde{\eta}_{n+1}|$:

$$\theta_n := c_n^2 = \frac{||\mathbf{q}_n||^2}{||\mathbf{r}_n||^2}. \tag{17.8}$$

Moreover, $||\mathbf{q}_n||^2$ satisfies the recursion (5.24):

$$\frac{1}{||\mathbf{q}_n||^2} = \frac{1}{||\mathbf{q}_{n-1}||^2} + \frac{1}{||\mathbf{r}_n||^2}. \tag{17.9}$$

Of course, in practice, a substitution $\upsilon_n := ||\mathbf{q}_n||^2$ will be made, because we do not compute $\mathbf{q}_n$.

Zhou and Walker (1994) suggest applying the smoothing process (17.1) with this choice of weights to any kind of Krylov space solver. They call this QMR *smoothing*.

Comparing (17.9) with (17.6) we see that $||\mathbf{q}_n||$ here takes the place of $||\widetilde{\mathbf{r}}_n||$. In particular, as in (17.6) and (17.7), we now have

$$\frac{1}{||\mathbf{q}_n||^2} = \sum_{k=0}^{n} \frac{1}{||\mathbf{r}_k||^2}, \quad ||\mathbf{r}_n||^2 = \frac{||\mathbf{q}_n||^2}{1 - ||\mathbf{q}_n||^2/||\mathbf{q}_{n-1}||^2} \quad (n \geq 1). \tag{17.10}$$

There is an alternative to formulae (17.8) and (17.9) that leads to the same weights: according to (5.21) we have

$$||\mathbf{r}_n|| = \frac{1}{c_n} ||\mathbf{q}_n|| = \frac{|s_1 \cdots s_n|}{c_n} ||\mathbf{r}_0|| = \frac{|s_n|}{c_n} ||\mathbf{q}_{n-1}||.$$

In view of the interpretation of s_n and c_n as sine and cosine, this suggests defining

$$\tau_n := \arctan \frac{||\mathbf{r}_n||}{||\mathbf{q}_{n-1}||} \in \left[0, \frac{\pi}{2}\right).$$

Then we let

$$c_n := \cos \tau_n = \frac{||\mathbf{q}_{n-1}||}{\sqrt{||\mathbf{r}_n||^2 + ||\mathbf{q}_{n-1}||^2}}, \quad s_n := \sin \tau_n = \frac{||\mathbf{r}_n||}{\sqrt{||\mathbf{r}_n||^2 + ||\mathbf{q}_{n-1}||^2}}.$$

It is easy to verify that this construction is consistent with the former one of (17.8) and (17.9), and that, as before, $||\mathbf{r}_n||\, c_n = ||\mathbf{q}_n|| = ||\mathbf{q}_{n-1}||\, s_n$. Note that here $s_n \in [0, 1)$, while in the QMR method s_n can take complex values.

17.4. An alternative smoothing algorithm using direction vectors

Zhou and Walker (1994) noticed that when MR or QMR smoothing is applied to BIOMIN or BIOMINS, better numerical results can be obtained with a reformulated algorithm that updates the smoothed iterates using direction vectors. Assume that our iterates and residuals are generated by a formula of the form

$$\mathbf{x}_{n+1} := \mathbf{x}_n + \mathbf{v}_n \omega_n, \quad \mathbf{r}_{n+1} := \mathbf{r}_n - \mathbf{A}\mathbf{v}_n \omega_n. \tag{17.11}$$

Here, $\mathbf{v}_n$ and ω_n can, but need not, be the quantities from BIOMIN. In order to rewrite the smoothing formulae (17.1) we introduce the difference $\mathbf{u}_n := \mathbf{x}_n - \widetilde{\mathbf{x}}_{n-1}$, so that

$$\widetilde{\mathbf{x}}_n := \widetilde{\mathbf{x}}_{n-1} + \mathbf{u}_n \theta_n, \quad \widetilde{\mathbf{r}}_n := \widetilde{\mathbf{r}}_{n-1} - \mathbf{A}\mathbf{u}_n \theta_n. \tag{17.12}$$

Next we need update formulae for $\mathbf{u}_n$ and $\mathbf{A}\mathbf{u}_n$; we do not want to spend an extra matrix-vector product on the latter and consider it as a single vector. Substituting the above update formulae into $\mathbf{u}_{n+1} = \mathbf{x}_{n+1} - \widetilde{\mathbf{x}}_n$ yields

$$\mathbf{u}_{n+1} := \mathbf{v}_n \omega_n + \mathbf{u}_n (1 - \theta_n), \quad \mathbf{A}\mathbf{u}_{n+1} := \mathbf{A}\mathbf{v}_n \omega_n + \mathbf{A}\mathbf{u}_n (1 - \theta_n). \tag{17.13}$$

Consequently, if for a Krylov space solver (17.11) holds, then the formulae (17.12) and (17.13) are an alternative form of the smoothing process based on (17.1). For MR smoothing there is also a simplified formula for the weight θ_n:

$$\theta_n := \frac{\langle \mathbf{A}\mathbf{u}_n, \widetilde{\mathbf{r}}_{n-1} \rangle}{||\mathbf{A}\mathbf{u}_n||^2}. \tag{17.14}$$

17.5. The peak–plateau connection

Numerical experiments with the QMR methods and with MR smoothing show that peaks in the residual norm plot of BICG or BICGS are always matched by plateaux in the plot for the smoothed method, a *plateau* being a region where the residual norm stagnates or decreases only slowly. This phenomenon was studied in Brown (1991) and Cullum (1995), but it was finally Cullum and Greenbaum (1996) who came up with a simple explanation based on formulae we derived above, notably (17.7) and the analogue one in (17.10); see also Walker (1995). They also proved by example that the peaks in a FOM residual plot need not come from a near-singular $\mathbf{H}_n$.

Indeed, according to (17.7), if $||\mathbf{r}_n|| \gg ||\widetilde{\mathbf{r}}_n||$, then the denominator must be small, that is, $||\widetilde{\mathbf{r}}_n|| \approx ||\widetilde{\mathbf{r}}_{n-1}||$, and *vice versa*. On the other hand, if the norm of the smoothed residual decreases quickly, then the denominator will

be close to 1, and thus the residual of the original method is nearly as small as the smoothed one. Consequently, in a very vague sense, either both the original method and the smoothed one do well, or both do badly. Equation (17.6) also makes clear that the smoothed residual cannot be much smaller than the original one; in fact, it follows that

$$||\widetilde{\mathbf{r}}_n|| \geq \frac{1}{\sqrt{n+1}} \min_{0 \leq k \leq n} ||\mathbf{r}_k||,$$

and equality only holds if $||\mathbf{r}_k|| = ||\mathbf{r}_n||$ for $k \leq n$. However, we need to recall that (17.6) and (17.7) assume orthogonal original residuals. Therefore, the application of MR smoothing to BiCG residuals is not covered.

Regarding the QMR method (or the application of QMR smoothing to the BiCG residuals), we can draw the same conclusions on the plateau behaviour of the quasi-residuals $\mathbf{q}_n$, since these appear in (17.10). Moreover, we know that the QMR residual is at most $\sqrt{n+1}$ times larger than the quasi-residual, and in practice the factor is often closer to 1. Since the peaks in the BiCG residual norm plot can be several orders of magnitude high, this factor is rather unimportant in this discussion.

18. Accuracy considerations

Unfortunately, in finite precision arithmetic, inherent round-off problems jeopardize the use of the BiO and BiOC processes and related algorithms. There are at least four effects that can cause trouble:

(i) the loss of (bi)orthogonality
(ii) the low relative accuracy of certain inner products, notably δ_n and δ'_n
(iii) inaccurate Krylov space extension
(iv) the deviation between the recursively computed and the true residual.

There has been considerable work on analysing some of these effects, and also on finding ways to reduce them or compensate for them. Regarding the nonsymmetric Lanczos process, this is an area where results are very recent and investigations are still going on: one wants to know where and why Lanczos-type algorithms lose accuracy, and how one can avoid that at a reasonable price. The production of quality software relies heavily on such findings. But this also means that a good implementation of these algorithms is much more complicated than one would expect from the basic descriptions we have given here. We can only give a very brief and superficial overview of this area, however.

Considerable effort has also been invested into the backwards error analysis of the symmetric Lanczos and the standard CG algorithms (Greenbaum 1989, Greenbaum 1994*b*, Greenbaum and Strakos 1992). More recently, this effort has been extended to BiOMin (Tong and Ye 1995). But we cannot discuss this work here.

18.1. Loss of biorthogonality

Recall that, in the nth step of the BIO algorithm, the vectors $\mathbf{y}_{n+1}$ and $\widetilde{\mathbf{y}}_{n+1}$ are determined so that $\widetilde{\mathbf{y}}_n \perp_{\mathbf{B}} \mathbf{y}_{n+1}$, $\widetilde{\mathbf{y}}_{n-1} \perp_{\mathbf{B}} \mathbf{y}_{n+1}$ and $\widetilde{\mathbf{y}}_{n+1} \perp_{\mathbf{B}} \mathbf{y}_n$, $\widetilde{\mathbf{y}}_{n+1} \perp_{\mathbf{B}} \mathbf{y}_{n-1}$. Nevertheless, the biorthogonality $\widetilde{\mathbf{y}}_m \perp_{\mathbf{B}} \mathbf{y}_{n+1}$, $\widetilde{\mathbf{y}}_{n+1} \perp_{\mathbf{B}} \mathbf{y}_m$ theoretically holds for all $m \leq n$. But due to round-off one encounters a loss of this inherited biorthogonality when $n - m$ becomes large. This is no surprise: orthogonal projection necessarily reduces the size of a vector and thus its relative accuracy. If recursively generated projections of vectors are the relevant data used in the process of building up dual bases, and if one counts on inherited orthogonality, it is not surprising that working with finite precision may have a strong effect.

Lanczos (1952, pp. 39–40) was aware of this loss and suggested, on one hand, full reorthogonalization as a possible yet expensive remedy and, on the other hand, for the iterative solution of linear systems, the modification of the right-hand side by damping the components that correspond to the large eigenvalues. This second remedy, which he called 'purification' of the right-hand side, is similar in spirit to what we now call polynomial preconditioning, but he applied it only before the Lanczos process and not at every step. By suitable preconditioning the loss of (bi)orthogonality is reduced and, at the same time, becomes less relevant since the residual becomes sufficiently small long before n is comparable to N in size.

On the other hand, for the eigenvalue problem, where only few preconditioning techniques apply and the accuracy of the tridiagonal matrix is crucial, the loss of orthogonality is a serious problem even in the symmetric case, for which this numerical phenomenon was analysed by Paige (1971, 1976, 1980). His theory is also discussed in Cullum and Willoughby (1985) and Parlett (1980). It allows us to recognize when a critical step occurs that will induce loss of orthogonality. This loss is coupled with the occurrence of extra, so-called *spurious* copies of eigenvalues and eigenvectors. This is nicely demonstrated by the numerical experiments displayed in Parlett (1994). For Hermitian $\mathbf{A}$, Parlett and his coworkers (Parlett 1980, Parlett and Nour-Omid 1989, Parlett and Reid 1981, Parlett and Scott 1979, Parlett et al. 1985, Simon 1984*a*, Simon 1984*b*) as well as Cullum and Willoughby (Cullum and Willoughby 1985, Cullum 1994), and others have explored various ways to get around this problem in practice. While Cullum and Willoughby developed a method to distinguish spurious eigenvalues from true ones, Parlett's group chose to avoid the spurious ones from the beginning by *partial* or *selective reorthogonalization*, that is, by orthogonalizing $\mathbf{y}_{\mathbf{temp}}$ additionally with respect to those basis vectors that were constructed earlier in certain critical steps. All this earlier work is on the symmetric Lanczos process based on three-term recurrences.

Recently, Bai (1994) generalized Paige's theory to the nonsymmetric case and Parlett's student Day (Day, III 1993, Day 1997) adopted the partial reorthogonalization technique; he refers to it in this case as *maintaining duality*. Again, certain earlier computed Lanczos vectors are included in the now two-sided Gram–Schmidt process. This is quite an effective remedy for the loss of biorthogonality, but of course, it causes considerable overhead in memory requirement, computational cost, and program complexity. For eigenvalue computations this extra effort may well be worthwhile, but for linear solvers it seems too costly. Moreover, it is impossible to extend this technique to squared and product methods.

Day also aims at reducing the local error as much as possible. First, since the BIO algorithm involves two-sided Gram–Schmidt orthogonalization, we can implement it in the modified form, that is, replace (2.21e) and (2.21f) by

$$\begin{aligned} \mathbf{y}_{\text{temp}} &:= \mathbf{A}\mathbf{y}_n - \mathbf{y}_{n-1}\beta_{n-1}, & \widetilde{\mathbf{y}}_{\text{temp}} &:= \mathbf{A}^\star\widetilde{\mathbf{y}}_n - \widetilde{\mathbf{y}}_{n-1}\widetilde{\beta}_{n-1}, \\ \alpha_n &:= \langle \widetilde{\mathbf{y}}_n, \mathbf{y}_{\text{temp}}\rangle_{\mathbf{B}} / \delta_n, & \widetilde{\alpha}_n &:= \overline{\alpha_n}, \\ \mathbf{y}_{\text{temp}} &:= \mathbf{y}_{\text{temp}} - \mathbf{y}_n\alpha_n, & \widetilde{\mathbf{y}}_{\text{temp}} &:= \widetilde{\mathbf{y}}_{\text{temp}} - \widetilde{\mathbf{y}}_n\widetilde{\alpha}_n. \end{aligned}$$

Then we can make a tiny correction in order to reinforce the orthogonality that we just took into account:

$$\begin{aligned} \partial\alpha_n &:= \langle \widetilde{\mathbf{y}}_n, \mathbf{y}_{\text{temp}}\rangle_{\mathbf{B}} / \delta_n, & \partial\widetilde{\alpha}_n &:= \langle \mathbf{y}_n, \widetilde{\mathbf{y}}_{\text{temp}}\rangle_{\mathbf{B}} / \overline{\delta_n}, \\ \alpha_n &:= \alpha_n + \partial\alpha_n, & \widetilde{\alpha}_n &:= \widetilde{\alpha}_n + \partial\widetilde{\alpha}_n, \\ \mathbf{y}_{\text{temp}} &:= \mathbf{y}_{\text{temp}} - \mathbf{y}_n\partial\alpha_n, & \widetilde{\mathbf{y}}_{\text{temp}} &:= \widetilde{\mathbf{y}}_{\text{temp}} - \widetilde{\mathbf{y}}_n\partial\widetilde{\alpha}_n. \end{aligned}$$

While Day proposes to apply this local reorthogonalization at every step, one can choose to include it only when $||\mathbf{y}_{\text{temp}}||$ or $||\widetilde{\mathbf{y}}_{\text{temp}}||$ is much shorter than $||\mathbf{A}\mathbf{y}_n||$ or $||\mathbf{A}^\star\widetilde{\mathbf{y}}_n||$, respectively. We can also implement this enhancement in LTPMs like BICGSTAB2.

18.2. Low relative accuracy of certain inner products

The inner products $\delta_n := \langle \widetilde{\mathbf{y}}_n, \mathbf{y}_n\rangle_{\mathbf{B}}$ and $\delta'_n := \langle \widetilde{\mathbf{v}}_n, \mathbf{A}\mathbf{v}_n\rangle_{\mathbf{B}}$ are crucial for determining the recurrence coefficients of the BIO and BIOMIN algorithms:

$$\alpha_n := \frac{\langle \widetilde{\mathbf{y}}_n, \mathbf{A}\mathbf{y}_n\rangle_{\mathbf{B}}}{\delta_n}, \quad \beta_{n-1} := \frac{\gamma_n\delta_n}{\delta_{n-1}}, \quad \omega_n := \frac{\delta_n}{\delta'_n}, \quad \psi_n := -\frac{\delta_{n+1}}{\delta_n}.$$

Most of the algorithms we discussed break down if one of these inner products is used but vanishes. Then one has to resort to look-ahead, which is discussed in the next section. However, typically, the look-ahead tolerance is chosen to be rather small. It is therefore quite normal to proceed without look-ahead, although

$$\frac{\delta_n}{||\widetilde{\mathbf{y}}_n||\,||\mathbf{B}\mathbf{y}_n||} \ll 1 \quad \text{or} \quad \frac{\delta'_n}{||\widetilde{\mathbf{v}}_n||\,||\mathbf{B}\mathbf{A}\mathbf{v}_n||} \ll 1,$$

and this means that δ_n and δ'_n suffer from low relative accuracy. It does not help to compute them in double precision.

Note that this is a difficulty that occurs only in the nonsymmetric case and, regarding δ'_n, the symmetric indefinite case. A possible remedy consists of switching to a suitable one-sided Lanczos process (see Section 6), but so far tests have not been so successful.

The same issue comes up in LTPMs, but there the inner products for δ_n and δ'_n are different. Taking appropriate measures leads to improved versions of algorithms from the BICGSTAB family; see Sleijpen and van der Vorst (1995*a*, 1995*b*) and Gutknecht and Ressel (1997).

18.3. Inaccurate Krylov space extension

The generation of well-conditioned Krylov space bases is the prime aim of the Lanczos process. In this regard it is important that in the recursions the term that increases the dimension is not small compared to those that lie in the current subspace. For example, in the BIO algorithm it is dangerous if $||\mathbf{A}\mathbf{y}_n||$ is considerably smaller than $||\mathbf{y}_n\alpha_n||$ or $||\mathbf{y}_{n-1}\beta_{n-1}||$; in the BIOC algorithm it is dangerous if $||\mathbf{A}\mathbf{v}_n||$ is much smaller than $||\mathbf{y}_n\varphi_n||$; and in the BICGSTAB algorithm $||\mathbf{A}\widehat{\mathbf{w}}_n^n\omega_n||$ should not be small compared to $||\mathbf{w}_n^n||$, nor should $||\mathbf{A}\mathbf{w}_{n+1}^n\chi_{n+1}||$ be small compared to $||\mathbf{w}_{n+1}^n||$. While one has a choice to modify χ_{n+1} in BICGSTAB2, the other cases call for the application of look-ahead; see Section 19.

18.4. Deviation between recursive and true residual

Krylov space solvers normally update the residuals recursively, since the computation of the true residual $\mathbf{b} - \mathbf{A}\mathbf{x}_n$ costs an extra matrix-vector multiplication and, according to folklore, convergence is slower if the true residuals are used for further computation; see, for instance, van der Vorst (1992) and Greenbaum (1997). However, while the size of the true residuals is bounded below by the round-off errors that occur in its evaluation, recursively computed residuals keep getting smaller and smaller if a method converges. This is easy to understand for most Krylov space solvers: they are scale-invariant with respect to the size of the residuals, as long as we neglect the iterates. Therefore, at some point the recursively computed and the true residuals necessarily start to deviate from each other completely. Normally, from then on the true residual remains on roughly the same level, while the recursive one continues to decrease.

Numerical experiments readily show that this branch point is not just determined by the round-off in the evaluation of the true residual, but that it may be reached much earlier. In particular, examples with BICG or BICGS with peaks in the residual norm plot that are much higher than $||\mathbf{r}_0||$ seem to indicate that the tallest peak, $\max||\mathbf{r}_n||$, determines the ultimate

level of the true residual. Sleijpen et al. (1994) provide a theory to support this observation. However, a more careful analysis of Greenbaum (1994*a*, 1997) shows that under the assumption that the Krylov space solver uses direction vectors, as in (17.11), it is the maximum norm of the iterates (or the corrections) that matters. Of course, very high peaks in the residual norm plot normally go along with iterates whose norm exceeds by far the one of $\mathbf{x}_{\mathrm{ex}}$. Therefore such peaks normally imply a loss of accuracy.

By estimating the round-off in the evaluation of $\mathbf{x}_{n+1}$ and $\mathbf{r}_{n+1}$ according to (17.11), Greenbaum (1997) finds the following result.

Theorem 18.1 If iterates and residuals are updated according to (17.11), the difference between the true residual $\mathbf{b} - \mathbf{A}\mathbf{x}_n$ and the recursively computed residual $\mathbf{r}_n$ satisfies

$$\frac{||\mathbf{b} - \mathbf{A}\mathbf{x}_n - \mathbf{r}_n||}{||\mathbf{A}||\,||\mathbf{x}||} \leq (\epsilon + \mathcal{O}(\epsilon^2))\,[n + 2 + (1 + \gamma + (n+1)(10 + 2\gamma))\Theta_n]\,,$$

where ϵ denotes the machine-epsilon, γ is a constant that is needed to estimate the round-off in the matrix-vector product according to

$$||\mathbf{A}\mathbf{v}_n - \mathrm{fl}(\mathbf{A}\mathbf{v}_n)|| \leq \gamma\epsilon||\mathbf{A}||\,||\mathbf{v}_n||,$$

and

$$\Theta_n := \max_{k \leq n} \frac{||\mathbf{x}_k||}{||\mathbf{x}_{\mathrm{ex}}||}.$$

This estimate is only based on (17.11), and it does not matter where the direction vectors $\mathbf{v}_n$ and the step sizes ω_n come from. Therefore, errors in these quantities do not influence the gap between true and recursive residuals. Clearly, the theorem not only applies to BIOMIN, but, for instance, also to the smoothing process (17.12) and to BICGSTAB if we force (16.9e) to be executed in the given order. It does not directly apply to our general $(3, 2 \times 2)$-type LTPM algorithm (Algorithm 16), since the latter also involves three-term recurrences. For these the maximum residual plays an essential role too. Applying these considerations to the BICGSTAB(2) recursions (16.14a)–(16.14b), we conclude that if the Krylov space vectors associated with $\zeta t_{l-1}(\zeta)$ and $\zeta^2 t_{l-1}(\zeta)$ are close to being linearly dependent, the corresponding terms in

$$\mathbf{w}_{n+1}^{n+1} := \mathbf{w}_{n+1}^{n-1} + \mathbf{A}\mathbf{w}_{n+1}^{n-1}\chi_n + \mathbf{A}^2\mathbf{w}_{n+1}^{n-1}\chi_{n+1}$$

could be large compared to $\mathbf{w}_{n+1}^{n-1}$ and $\mathbf{w}_{n+1}^{n+1}$, thus causing a large deviation between $\mathbf{x}_{n+1}^{n+1}$ and its recursively computed residual $\mathbf{w}_{n+1}^{n+1}$.

It is often possible to overcome the loss of accuracy discussed above by a modification that was first proposed for BIOMINS by Neumaier (1994), but is equally applicable to other methods, although for some it will cost an additional matrix-vector multiplication per step; see Sleijpen and van der

Vorst (1996). One can think of it as a repeated shift of the origin or an *implicit iterative refinement*. First, we let

$$\mathbf{b}' := \mathbf{b} - \mathbf{A}\mathbf{x}_0, \quad \mathbf{x}' := \mathbf{x}_0, \quad \mathbf{x}_0 := \mathbf{o},$$

so that $\mathbf{b} - \mathbf{A}\mathbf{x} = \mathbf{b}' - \mathbf{A}\mathbf{h}$, where $\mathbf{h} := \mathbf{x} - \mathbf{x}'$. We then apply our algorithm of choice to $\mathbf{A}\mathbf{h} = \mathbf{b}'$. At step n, if the *update condition*

$$||\mathbf{r}_n|| < ||\mathbf{b}'||\,\gamma' \quad (\text{where } \gamma' \in (0,1] \text{ is given}) \tag{18.1}$$

is satisfied, we include the reassignments

$$\mathbf{b}' := \mathbf{b}' - \mathbf{A}\mathbf{x}_n, \quad \mathbf{x}' := \mathbf{x}' + \mathbf{x}_n, \quad \mathbf{x}_n := \mathbf{o}. \tag{18.2}$$

Note that at every step, we then have

$$\mathbf{r}_n = \mathbf{b}' - \mathbf{A}\mathbf{x}_n = \mathbf{b} - \mathbf{A}(\mathbf{x}' + \mathbf{x}_n).$$

Neumaier actually computed the true residual at every step and chose $\gamma' = 1$, which means that the update is performed at every step where the residual decreases, hence, nearly always. Sleijpen and van der Vorst (1996) followed up on this idea, provided an analysis, and suggested several alternatives to the update condition (18.1). According to our numerical tests, the best strategy depends on the particular example.

In general, each update (18.2) requires an extra matrix-vector product. However, Neumaier found a way to use it in BIOMINS for replacing one of the two other such products, and Sleijpen and van der Vorst achieved the same for BICGSTAB.

19. Look-ahead Lanczos algorithms

Look-ahead Lanczos algorithms are extensions of Lanczos-type algorithms, in particular the BIO and BIOC algorithms, that circumvent the breakdowns one might encounter, or at least most of them. The look-ahead BIO algorithm was first thought of by Gragg (1974) in a paper on matrix interpretations of recursions for continued fractions and Padé fractions; however, he only followed up on his idea in the context of the partial realization problem of system theory (Gragg and Lindquist 1983). Years later it materialized in Taylor's thesis (Taylor 1982) and the paper by Parlett et al. (1985), who, however, concentrated on steps of length two only and used a very different approach, namely by thinking of the BIO algorithm as a two-sided Gram–Schmidt process. Look-ahead was then rediscovered by Gutknecht (1992, 1994*a*) in connection with work on continued fractions associated with rational interpolation (Gutknecht 1989*a*) and joint work with Gene Golub on the modified Chebyshev algorithm (Golub and Gutknecht 1990). At the

same time[18], the subject was also taken up by Joubert (1990) and Parlett (1992), and latterly also by Boley, Elhay, Golub and Gutknecht (1991), Freund et al. (1993), Nachtigal (1991), Brezinski, Redivo Zaglia and Sadok (1992*b*), Hochbruck (1992), and others. Much of the earlier work (including Gragg's note) was on exact breakdowns, and the idea was to treat near-breakdowns as if they were exact ones. But Gutknecht (1994*a*, Sections 9 and 10), Freund et al. (1993), Hochbruck (1992), Nachtigal (1991), and Parlett (1992) addressed explicitly the general near-breakdown case. Moreover, in contrast to the continued fraction approach, the two-sided Gram–Schmidt approach requires no modification for near-breakdowns.

We first derive a *look-ahead* BIO *algorithm*, or LABIO for short, and then treat an analogue *look-ahead* BIOC *algorithm* with mixed recurrences. In both cases we first describe the standard versions as given in Gutknecht (1994*a*), Freund et al. (1993), and Freund and Nachtigal (1994) and then describe recently found simplifications. We do not state recursions for the iterates, because these two look-ahead algorithms are normally coupled either with QMR or with 'inconsistent' update formulae for the Galerkin iterates, as in our inconsistent BIORES and BIOMIN algorithms. Finally we refer to some other look-ahead algorithms, including the *composite-step* BICG *algorithm* of Bank and Chan (1993) and Bank and Chan (1994) and the GMRZ and other algorithms of Brezinski, Redivo Zaglia and Sadok (1991).

In retrospect, the LABIO algorithm can be understood as follows: if a breakdown or near-breakdown of the BIO algorithm occurs, we avoid it by temporarily reducing the number of orthogonality conditions that have to be fulfilled, in fact dropping the biorthogonality to the most recent basis vectors. It turns out that in all but certain very exceptional situations (which are referred to as *incurable breakdowns*) one can return to the full set of conditions after just one or a few steps. The resulting pairs of Lanczos vectors are then in a certain way block biorthogonal.

One could also consider making use of the freedom capitalized upon in the one-sided Lanczos algorithm and apply look-ahead only to the right Lanczos vectors.

As before, we could allow for a formal inner product matrix $\mathbf{B}$ that commutes with $\mathbf{A}$, but for simplicity we assume $\mathbf{B} = \mathbf{I}$.

19.1. LABIO*: the look-ahead* BIO *algorithm*

We know from Lemma 12.2 that a regular Lanczos polynomial, and, thus, a pair of *regular* Lanczos vectors $\tilde{\mathbf{y}}_n$, $\mathbf{y}_n$ ($\neq \mathbf{o}$) satisfying (2.4), exists if and only if the nth leading principal submatrix $\mathbf{M}_n$ of the moment matrix

[18] The publication dates are misleading; one has to look at the dates where the papers were submitted or preprints were made available.

is nonsingular. In finite precision arithmetic we also need to avoid near-singular sections; hence, we want to compute a pair of regular Lanczos vectors only if $\mathbf{M}_n$ is in a certain sense well-conditioned. However, we cannot enforce this requirement directly: first, $\mathbf{M}_n$ itself is not available and even if it were, we would not want to have to compute its condition number; second, leading principal submatrices of a Hankel matrix are notorious for their bad condition; and third, the formulae below will show that, primarily, certain other matrices need to be well-conditioned.

We let $0 = n_0 < n_1 < n_2 < \ldots$ be a set of indices, where we can and want to enforce all orthogonality conditions, that is, where

$$\widetilde{\mathcal{K}}_{n_j} \perp \mathbf{y}_{n_j}, \quad \widetilde{\mathbf{y}}_{n_j} \perp \mathcal{K}_{n_j}. \tag{19.1}$$

We will call these indices *well-conditioned*, and likewise the corresponding Lanczos polynomial and vectors will be referred to as well-conditioned. Clearly, if a Lanczos polynomial is well-conditioned, then it is also regular. A step for constructing well-conditioned Lanczos vectors is therefore sometimes called a *regular step*. When n is not a well-conditioned index, we refer to $\mathbf{y}_n$ and $\widetilde{\mathbf{y}}_n$ as *inner vectors*; and we call a step for computing these vectors an *inner step*. Of course, the well-conditioned indices will be chosen such that the problem of finding the Lanczos polynomial and the Lanczos vectors is well-conditioned in the usual sense. We aim at generating sequences of Lanczos vectors that satisfy (2.5) and (2.6) as before, but where (2.4) is relaxed to

$$\widetilde{\mathcal{K}}_{n_l} \perp \mathbf{y}_n, \quad \widetilde{\mathbf{y}}_n \perp \mathcal{K}_{n_l} \quad \text{if } n_l \leq n < n_{l+1}. \tag{19.2}$$

This implies that these sequences are block-biorthogonal: if we define matrix blocks containing groups of Lanczos vectors,

$$Y_l := [\ \mathbf{y}_{n_l} \quad \mathbf{y}_{n_l+1} \quad \cdots \quad \mathbf{y}_{n_{l+1}-1}\ ], \quad \widetilde{Y}_l := [\ \widetilde{\mathbf{y}}_{n_l} \quad \widetilde{\mathbf{y}}_{n_l+1} \quad \cdots \quad \widetilde{\mathbf{y}}_{n_{l+1}-1}\ ], \tag{19.3}$$

and matrix blocks containing the inner products of these vectors,

$$D_l := \widetilde{Y}_l^{\star} Y_l = (\langle \widetilde{\mathbf{y}}_i, \mathbf{y}_k \rangle)_{i,k=n_l}^{n_{l+1}-1}, \tag{19.4}$$

then we have

$$\widetilde{Y}_j^{\star} Y_l = \begin{cases} D_l & \text{if } j = l, \\ O & \text{if } j \neq l. \end{cases} \tag{19.5}$$

For the derivation of the recurrences we assume that every n is associated with the l for which (19.2) holds. We let

$$h_l := n_{l+1} - n_l,$$

and denote the possibly incompleted last blocks by

$$Y_{l;n} := [\ \mathbf{y}_{n_l} \quad \mathbf{y}_{n_l+1} \quad \cdots \quad \mathbf{y}_n\ ], \quad \widetilde{Y}_{l;n} := [\ \widetilde{\mathbf{y}}_{n_l} \quad \widetilde{\mathbf{y}}_{n_l+1} \quad \cdots \quad \widetilde{\mathbf{y}}_n\ ], \tag{19.6}$$

and

$$D_{l;n} := (\langle \tilde{\mathbf{y}}_i, \mathbf{y}_k \rangle)_{i,k=n_l}^{n}. \tag{19.7}$$

Then we have, in consistency with our earlier notation,

$$\mathbf{Y}_{n+1} := [\ Y_0 \ \cdots \ Y_{l-1} \ \ Y_{l;n}\ ], \quad \tilde{\mathbf{Y}}_{n+1} := [\ \tilde{Y}_0 \ \cdots \ \tilde{Y}_{l-1} \ \ \tilde{Y}_{l;n}\ ], \tag{19.8}$$

and (19.5) leads to

$$\tilde{\mathbf{Y}}_{n+1}^{\star}\mathbf{Y}_{n+1} = \mathbf{D}_{n+1} := \text{block diag}\,(D_0, \ldots, D_{l-1}, D_{l;n}). \tag{19.9}$$

We still have to show that such sequences of Lanczos vectors exist, and how they can be constructed. Let us for the moment assume that they exist. Clearly, like any Krylov space basis, they could be constructed according to (2.7), or, in shorthand notation, by

$$\mathbf{A}\mathbf{Y}_n = \mathbf{Y}_{n+1}\underline{\mathbf{T}}_n, \quad \mathbf{A}^{\star}\tilde{\mathbf{Y}}_n = \tilde{\mathbf{Y}}_{n+1}\underline{\tilde{\mathbf{T}}}_n. \tag{19.10}$$

Assuming that the last block is completed, we conclude from (19.1) or (19.5) that $\tilde{\mathbf{Y}}_n^{\star}\mathbf{Y}_{n+1} = [\ \mathbf{D}_n \mid \mathbf{o}\ ]$ and $\tilde{\mathbf{Y}}_{n+1}^{\star}\mathbf{Y}_n = [\ \mathbf{D}_n^{\top} \mid \mathbf{o}\ ]^{\top}$, so that, as in (2.12),

$$\mathbf{D}_n\mathbf{T}_n = \tilde{\mathbf{Y}}_n^{\star}\mathbf{A}\mathbf{Y}_n = \tilde{\mathbf{T}}_n^{\star}\mathbf{D}_n \quad \text{if } n = n_{l+1} - 1.$$

Here, the product on the left is a block upper Hessenberg matrix, while the one on the right is a block lower Hessenberg matrix. Consequently, it must be block tridiagonal, and thus $\mathbf{T}_n$ and $\tilde{\mathbf{T}}_n$ are themselves block tridiagonal, in addition to being Hessenberg matrices. When all this is formulated in terms of the Lanczos polynomials instead of the Lanczos vectors, as in Gutknecht (1994*a*), it is obvious that we can still choose $\tilde{\mathbf{T}}_n = \overline{\mathbf{T}_n}$. If we want to allow for independent scale factors $\tilde{\gamma}_n$ and γ_n for the left and right Lanczos vectors, we can achieve this by diagonal scaling as in (2.29). Therefore, in the following we only derive the formulae for the elements of $\mathbf{T}_n$; we know that those for $\tilde{\mathbf{T}}_n$ look analogous and do not require computing additional inner products.

So, when the lth block is just completed, that is, when $n = n_{l+1} - 1$, then $\mathbf{T}_n$ is of the form

$$\mathbf{T}_n =: \begin{bmatrix} A_0 & B_0 & & & \\ C_0 & A_1 & B_1 & & \\ & C_1 & A_2 & \ddots & \\ & & \ddots & \ddots & B_{l-1} \\ & & & C_{l-1} & A_l \end{bmatrix},$$

where

$$B_{l-1} =: [\ b_{n_l} \ \cdots \ b_n\ ]$$

is a block of size $h_{l-1} \times h_l$ that is in general full, C_{l-1} is a $h_{l-1} \times h_l$ block

that is zero except for the element γ_{n_l-1} in the upper right corner, and A_l is a $h_l \times h_l$ block of Hessenberg form that we write as

$$A_l =: \begin{bmatrix} \alpha_{n_l} & & & & \\ \gamma_{n_l} & a_{n_l+1} & \cdots & a_{n-1} & a_n \\ & \gamma_{n_l+1} & & & \\ & & \ddots & & \\ & & & \gamma_{n-1} & \end{bmatrix}$$

The extended matrix $\underline{\mathbf{T}}_n$ has at the bottom the additional row $[\ \mathbf{o}^\top \mid \gamma_n\ ]$. In this notation the recurrences for the Lanczos vectors can be written as

$$\left.\begin{aligned} \mathbf{y}_{n+1} &:= (\mathbf{A}\mathbf{y}_n - Y_{l;n}a_n - Y_{l-1}b_n)/\gamma_n \\ \widetilde{\mathbf{y}}_{n+1} &:= (\mathbf{A}^\star\widetilde{\mathbf{y}}_n - \widetilde{Y}_{l;n}\widetilde{a}_n - \widetilde{Y}_{l-1}\widetilde{b}_n)/\widetilde{\gamma}_n \end{aligned}\right\} \quad \text{if } n_l < n+1 \le n_{l+1}. \quad (19.11)$$

Here, γ_n and $\widetilde{\gamma}_n$ are again used to normalize the Lanczos vectors.

By now we know that if there exist Lanczos vectors that fulfil (19.2), then they satisfy the above recurrence. It is straightforward to see by induction that, conversely, these recurrences produce such vectors if

(i) we choose the well-conditioned indices such that the diagonal blocks D_l are well-conditioned

(ii) we determine b_n such that the biorthogonality to the previous block is enforced

(iii) we determine a_n in a regular step (that is, when $n+1 = n_{l+1}$) such that the biorthogonality to the just completed block is enforced.

In the inner steps, a_n can be chosen arbitrarily. In particular, to reduce the computational work, we can let it be the zero vector, although for stability we might want to choose differently. For example, one can use these parameters to make the right Lanczos vectors within a block orthogonal to each other, as proposed in Boley et al. (1991).

Enforcing the mentioned conditions readily yields

$$\begin{aligned} b_n &:= D_{l-1}^{-1}\widetilde{Y}_{l-1}^\star\mathbf{A}\mathbf{y}_n, \quad & \widetilde{b}_n &:= D_{l-1}^{\star-1}Y_{l-1}^\star\mathbf{A}^\star\widetilde{\mathbf{y}}_n, \quad & &\text{if } n_l < n+1 \le n_{l+1}, \\ a_n &:= D_l^{-1}\widetilde{Y}_l^\star\mathbf{A}\mathbf{y}_n, \quad & \widetilde{a}_n &:= D_l^{\star-1}Y_l^\star\mathbf{A}^\star\widetilde{\mathbf{y}}_n, \quad & &\text{if } n+1 = n_{l+1}. \end{aligned} \quad (19.12)$$

Formulae (19.11) and (19.12) are the standard ones for a look-ahead step, as given in Gutknecht (1994*a*, §9) and Freund et al. (1993). In the latter paper it is pointed out that by making use of recursions among the inner products the large number of inner products that seem to be needed for evaluating (19.4) and (19.12) can be reduced to just $2h_l$, the same number as for h_l normal steps. Normalizing the Lanczos vectors costs another $2h_l$ inner products. Moreover, one has to store the current and the previous pair of blocks of Lanczos vectors.

However, there is a way to simplify recurrence (19.11). In the formula for b_n in (19.12), we note that due to (19.2) only the last column of $\widetilde{Y}_{l-1}$ contributes to $\widetilde{Y}_{l-1}^{\star}\mathbf{A}\mathbf{y}_n$. Thus, we can replace $\widetilde{Y}_{l-1}^{\star}$ by $\mathbf{l}\,\widetilde{\mathbf{y}}_{n_l-1}^{\star}$, where $\mathbf{l} := \mathbf{l}_{h_{l-1}} := [\ 0\ \ \cdots\ \ 0\ \ 1\ ]^{\top} \in \mathbb{R}^{h_{l-1}}$. Consequently, if we let

$$\mathbf{y}'_{l-1} := Y_{l-1}D_{l-1}^{-1}\mathbf{l}, \quad \widetilde{\mathbf{y}}'_{l-1} := \widetilde{Y}_{l-1}D_{l-1}^{\star -1}\mathbf{l},$$

and

$$\beta'_{n-1} := \widetilde{\mathbf{y}}_{n_l-1}^{\star}\mathbf{A}\mathbf{y}_n, \quad \widetilde{\beta}'_{n-1} := \mathbf{y}_{n_l-1}^{\star}\mathbf{A}^{\star}\widetilde{\mathbf{y}}_n \quad (n_l \le n < n_{l+1}),$$

then

$$Y_{l-1}b_n = (Y_{l-1}D_{l-1}^{-1}\mathbf{l})(\widetilde{\mathbf{y}}_{n_l-1}^{\star}\mathbf{A}\mathbf{y}_n) = \mathbf{y}'_{l-1}\beta'_{n-1},$$

and likewise $\widetilde{Y}_{l-1}\widetilde{b}_n = \widetilde{\mathbf{y}}'_{l-1}\widetilde{\beta}'_{n-1}$. Moreover, using the same argument again,

$$(\widetilde{\mathbf{y}}'_{l-1})^{\star}\mathbf{A}\mathbf{y}_n = \mathbf{l}^{\top}D_{l-1}^{-1}\widetilde{Y}_{l-1}^{\star}\mathbf{A}\mathbf{y}_n = (\mathbf{l}^{\top}D_{l-1}^{-1}\mathbf{l})\widetilde{\mathbf{y}}_{n_l-1}^{\star}\mathbf{A}\mathbf{y}_n = (\mathbf{l}^{\top}D_{l-1}^{-1}\mathbf{l})\beta'_{n-1}.$$

Thus we can redefine β'_{n-1} in terms of the new vector $\widetilde{\mathbf{y}}'_{l-1}$ instead of $\widetilde{\mathbf{y}}_{n_l-1}$, and the analogue holds for $\widetilde{\beta}'_{n-1}$:

$$\left.\begin{aligned} \beta'_{n-1} &:= (\mathbf{l}^{\top}D_{l-1}^{-1}\mathbf{l})^{-1}(\widetilde{\mathbf{y}}'_{l-1})^{\star}\mathbf{A}\mathbf{y}_n \\ \widetilde{\beta}'_{n-1} &:= (\mathbf{l}^{\top}D_{l-1}^{-1}\mathbf{l})^{-1}(\mathbf{y}'_{l-1})^{\star}\mathbf{A}^{\star}\widetilde{\mathbf{y}}_n \end{aligned}\right\} \quad (n_l < n+1 \le n_{l+1}). \tag{19.13}$$

The parenthesis contains only the bottom right element of D_{l-1}^{-1}. Putting the pieces together we see that the recursions (19.11) simplify to

$$\left.\begin{aligned} \mathbf{y}_{n+1} &:= (\mathbf{A}\mathbf{y}_n - Y_{l;n}a_n - \mathbf{y}'_{l-1}\beta'_{n-1})/\gamma_n \\ \widetilde{\mathbf{y}}_{n+1} &:= (\mathbf{A}^{\star}\widetilde{\mathbf{y}}_n - \widetilde{Y}_{l;n}\widetilde{a}_n - \widetilde{\mathbf{y}}'_{l-1}\widetilde{\beta}'_{n-1})/\widetilde{\gamma}_n \end{aligned}\right\} \quad \text{if } n_l < n+1 \le n_{l+1}. \tag{19.14}$$

In other words, the previous blocks are replaced by single vectors $\mathbf{y}'_{l-1}$ and $\widetilde{\mathbf{y}}'_{l-1}$, respectively. It can be shown (Hochbruck 1996) that

$$\widetilde{\mathcal{K}}_{n_l-1} \perp \mathbf{y}'_{l-1}, \quad \widetilde{\mathbf{y}}'_{l-1} \perp \mathcal{K}_{n_l-1}. \tag{19.15}$$

At the root of this simplification is the fact that the blocks

$$B_{l-1} = D_{l-1}^{-1}\mathbf{l}\,\widetilde{\mathbf{y}}_{n_l-1}^{\star}\mathbf{A}Y_l$$

are of rank one, as was pointed out in Gutknecht (1992). Using a similar argument Freund and Zha (1993) implicitly capitalized upon this in their Hankel solver, but only recently it was pointed out by Hochbruck (1996), who used yet another approach, that this leads to the above simplification of the look-ahead Lanczos process. In the polynomial formulation the vectors $\mathbf{y}'_{l-1}$ and $\mathbf{y}_{n_l}$ correspond to polynomials that are the denominators of a regular (or, well-conditioned) pair of Padé approximants; see Gutknecht (1993*b*), Gutknecht and Gragg (1994). In the case of an exact breakdown, the above simplification is irrelevant since the block B_{l-1} then has only a

single nonzero element, the one in the upper right corner, and $\mathbf{y}'_{l-1}$ is then a multiple of $\mathbf{y}_{n_{l-1}}$; see Gutknecht (1992).

A look-ahead algorithm always requires a *look-ahead strategy*, a recipe as to when to start a look-ahead step and when to terminate it. The above formulae clearly indicate that we need to avoid singular or near-singular blocks D_j, and this suggests requiring that the smallest singular value, $\sigma_{\min}(D_j)$, be larger than a certain tolerance. Indeed, Parlett (1992) showed that the following quantitative result holds: if the Lanczos vectors are normalized, then

$$\min\left\{\sigma_{\min}(\widetilde{\mathbf{Y}}_{n_l}),\ \sigma_{\min}(\mathbf{Y}_{n_l})\right\} \geq \frac{1}{\sqrt{n_l+1}} \min_{0\leq j\leq l} \sigma_{\min}(D_j),$$

However, numerical examples reported in Freund et al. (1993) showed that, in finite precision arithmetic, we should not rely on this result and not monitor $\sigma_{\min}(D_{l;n})$ alone. Instead, it is suggested that the 1-norms of the coefficient vectors in (19.11) be kept below a certain bound when $n+1 = n_{l+1}$, that is, when the new index $n+1$ is declared as well-conditioned:

$$||a_n||_1 \leq \Upsilon,\ \ ||\widetilde{a}_n||_1 \leq \Upsilon,\ \ ||b_n||_1 \leq \Upsilon,\ \ ||\widetilde{b}_n||_1 \leq \Upsilon \implies n_{l+1} := n+1.$$

Here, the bound Υ depends on $\mathbf{A}$ and is typically of the order of $||\mathbf{A}||$. For example, one can start with $\Upsilon := \max\{||\mathbf{A}\mathbf{y}_0||, ||\mathbf{A}^\star\widetilde{\mathbf{y}}_0||\}$ and then adjust this bound dynamically during the algorithm. See Freund et al. (1993) for more details. If the simplified formulae (19.14) are used, the bound for b_n and $\widetilde{b}_n$ can be replaced by one for $\mathbf{y}'_{l-1}\beta'_{n-1}$ and $\widetilde{\mathbf{y}}'_{l-1}\widetilde{\beta}'_{n-1}$. There is, unfortunately, the possibility that $D_{l;n}$ remains singular or ill-conditioned for all $n > n_l$. This is what is called an *incurable breakdown*. Then there is no way to escape a restart of the algorithm, but when solving a linear system one can of course restart from the most recent approximation.

Clearly, the same approach can be used to specify an LABIC algorithm. The look-ahead approach for Lanczos-type product methods (LTPMs) introduced in Gutknecht and Ressel (1996) is based on the standard look-ahead procedure, but can also accommodate the above simplification.

19.2. LABIOC: *the look-ahead* BIOC *algorithm*

The BIOC algorithm is susceptible to both Lanczos and pivot breakdowns, and thus a look-ahead generalization of it should be able to cope with both. In addition to the sequence of regular indices n_l for the Lanczos vectors, we therefore have a second sequence of regular indices m_k for the direction vectors. From Lemma 12.2 we know that $\det \mathbf{M}_{n_l} \neq 0$ and $\det \mathbf{M}'_{m_k} \neq 0$ but, as in the last subsection, we only consider here those regular indices that are in some sense well-conditioned. In addition to (19.2) we now want

to enforce

$$\widetilde{\mathcal{K}}_{m_k} \perp \mathbf{A}\mathbf{v}_n, \quad \mathbf{A}^\star\widetilde{\mathbf{v}}_n \perp \mathcal{K}_{m_k}, \quad \text{if } m_k \le n < m_{k+1}. \tag{19.16}$$

In the following, we associate with every n a pair (k,l) such that

$$m_k < n \le m_{k+1} \quad \text{and} \quad n_l \le n < n_{l+1} \tag{19.17}$$

hold, and we let $h'_k := m_{k+1} - m_k$. Defining, in analogy to (19.3)–(19.4) and (19.6)–(19.8), blocks V_k, $\widetilde{V}_k$, D'_k, $V_{k;n}$, $\widetilde{V}_{k;n}$, and $D'_{k;n}$, we then have, as in (19.9),

$$\widetilde{\mathbf{V}}^\star_{n+1}\mathbf{A}\mathbf{V}_{n+1} = \mathbf{D}'_{n+1} := \text{block diag}\,(D'_0, \ldots, D'_{k-1}, D'_{k;n}). \tag{19.18}$$

Since we want each of the sets $\{\mathbf{y}_i\}$, $\{\widetilde{\mathbf{y}}_i\}$, $\{\mathbf{v}_i\}$, $\{\widetilde{\mathbf{v}}_i\}$ to be a nested basis of the respective Krylov space, it is clear that there should exist recurrences with a matrix representation

$$\begin{aligned} \mathbf{Y}_n &= \mathbf{V}_n\mathbf{U}_n, & \mathbf{A}\mathbf{V}_n &= \mathbf{Y}_{n+1}\underline{\mathbf{L}}_n, \\ \widetilde{\mathbf{Y}}_n &= \widetilde{\mathbf{V}}_n\widetilde{\mathbf{U}}_n, & \mathbf{A}^\star\widetilde{\mathbf{V}}_n &= \widetilde{\mathbf{Y}}_{n+1}\underline{\widetilde{\mathbf{L}}}_n, \end{aligned} \tag{19.19}$$

where $\underline{\mathbf{L}}_n$ and $\underline{\widetilde{\mathbf{L}}}_n$ are of upper Hessenberg form and $\mathbf{U}_n$ and $\widetilde{\mathbf{U}}_n$ are unit upper triangular. From the polynomial formulation of the BIO algorithm we could again readily conclude that we can assume $\underline{\widetilde{\mathbf{L}}}_n$ and $\widetilde{\mathbf{U}}_n$ are diagonally scaled versions of $\overline{\underline{\mathbf{L}}_n}$ and $\overline{\mathbf{U}_n}$, respectively. Moreover, as in Section 7, we can return to (19.10) by eliminating $\mathbf{V}_n$ and $\widetilde{\mathbf{V}}_n$ from (19.19), while by eliminating $\mathbf{Y}_n$ and $\widetilde{\mathbf{Y}}_n$ we find a block version of (7.10), if we assume the definitions $\underline{\mathbf{T}}_n := \underline{\mathbf{L}}_n\mathbf{U}_n$ and $\underline{\mathbf{T}}'_n := \mathbf{U}_{n+1}\underline{\mathbf{L}}_n$ from (7.8). But we cannot conclude that these are block LU and block UL factorizations of the block triangular matrices $\underline{\mathbf{T}}_n$ and $\underline{\mathbf{T}}'_n$, respectively. The block triangularity of $\underline{\mathbf{T}}'_n$ can be verified as above for $\underline{\mathbf{T}}_n$, but we must keep in mind that the blocks can be of different sizes to those of $\underline{\mathbf{T}}_n$.

If only exact breakdowns are considered, it has been shown that we indeed have block factorizations and that the block sizes are linked to each other in a well-defined way; see Gutknecht (1994*a*). The link between the block sizes follows directly from the block structure theorem for the Padé table. If near-breakdowns are included, there are still arguments to have the breakdowns linked, but not in the same rigid way: typically, for every n_l there is an m_k such that $n_l = m_k$ or $n_l = m_k + 1$. In the first case, $(\widehat{p}_{n_l-1}, p_{n_l})$ is a row-regular (or a row-well-conditioned) pair; in the second case, $(p_{n_l-1}, \widehat{p}_{n_l-1})$ is a column-regular (or a column-well-conditioned) pair; see Hochbruck (1996). The notions of row-regularity and column-regularity, which play a crucial role here, were introduced in Gutknecht (1993*b*). They mean that the respective polynomials (which are FOPS for the two closely related functionals Φ and Φ' of Section 12) are not scalar multiples of each

other; and they imply that these polynomials are even relatively prime; see also Gutknecht and Hochbruck (1995), Gutknecht and Gragg (1994).

But let us first consider the general case: after replacing n by $n+1$ in the first line of (19.19), we can always write the last column of these matrix equations as

$$\begin{aligned} \mathbf{y}_n &= \mathbf{V}_{m_k} g_n + V_{k;n-1} g_{n;k} + \mathbf{v}_n, \\ \mathbf{A}\mathbf{v}_n &= \mathbf{Y}_{n_l} f_n + Y_{l;n} f_{n;l} + \mathbf{y}_{n+1}\gamma_n. \end{aligned} \tag{19.20}$$

The conditions (19.2) and (19.16) that led to (19.9) and (19.18), respectively, yield under assumption (19.17)

$$g_n := \mathbf{D}'^{-1}_{m_k} \widetilde{\mathbf{V}}^{\star}_{m_k} \mathbf{A}\mathbf{y}_n, \quad f_n := \mathbf{D}^{-1}_{n_l} \widetilde{\mathbf{Y}}^{\star}_{n_l} \mathbf{A}\mathbf{v}_n. \tag{19.21}$$

These formulae suggest that the recurrences (19.20) are long. However, in (19.21) only few blocks of the block diagonal matrices are multiplied by nonzero blocks of the vectors that follow. Recall that we have

$$\tilde{\mathbf{v}}_i^{\star} \mathbf{A}\mathbf{y}_n = 0 \quad \text{if } i < n_l - 1, \quad \tilde{\mathbf{y}}_i^{\star} \mathbf{A}\mathbf{v}_n = 0 \quad \text{if } i < m_k. \tag{19.22}$$

Therefore, if we define

$$\begin{aligned} k^{\star} &:= \max\{j : j \le k,\ m_j \le \max\{n_l - 1, 0\}\}, \\ l^{\star} &:= \max\{j : j \le l,\ n_j \le m_k\}, \end{aligned}$$

then in (19.21) only the blocks $D'_{k^{\star}}, \ldots, D'_{k-1}$ of $\mathbf{D}'_{m_k}$ and $D_{l^{\star}}, \ldots, D_{l-1}$ of $\mathbf{D}_{n_l}$ matter. Therefore, (19.20) becomes

$$\begin{aligned} \mathbf{v}_n &:= \mathbf{y}_n - \textstyle\sum_{j=k^{\star}}^{k-1} V_j g_{n;j} - V_{k;n-1} g_{n;k}, \\ \mathbf{y}_{n+1} &:= \left(\mathbf{A}\mathbf{v}_n - \textstyle\sum_{j=l^{\star}}^{l-1} Y_j f_{n;j} - Y_{l;n} f_{n;l}\right) \frac{1}{\gamma_n}, \end{aligned} \tag{19.23}$$

where

$$\begin{aligned} g_{n;j} &:= D'^{-1}_j \widetilde{V}^{\star}_j \mathbf{A}\mathbf{y}_n \quad && (j = k^{\star}, \ldots, k-1 \text{ if } n < m_{k+1}, \\ & && \ j = k^{\star}, \ldots, k \text{ if } n = m_{k+1}), \\ f_{n;j} &:= D^{-1}_j \widetilde{Y}^{\star}_j \mathbf{A}\mathbf{v}_n \quad && (j = l^{\star}, \ldots, l-1 \text{ if } n < n_{l+1} - 1, \\ & && \ j = l^{\star}, \ldots, l \text{ if } n = n_{l+1} - 1), \end{aligned} \tag{19.24}$$

while $g_{n;k}$ and $f_{n;l}$ are arbitrary if $n < m_{k+1}$ or $n < n_{l+1} - 1$, respectively. This means that if we compute inner vectors, then these two coefficients can be chosen as zero vectors, so that the corresponding terms in (19.23) can be dropped. The recurrences (19.23)–(19.24) are due to Freund and Nachtigal (1994); see also Freund and Nachtigal (1993). Of course, analogous formulae exist for the left-hand side vectors. This look-ahead version of the BIOC algorithm is more general than the one sketched in §10 of Gutknecht (1994*a*), because the two sequences of well-conditioned indices, $\{n_l\}$ and $\{m_k\}$ are not assumed to be linked in a certain way.

As mentioned above, typically either $n_l = m_k$ or $n_l = m_k + 1$, and then we can draw further conclusions:

$$\begin{array}{lll} n_l = m_k & \Longrightarrow \quad k^\star = k-1, & g_{n;k-1} = D_{k-1}'^{-1}\,\mathbf{l}\,\widetilde{\mathbf{v}}_{m_k-1}^\star \mathbf{A}\mathbf{y}_n, \\ & \quad\; l^\star = l & (i.e.,\ f_n = \mathbf{o}), \\ n_l = m_k + 1 & \Longrightarrow \quad k^\star = k & (i.e.,\ g_n = \mathbf{o}), \\ & \quad\; l^\star = l-1, & f_{n;l-1} = D_{l-1}^{-1}\,\mathbf{l}\,\widetilde{\mathbf{y}}_{n_l-1}^\star \mathbf{A}\mathbf{v}_n, \end{array} \tag{19.25}$$

Here, we have already taken into account that, in view of (19.22), in these situations only the last columns of $\widetilde{V}_{k-1}$ and $\widetilde{Y}_{l-1}$, respectively, yield a nonzero contribution to $\widetilde{V}_{k-1}^\star \mathbf{A}\mathbf{y}_n$ and $\widetilde{Y}_{l-1}^\star \mathbf{A}\mathbf{v}_n$. This gives rise to a simplification analogous to the one that led from (19.12) to (19.13)–(19.14). Letting

$$\begin{array}{rclrcl} \mathbf{v}_{k-1}' &:=& V_{k-1}D_{k-1}'^{-1}\mathbf{l}, & \widetilde{\mathbf{v}}_{k-1}' &:=& \widetilde{V}_{k-1}D_{k-1}'^{\star-1}\mathbf{l}, \\ \psi_n' &:=& (\mathbf{l}^\top D_{k-1}'^{-1}\mathbf{l})^{-1}(\widetilde{\mathbf{v}}_{k-1}')^\star \mathbf{A}\mathbf{y}_n, & \widetilde{\psi}_n' &:=& (\mathbf{l}^\top D_{k-1}'^{-1}\mathbf{l})^{-1}(\mathbf{v}_{k-1}')^\star \mathbf{A}^\star \widetilde{\mathbf{y}}_n, \\ \mathbf{y}_{l-1}' &:=& Y_{l-1}D_{l-1}^{-1}\mathbf{l}, & \widetilde{\mathbf{y}}_{l-1}' &:=& \widetilde{Y}_{l-1}D_{l-1}^{\star-1}\mathbf{l}, \\ \varphi_n' &:=& (\mathbf{l}^\top D_{l-1}^{-1}\mathbf{l})^{-1}(\widetilde{\mathbf{y}}_{l-1}')^\star \mathbf{A}\mathbf{v}_n, & \widetilde{\varphi}_n' &:=& (\mathbf{l}^\top D_{l-1}^{-1}\mathbf{l})^{-1}(\mathbf{y}_{l-1})^\star \mathbf{A}^\star \widetilde{\mathbf{v}}_n, \end{array} \tag{19.26}$$

we finally obtain the following simplified recurrences for the right-hand side vectors: if $n_l = m_k$, then

$$\begin{array}{rcll} \mathbf{v}_n &=& \mathbf{y}_n - \mathbf{v}_{k-1}'\psi_n' & \text{if } m_k < n < m_{k+1}, \\ \mathbf{v}_n &=& \mathbf{y}_n - V_{k;n-1}g_{n;k} - \mathbf{v}_{k-1}'\psi_n' & \text{if } n = m_{k+1}, \\ \mathbf{y}_{n+1} &=& \mathbf{A}\mathbf{v}_n/\gamma_n & \text{if } n_l \le n < n_{l+1} - 1, \\ \mathbf{y}_{n+1} &=& (\mathbf{A}\mathbf{v}_n - Y_{l;n}f_{n;l})/\gamma_n & \text{if } n = n_{l+1} - 1, \end{array} \tag{19.27}$$

while if $n_l = m_k + 1$, then

$$\begin{array}{rcll} \mathbf{v}_n &=& \mathbf{y}_n & \text{if } m_k < n < m_{k+1}, \\ \mathbf{v}_n &=& \mathbf{y}_n - V_{k;n-1}g_{n;k} & \text{if } n = m_{k+1}, \\ \mathbf{y}_{n+1} &=& (\mathbf{A}\mathbf{v}_n - \mathbf{y}_{l-1}'\varphi_n')/\gamma_n & \text{if } n_l \le n < n_{l+1} - 1, \\ \mathbf{y}_{n+1} &=& (\mathbf{A}\mathbf{v}_n - Y_{l;n}f_{n;l} - \mathbf{y}_{l-1}'\varphi_n')/\gamma_n & \text{if } n = n_{l+1} - 1. \end{array} \tag{19.28}$$

Again, analogous recurrences exist for the left-hand side vectors. The above formulae appear in polynomial formulation based on a different derivation in Hochbruck (1996). Again, the previous blocks are replaced by a single vector, and, actually, only one of the Lanczos vectors or one of the direction vectors is needed. Now, these two auxiliary vectors satisfy, respectively, (19.15) and

$$\widetilde{\mathcal{K}}_{m_k-1} \perp \mathbf{A}\mathbf{v}_{k-1}', \quad \mathbf{A}^\star \widetilde{\mathbf{v}}_{l-1}' \perp \mathcal{K}_{m_k-1}. \tag{19.29}$$

If we also know that $m_{k+1} = n_{l+1}$ or $m_{k+1} + 1 = n_{l+1}$ holds at the end of

the look-ahead step, then we can further capitalize upon this: only one of the two recurrences for the regular step in (19.27) or (19.28), respectively, is needed, but which of the two recurrences can be dropped depends on the situation at the end of the step; see Hochbruck (1996).

19.3. Other look-ahead Lanczos algorithms

A simple, but also limited, look-ahead approach is the *composite step* BICG *algorithm* of Bank and Chan (1993, 1994): it is the poor man's look-ahead Lanczos solver. It requires that no Lanczos breakdowns occur, which implies that $|m_{k+1} - m_k| \leq 2$, as one can show by arguments involving the moment matrices or the block structure theorem of the Padé table; see, for instance, Gutknecht (1990, Theorem 3.6). The algorithm is a variation of BIOMIN in which such pivot breakdowns are cured by a special double step: an undefined Galerkin iterate $\mathbf{x}_{n+1}$ is skipped, and $\mathbf{x}_{n+2}$ and its residual are then constructed according to

$$\mathbf{x}_{n+2} := \mathbf{x}_n + \mathbf{v}_n\omega_n' + \mathbf{z}_{n+1}\omega_n'', \quad \mathbf{r}_{n+2} := \mathbf{r}_n - \mathbf{A}\mathbf{v}_n\omega_n' - \mathbf{A}\mathbf{z}_{n+1}\omega_n'',$$

where $\mathbf{z}_{n+1}$ is an auxiliary vector that is itself a linear combination of $\mathbf{r}_n$ and $\mathbf{A}\mathbf{v}_n$. As we know, there are other algorithms that are not susceptible to pivot breakdowns, in particular, BIOQMR and inconsistent BIORES. The composite step BICG algorithm has the merit that it uses the standard two-term BIOMIN version of the BICG method as default.

The composite step approach has been extended to both BIOMINS (Chan and Szeto 1994) and to LTPMs based on the coupled two-term Lanczos recurrences (Chan and Szeto 1996). Incidentally, the idea can be traced back to Luenberger (1979) and Fletcher (1976), who designed for symmetric indefinite systems CG algorithms that, in case of a breakdown, make use of such double steps by exploiting the concept of *hyperbolic pairs*.

We mentioned earlier that by choosing the inner vectors appropriately in a look-ahead algorithm we can further improve its numerical stability. For the algorithm of Parlett et al. (1985), which was also restricted to steps of length at most two, Khelifi (1991) investigated this freedom and specified an optimal choice.

Recurrences for formal orthogonal polynomials (FOPs) and the closely related Padé approximants, which are so-called convergents ('partial sums') of certain continued fractions, can serve as the basis for look-ahead algorithms restricted to exact breakdowns; see Gutknecht (1992) and (1994*a*), where these algorithms are called *nongeneric*. The relevant continued fractions, the so-called *q-fractions*, can be traced back at least to Chebyshev; see Gutknecht and Gragg (1994) for some historical remarks. Brezinski, Redivo Zaglia, and Sadok have taken up this approach in a series of papers. In the first one, Brezinski et al. (1992*b*), they introduce the MRZ algorithm, which

is basically the same as the nongeneric BIODIR algorithm of Gutknecht (1994*a*). (One difference is that Brezinski et al. (1992*b*) suggest using $(\mathbf{A}^\star)^i\tilde{\mathbf{y}}_0$ as the left vectors, which does not work in finite precision arithmetic due to the extremely bad condition of this basis.) In Brezinski and Sadok (1991) the same recurrences are used to define a nongeneric version of the BIODIRS$_2$ algorithm from Section 7 of Gutknecht (1990), and in Cao (1997*a*) they are applied to the BICGSTAB family.

Brezinski et al. (1991) first suggest two methods (SMRZ and BMRZ) that apply mixed recurrences, but can handle only exact pivot breakdowns, in contrast to other nongeneric algorithms that can cure exact breakdowns of both types, such as, for instance, nongeneric BIOMIN from (Gutknecht 1994*a*). The authors then turn to the treatment of near-breakdowns in the BIODIR algorithm. Their GMRZ algorithm is based on polynomial recurrences of the form

$$\begin{aligned} \widehat{p}_{m_{k+1}}(\zeta) &:= c'_k(\zeta)\widehat{p}_{m_k}(\zeta) + d'_k(\zeta)\widehat{p}_{m_{k-1}}(\zeta), \\ \widehat{p}_{m_{k+2}}(\zeta) &:= c''_k(\zeta)\widehat{p}_{m_k}(\zeta) + d''_k(\zeta)\widehat{p}_{m_{k-1}}(\zeta), \end{aligned} \tag{19.30}$$

for the direction polynomials, where it is assumed that $\widehat{p}_{m_{k-1}}$ and $\widehat{p}_{m_k}$ as well as $\widehat{p}_{m_{k+1}}$ and $\widehat{p}_{m_{k+2}}$ are pairs of successive regular FOPs (with no ill-conditioned, but in exact arithmetic regular FOPs between them). If, for simplicity, we exclude the possibility of exact breakdowns, this means that we require that

$$m_k = m_{k-1} + 1 \quad \text{and} \quad m_{k+2} = m_{k+1} + 1. \tag{19.31}$$

In (19.30), c'_k, d'_k, c''_k, and d''_k are polynomials of the appropriate degrees, namely h'_k, $h'_k - 1$, $h'_k + 1$, and h'_k, respectively, whose coefficients are determined by enforcing the biconjugacy conditions. Translation into Krylov space notation yields recurrences for the direction vectors. Note that even when these are only applied to the right-hand side vectors, this costs $2h_k + 1$ matrix-vector multiplications with $\mathbf{A}$ (not including those that might be needed for inner products), as opposed to the $h_k + 1$ required by LABIC. Of course, the left Krylov space needs to be generated too, and the authors again suggest using $\{(\mathbf{A}^\star)^n\tilde{\mathbf{v}}_0\}$ as basis.

Additional recurrences are needed to update the iterates and the residuals of BIODIR. They are based on the following recursion for the residual polynomials:

$$p_{m_{k+1}}(\zeta) := c_k(\zeta)\widehat{p}_{m_k}(\zeta) + (1 + \zeta e_k(\zeta))p_{m_k}(\zeta), \tag{19.32}$$

where c_k and e_k have degrees $h'_k - 1$ and $h'_k - 2$ at most. Since the residual polynomials are normalized at $\zeta = 0$ and need not have full degree, one can easily verify that they are regular (in a suitably adapted sense) for the same indices m_k as the direction polynomials. However, if $\widehat{p}_{m_k}$ is a multiple of p_{m_k} (and $m_k > 0$), the above formula cannot be true since it would imply

that $p_{m_{k+1}}$ is also a multiple of p_{m_k}, while successive regular polynomials are known to be relatively prime.

The authors indeed realized later that an extra condition has to be observed; see Brezinski et al. (1993), a paper that gives an overview of the methods proposed by this group. Incidentally, the missing condition is equivalent to $(\widehat{p}_{m_k}, p_{m_k})$ being a *column-regular pair*. As mentioned, this means that the two polynomials are not scalar multiples of each other; even more, they are then automatically relatively prime. Likewise, the restriction $m_k = m_{k-1} + 1$ of (19.31) means that $\widehat{p}_{m_{k-1}}$ and $\widehat{p}_{m_k}$ are a *regular pair*, which also implies that they are relatively prime; see Gutknecht and Gragg (1994). Fortunately, column-regularity implies regularity. Hence, if we do not consider the possibility of exact breakdowns, then the GMRZ algorithm requires us to treat successive look-ahead blocks as a single large one, until we find a pair $(\widehat{p}_{m_k}, p_{m_k})$ of well-conditioned column-regular (*i.e.*, column-well-conditioned) polynomials. This means more overhead and less numerical stability than when blocks of minimum length can be used.

In Brezinski et al. (1991) (see also Brezinski, Redivo Zaglia and Sadok (1992*a*)), a BSMRZ algorithm is introduced additionally. It is supposed to cure near-breakdowns of BIOMIN and is based on (19.32) and an analogue recurrence for generating $\widehat{p}_{m_{k+1}}$. Hence, it proceeds from the pair $(p_{m_k}, \widehat{p}_{m_k})$ to the pair $(p_{m_{k+1}}, \widehat{p}_{m_{k+1}})$, and thus these pairs are again required to be column-regular. Consequently, in this algorithm too, the steps are in general longer than in our LABIOC algorithm, discussed above, even if we only allowed steps that start and end with a column-regular or row-regular pair and thus always applied either (19.27) or (19.28). Moreover, the overhead is again higher, since two matrix-vector products are needed to expand the right Krylov space.

We emphasize that this approach, which was later also applied to the BICGS method (Brezinski and Redivo Zaglia 1994) and LTPMs (Brezinski and Redivo Zaglia 1995), differs considerably from ours, described in detail above, not only because of the preference for BIODIR and BIOMIN, but because of the different type of recurrence. The connection between the two types has been clarified by Hochbruck (1996). The recursions (19.30) with the restriction (19.31) represent a special case of those of Cabay and Meleshko (1993); see also Gutknecht and Gragg (1994). To attain the generality of the Cabay–Meleshko recurrences we would have to replace $\widehat{p}_{m_{k-1}}$ and $\widehat{p}_{m_{k+1}}$ by differently defined polynomials of maximum degree m_{k-1} and m_{k+1}, respectively.

As look-ahead strategy, Brezinski et al. suggest choosing the step size h'_k such that, for a suitably chosen $\varepsilon_1 > 0$,

$$\langle (\mathbf{A}^\star)^{m_k} \widetilde{\mathbf{v}}_0, \mathbf{A}^{i+1} \mathbf{v}_{m_k} \rangle \begin{cases} \leq \varepsilon_1 & \text{if } 0 \leq i < h'_k - 1, \\ > \varepsilon_1 & \text{if } i = h'_k - 1. \end{cases} \tag{19.33}$$

Note that this condition does not guarantee the nonsingularity of the block D'_k that one would use in LABIC, and thus it cannot guarantee that $\mathbf{v}_{m_{k+1}}$ is regular, let alone well-conditioned. (As an example, consider the situation where all the off-diagonal inner products in (19.33) are ε_1 and those on the diagonal are slightly larger.) The authors did not in fact prove that the resulting linear system for the coefficients is nonsingular, but suggest prescribing in addition to (19.33) a threshold for the size of the pivots in the Gaussian elimination; see Brezinski et al.(1992*a*, 1993).

Yet another proposal for curing Lanczos breakdowns has been made by Ye (1994). However, it requires storing all the Lanczos vectors so that when a breakdown occurs, $\widetilde{\mathbf{y}}_{n+1}$ can be replaced by a 'newstart vector' that is orthogonal to all previously computed right Lanczos vectors, as in (3.3b). A corresponding linear solver with QMR smoothing is described in Tong and Ye (1996).

20. Outlook

We hope to have convinced the reader that despite some obvious difficulties (such as breakdowns and loss of biorthogonality due to round-off) the unsymmetric Lanczos process is the basis of a series of very effective and reliable algorithms. We do not expect that yet another new algorithm of this type will markedly surpass all those that we know already, but nevertheless there is still research to be done in this area: convergence is not yet well understood, further investigations on how to improve the accuracy and stability of these algorithms are worthwhile, and there is still a shortage of quality software both for conventional and parallel computer architectures.

Acknowledgments

The author is first of all indebted to Klaus Ressel for carefully reading several versions of this paper and for performing some of the recent research that is described. The section on look-ahead profited crucially from hints and comments provided by Marlis Hochbruck. Further important input came from Bill Gragg, Anne Greenbaum, Beresford Parlett, Gerard Sleijpen and Eric de Sturler. Finally, I have to thank Arieh Iserles for his patience and flexibility.

REFERENCES

J. I. Aliaga, V. Hernandez and D. L. Boley (1994), Using the block clustered nonsymmetric Lanczos algorithm to solve control problems for MIMO linear systems, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, pp. 387–389.

W. E. Arnoldi (1951), 'The principle of minimized iterations in the solution of the matrix eigenvalue problem', *Quart. Appl. Math.* **9**, 17–29.

S. F. Ashby, T. A. Manteuffel and P. E. Saylor (1990), 'A taxonomy for conjugate gradient methods', *SIAM J. Numer. Anal.* **27**, 1542–1568.

Z. Bai (1994), 'Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem', *Math. Comp.* **62**, 209–226.

R. E. Bank and T. F. Chan (1993), 'An analysis of the composite step biconjugate gradient method', *Numer. Math.* **66**, 295–319.

R. E. Bank and T. F. Chan (1994), 'A composite step bi-conjugate gradient algorithm for nonsymmetric linear systems', *Numerical Algorithms* **7**, 1–16.

R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst (1994), *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia.

T. Barth and T. Manteuffel (1994), Variable metric conjugate gradient methods, in *Advances in Numerical Methods for Large Sparse Sets of Linear Systems* (M. Natori and T. Nodera, eds), number 10 *in* 'Parallel Processing for Scientific Computing', Keio University, Yokahama, Japan, pp. 165–188.

D. Boley and G. H. Golub (1991), 'The nonsymmetric Lanczos algorithm and controllability', *Systems Control Lett.* **16**, 97–105.

D. L. Boley (1994), Krylov space methods in linear control and model reduction: A survey, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, PA, pp. 377–379.

D. L. Boley, S. Elhay, G. H. Golub and M. H. Gutknecht (1991), 'Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights', *Numerical Algorithms* **1**, 21–43.

C. Brezinski and M. Redivo Zaglia (1994), 'Treatment of near-breakdown in the CGS algorithms', *Numerical Algorithms* **7**, 33–73.

C. Brezinski and M. Redivo Zaglia (1995), 'Look-ahead in Bi-CGSTAB and other product methods for linear systems', *BIT* **35**, 169–201.

C. Brezinski and H. Sadok (1991), 'Avoiding breakdown in the CGS algorithm', *Numerical Algorithms* **1**, 199–206.

C. Brezinski, M. Redivo Zaglia and H. Sadok (1991), 'Avoiding breakdown and near-breakdown in Lanczos type algorithms', *Numerical Algorithms* **1**, 261–284.

C. Brezinski, M. Redivo Zaglia and H. Sadok (1992*a*), 'Addendum to "Avoiding breakdown and near-breakdown in Lanczos type algorithms"', *Numerical Algorithms* **2**, 133–136.

C. Brezinski, M. Redivo Zaglia and H. Sadok (1992*b*), 'A breakdown-free Lanczos' type algorithm for solving linear systems', *Numer. Math.* **63**, 29–38.

C. Brezinski, M. Redivo Zaglia and H. Sadok (1993), Breakdowns in the implementation of the Lánczos method for solving linear systems, Technical Report ANO-320, Université Lille Flandres Artois.

P. N. Brown (1991), 'A theoretical comparison of the Arnoldi and GMRES algorithms', *SIAM J. Sci. Statist. Comput.* **12**, 58–78.

S. Cabay and R. Meleshko (1993), 'A weakly stable algorithm for Padé approximants and the inversion of Hankel matrices', *SIAM J. Matrix Anal. Appl.* **14**, 735–765.

Z.-H. Cao (1997*a*), 'Avoiding breakdown in variants of the BI-CGSTAB algorithm', *Linear Algebra Appl.* To appear.

Z.-H. Cao (1997*b*), 'On the QMR approach for iterative methods including coupled three-term recurrences for solving nonsymmetric linear systems', *Int. J. Num. Math. Engin.* To appear.

T. F. Chan and T. Szeto (1994), 'A composite step conjugate gradient squared algorithm for solving nonsymmetric linear systems', *Numerical Algorithms* **7**, 17–32.

T. F. Chan and T. Szeto (1996), 'Composite step product methods for solving nonsymmetric linear systems', *SIAM J. Sci. Comput.* **17**, 1491–1508.

T. F. Chan, L. de Pillis and H. van der Vorst (1991), A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems, Technical Report CAM 91-17, Dept. of Mathematics, University of California, Los Angeles.

T. F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto and C. H. Tong (1994), 'A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems', *SIAM J. Sci. Comput.* **15**, 338–347.

A. T. Chronopoulos and S. Ma (1989), On squaring Krylov subspace iterative methods for nonsymmetric linear systems, Technical Report 89-67, Computer Science Department, University of Minnesota.

J. Cullum (1995), 'Peaks, plateaus, numerical instabilities in a Galerkin/minimal residual pair of methods for solving $Ax = b$', *Appl. Numer. Math.* **19**, 255–278.

J. Cullum and A. Greenbaum (1996), 'Relations between Galerkin and norm-minimizing iterative methods for solving linear systems', *SIAM J. Matrix Anal. Appl.* **17**, 223–247.

J. K. Cullum (1994), Lanczos algorithms for large scale symmetric and nonsymmetric matrix eigenvalue problems, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, PA, pp. 11–31.

J. K. Cullum and R. A. Willoughby (1985), *Lanczos Algorithms for Large Symmetric Eigenvalue Computations (2 Vols.)*, Birkhäuser, Boston-Basel-Stuttgart.

D. Day (1997), 'An efficient implementation of the non-symmetric Lanczos algorithm', *SIAM J. Matrix Anal. Appl.* To appear.

D. M. Day, III (1993), Semi-duality in the two-sided Lanczos algorithm, PhD thesis, University of California at Berkeley.

V. Eijkhout (1994), LAPACK Working Note 78: Computational variants of the CGS and BiCGstab methods, Technical Report UT-CS-94-241, Computer Science Department, University of Tennessee.

S. C. Eisenstat, H. C. Elman and M. H. Schultz (1983), 'Variational iterative methods for nonsymmetric systems of linear equations', *SIAM J. Numer. Anal.* **20**, 345–357.

D. K. Faddeev and V. N. Faddeeva (1964), *Numerische Verfahren der linearen Algebra*, Oldenbourg, München. This is not the same book as *Computational Methods of Linear Algebra*.

K. V. Fernando and B. N. Parlett (1994), 'Accurate singular values and differential qd algorithms', *Numer. Math.* **67**, 191–229.

R. Fletcher (1976), Conjugate gradient methods for indefinite systems, in *Numerical Analysis, Dundee, 1975* (G. A. Watson, ed.), Vol. 506 of *Lecture Notes in Mathematics*, Springer, Berlin, pp. 73–89.

D. R. Fokkema (1996*a*), Enhanced implementation of BiCGstab(ℓ) for solving linear systems of equations, Preprint 976, Department of Mathematics, Utrecht University.

D. R. Fokkema (1996*b*), Subspace Methods for Linear, Nonlinear, and Eigen Problems, PhD thesis, Utrecht University.

D. R. Fokkema, G. L. G. Sleijpen and H. A. van der Vorst (1996), 'Generalized conjugate gradient squared', *J. Comput. Appl. Math.* **71**, 125–146.

R. W. Freund (1992), 'Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices', *SIAM J. Sci. Statist. Comput.* **13**, 425–448.

R. W. Freund (1993), 'A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems', *SIAM J. Sci. Comput.* **14**, 470–482.

R. W. Freund (1994), Lanczos-type algorithms for structured non-Hermitian eigenvalue problems, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, PA, pp. 243–245.

R. W. Freund and M. Malhotra (1997), 'A block-QMR algorithm for non-Hermitian linear systems with multiple right-hand sides', *Linear Algebra Appl.* To appear.

R. W. Freund and N. M. Nachtigal (1991), 'QMR: a quasi-minimal residual method for non-Hermitian linear systems', *Numer. Math.* **60**, 315–339.

R. W. Freund and N. M. Nachtigal (1993), Implementation details of the coupled QMR algorithm, in *Numerical Linear Algebra* (L. Reichel, A. Ruttan and R. S. Varga, eds), W. de Gruyter, pp. 123–140.

R. W. Freund and N. M. Nachtigal (1994), 'An implementation of the QMR method based on coupled two-term recurrences', *SIAM J. Sci. Comput.* **15**, 313–337.

R. W. Freund and N. M. Nachtigal (1996), 'QMRPACK: a package of QMR algorithms', *ACM Trans. Math. Software* **22**, 46–77.

R. W. Freund and T. Szeto (1991), A quasi-minimal residual squared algorithm for non-Hermitian linear systems, Technical Report 91.26, RIACS, NASA Ames Research Center, Moffett Field, CA.

R. W. Freund and H. Zha (1993), 'A look-ahead algorithm for the solution of general Hankel systems', *Numer. Math.* **64**, 295–321.

R. W. Freund, M. H. Gutknecht and N. M. Nachtigal (1993), 'An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices', *SIAM J. Sci. Comput.* **14**, 137–158.

G. Golub and C. van Loan (1989), *Matrix Computations*, 2nd edn, Johns Hopkins University Press, Baltimore, MD.

G. H. Golub and M. H. Gutknecht (1990), 'Modified moments for indefinite weight functions', *Numer. Math.* **57**, 607–624.

G. H. Golub and D. P. O'Leary (1989), 'Some history of the conjugate gradient and Lanczos algorithms: 1949–1976', *SIAM Rev.* **31**, 50–102.

G. Golub, B. Kågström and P. Van Dooren (1992), 'Direct block tridiagonalization of single-input single-output systems', *Systems Control Lett.* **18**, 109–120.

W. B. Gragg (1974), 'Matrix interpretations and applications of the continued fraction algorithm', *Rocky Mountain J. Math.* **4**, 213–225.

W. B. Gragg and A. Lindquist (1983), 'On the partial realization problem', *Linear Algebra Appl.* **50**, 277–319.

A. Greenbaum (1989), 'Predicting the behavior of finite precision Lanczos and conjugate gradient computations', *Linear Algebra Appl.* **113**, 7–63.

A. Greenbaum (1994*a*), Accuracy of computed solutions from conjugate-gradient-like methods, in *Advances in Numerical Methods for Large Sparse Sets of Linear Systems* (M. Natori and T. Nodera, eds), number 10 *in* 'Parallel Processing for Scientific Computing', Keio University, Yokahama, Japan, pp. 126–138.

A. Greenbaum (1994*b*), The Lanczos and conjugate gradient algorithms in finite precision arithmetic, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, PA, pp. 49–60.

A. Greenbaum (1997), 'Estimating the attainable accuracy of recursively computed residual methods', *SIAM J. Matrix Anal. Appl.* To appear.

A. Greenbaum and Z. Strakos (1992), 'Predicting the behavior of finite precision Lanczos and conjugate gradient computations', *SIAM J. Matrix Anal. Appl.* **13**, 121–137.

M. H. Gutknecht (1989*a*), 'Continued fractions associated with the Newton–Padé table', *Numer. Math.* **56**, 547–589.

M. H. Gutknecht (1989*b*), 'Stationary and almost stationary iterative (k,l)-step methods for linear and nonlinear systems of equations', *Numer. Math.* **56**, 179–213.

M. H. Gutknecht (1990), 'The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the qd algorithm', in *Preliminary Proceedings of the Copper Mountain Conference on Iterative Methods, April 1–5, 1990*, `http://www.scsc.ethz.ch/~mgh/pub/CopperMtn90.ps.Z` and `CopperMtn90-7.ps.Z`.

M. H. Gutknecht (1992), 'A completed theory of the unsymmetric Lanczos process and related algorithms, Part I', *SIAM J. Matrix Anal. Appl.* **13**, 594–639.

M. H. Gutknecht (1993*a*), 'Changing the norm in conjugate gradient type algorithms', *SIAM J. Numer. Anal.* **30**, 40–56.

M. H. Gutknecht (1993*b*), 'Stable row recurrences in the Padé table and generically superfast lookahead solvers for non-Hermitian Toeplitz systems', *Linear Algebra Appl.* **188/189**, 351–421.

M. H. Gutknecht (1993*c*), 'Variants of BiCGStab for matrices with complex spectrum', *SIAM J. Sci. Comput.* **14**, 1020–1033.

M. H. Gutknecht (1994*a*), 'A completed theory of the unsymmetric Lanczos process and related algorithms, Part II', *SIAM J. Matrix Anal. Appl.* **15**, 15–58.

M. H. Gutknecht (1994*b*), The Lanczos process and Padé approximation, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, PA, pp. 61–75.

M. H. Gutknecht (1994*c*), 'Local minimum residual smoothing', Talk at Oberwolfach, Germany.

M. H. Gutknecht and W. B. Gragg (1994), Stable look-ahead versions of the Euclidean and Chebyshev algorithms, IPS Research Report 94-04, IPS, ETH Zürich.

M. H. Gutknecht and M. Hochbruck (1995), 'Look-ahead Levinson and Schur algorithms for non-Hermitian Toeplitz systems', *Numer. Math.* **70**, 181–227.

M. H. Gutknecht and K. J. Ressel (1996), Look-ahead procedures for Lanczos-type product methods based on three-term recurrences, Tech. Report TR-96-19, Swiss Center for Scientific Computing.

M. H. Gutknecht and K. J. Ressel (1997), Attempts to enhance the BiCGStab family, Technical report, Swiss Center for Scientific Computing. In preparation.

M. Hanke (1997), 'Superlinear convergence rates for the Lanczos method applied to elliptic operators', *Numer. Math.* To appear.

R. M. Hayes (1954), Iterative methods of solving linear problems on Hilbert space, in *Contributions to the Solution of Simultaneous Linear Equations and the Determination of Eigenvalues* (O. Taussky, ed.), Vol. 49 of *Applied Mathematics Series*, National Bureau of Standards, pp. 71–103.

P. Henrici (1974), *Applied and Computational Complex Analysis, Vol. 1*, Wiley, New York.

M. R. Hestenes (1951), Iterative methods for solving linear equations, NAML Report 52-9, National Bureau of Standards, Los Angeles, CA. Reprinted in *J. Optim. Theory Appl.* 11, 323–334 (1973).

M. R. Hestenes (1980), *Conjugate Direction Methods in Optimization*, Springer, Berlin.

M. R. Hestenes and E. Stiefel (1952), 'Methods of conjugate gradients for solving linear systems', *J. Res. Nat. Bur. Standards* **49**, 409–435.

M. Hochbruck (1992), Lanczos- und Krylov-Verfahren für nicht-Hermitesche lineare Systeme, PhD thesis, Fakultät für Mathematik, Universität Karlsruhe.

M. Hochbruck (1996), 'The Padé table and its relation to certain numerical algorithms', Habilitationsschrift, Universität Tübingen, Germany.

M. Hochbruck and C. Lubich (1997*a*), 'Error analysis of Krylov methods in a nutshell', *SIAM J. Sci. Comput.* To appear.

M. Hochbruck and C. Lubich (1997*b*), 'On Krylov subspace approximations to the matrix exponential operator', *SIAM J. Numer. Anal.* To appear.

R. A. Horn and C. R. Johnson (1985), *Matrix Analysis*, Cambridge University Press.

A. S. Householder (1964), *The Theory of Matrices in Numerical Analysis*, Dover, New York.

K. C. Jea and D. M. Young (1983), 'On the simplification of generalized conjugate-gradient methods for nonsymmetrizable linear systems', *Linear Algebra Appl.* **52**, 399–417.

W. D. Joubert (1990), Generalized conjugate gradient and Lanczos methods for the solution of nonsymmetric systems of linear equations, PhD thesis, Center for Numerical Analysis, University of Texas at Austin. Tech. Rep. CNA-238.

W. D. Joubert (1992), 'Lanczos methods for the solution of nonsymmetric systems of linear equations', *SIAM J. Matrix Anal. Appl.* **13**, 926–943.

S. Kaniel (1966), 'Estimates for some computational techniques in linear algebra', *Math. Comp.* **20**, 369–378.

M. Khelifi (1991), 'Lanczos maximal algorithm for unsymmetric eigenvalue problems', *Appl. Numer. Math.* **7**, 179–193.

K. Kreuzer, H. Miller and W. Berger (1981), 'The Lanczos algorithm for self-adjoint operators', *Physics Letters* **81A**, 429–432.

E. Kreyszig (1978), *Introductory Functional Analysis with Applications*, Wiley.

C. Lanczos (1950), 'An iteration method for the solution of the eigenvalue problem of linear differential and integral operators', *J. Res. Nat. Bur. Standards* **45**, 255–281.

C. Lanczos (1952), 'Solution of systems of linear equations by minimized iterations', *J. Res. Nat. Bur. Standards* **49**, 33–53.

D. G. Luenberger (1979), 'Hyperbolic pairs in the method of conjugate gradients', *SIAM J. Appl. Math.* **17**, 1263–1267.

T. A. Manteuffel (1977), 'The Tchebyshev iteration for nonsymmetric linear systems', *Numer. Math.* **28**, 307–327.

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

J. A. Meijerink and H. A. van der Vorst (1981), 'Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems', *J. Comput. Phys.* **44**, 134–155.

W. Murray, ed. (1972), *Numerical Methods for Unconstrained Optimization*, Academic, London.

N. M. Nachtigal (1991), A look-ahead variant of the Lanczos algorithm and its application to the quasi-minimal residual method for non-Hermitian linear systems, PhD thesis, Department of Mathematics, MIT.

A. Neumaier (1994), 'Iterative regularization for large-scale ill-conditioned linear systems', Talk at Oberwolfach.

O. Nevanlinna (1993), *Convergence of Iterations for Linear Equations*, Birkhäuser, Basel.

C. C. Paige (1971), The computations of eigenvalues and eigenvectors of very large sparse matrices, PhD thesis, University of London.

C. C. Paige (1976), 'Error analysis of the Lanczos algorithms for tridiagonalizing a symmetric matrix', *J. Inst. Math. Appl.* **18**, 341–349.

C. C. Paige (1980), 'Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem', *Linear Algebra Appl.* **34**, 235–258.

C. C. Paige and M. A. Saunders (1975), 'Solution of sparse indefinite systems of linear equations', *SIAM J. Numer. Anal.* **12**, 617–629.

B. N. Parlett (1980), *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ.

B. N. Parlett (1992), 'Reduction to tridiagonal form and minimal realizations', *SIAM J. Matrix Anal. Appl.* **13**, 567–593.

B. N. Parlett (1994), Do we fully understand the symmetric Lanczos algorithm yet?, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, PA, pp. 93–107.

B. N. Parlett (1995), 'The new qd algorithms', in *Acta Numerica*, Vol. 4, Cambridge University Press, pp. 459–491.

B. N. Parlett and B. Nour-Omid (1989), 'Towards a black box Lanczos program', *Computer Physics Comm.* **53**, 169–179.

B. N. Parlett and J. K. Reid (1981), 'Tracking the progress of the Lanczos algorithm for large symmetric matrices', *IMA J. Numer. Anal.* **1**, 135–155.

B. N. Parlett and D. S. Scott (1979), 'The Lanczos algorithm with selective reorthogonalization', *Math. Comp.* **33**, 217–238.

B. N. Parlett, D. R. Taylor and Z. A. Liu (1985), 'A look-ahead Lanczos algorithm for unsymmetric matrices', *Math. Comp.* **44**, 105–124.

K. J. Ressel and M. H. Gutknecht (1996), QMR-smoothing for Lanczos-type product methods based on three-term recurrences, Tech. Report TR-96-18, Swiss Center for Scientific Computing.

W. Rudin (1973), *Functional Analysis*, McGraw-Hill, New York.

A. Ruhe (1979), 'Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices', *Math. Comp.* **33**, 680–687.

H. Rutishauser (1953), 'Beiträge zur Kenntnis des Biorthogonalisierungs-Algorithmus von Lanczos', *Z. Angew. Math. Phys.* **4**, 35–56.

H. Rutishauser (1957), *Der Quotienten-Differenzen-Algorithmus*, Mitt. Inst. angew. Math. ETH, Nr. 7, Birkhäuser, Basel.

H. Rutishauser (1990), *Lectures on Numerical Mathematics*, Birkhäuser, Boston.

Y. Saad (1980), 'Variations on Arnoldi's method for computing eigenelements of large unsymmetric systems', *Linear Algebra Appl.* **34**, 269–295.

Y. Saad (1981), 'Krylov subspace methods for solving large unsymmetric systems', *Math. Comp.* **37**, 105–126.

Y. Saad (1982), 'The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems', *SIAM J. Numer. Anal.* **2**, 485–506.

Y. Saad (1994), Theoretical error bounds and general analysis of a few Lanczos-type algorithms, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, PA, pp. 123–134.

Y. Saad (1996), *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston.

Y. Saad and M. H. Schultz (1986), 'GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Statist. Comput.* **7**, 856–869.

W. Schönauer (1987), *Scientific Computing on Vector Computers*, Elsevier, Amsterdam.

H. D. Simon (1984*a*), 'Analysis of the symmetric Lanczos algorithm with reorthogonalization methods', *Linear Algebra Appl.* **61**, 101–131.

H. D. Simon (1984*b*), 'The Lanczos algorithm with partial reorthogonalization', *Math. Comp.* **42**, 115–142.

G. L. G. Sleijpen and D. R. Fokkema (1993), 'BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum', *Electronic Trans. Numer. Anal.* **1**, 11–32.

G. L. G. Sleijpen and H. A. van der Vorst (1995*a*), 'Maintaining convergence properties of bicgstab methods in finite precision arithmetic', *Numerical Algorithms* **10**, 203–223.

G. L. G. Sleijpen and H. A. van der Vorst (1995*b*), 'An overview of approaches for the stable computation of hybrid BiCG methods', *Appl. Numer. Math.* **19**, 235–254.

G. L. G. Sleijpen and H. A. van der Vorst (1996), 'Reliable updated residuals in hybrid Bi-CG methods', *Computing* **56**, 141–163.

G. L. G. Sleijpen, H. A. van der Vorst and D. R. Fokkema (1994), 'BiCGstab(l) and other hybrid Bi-CG methods', *Numerical Algorithms* **7**, 75–109.

P. Sonneveld (1989), 'CGS, a fast Lanczos-type solver for nonsymmetric linear systems', *SIAM J. Sci. Statist. Comput.* **10**, 36–52.

G. W. Stewart (1994), Lanczos and linear systems, in *Proceedings of the Cornelius Lanczos International Centenary Conference* (J. D. Brown, M. T. Chu, D. C. Ellison and R. J. Plemmons, eds), SIAM, Philadelphia, PA, pp. 135–139.

T. J. Stieltjes (1884), 'Quelques recherches sur la théorie des quadratures dites mécaniques', *Ann. Sci. École Norm. Paris Sér. 3* **1**, 409–426. [*Oeuvres*, vol. 1, pp. 377–396].

Z. Strakoš (1991), 'On the real convergence rate of the conjugate gradient method', *Linear Algebra Appl.* **154–156**, 535–549.

D. R. Taylor (1982), Analysis of the Look Ahead Lanczos Algorithm, PhD thesis, Dept. of Mathematics, University of California, Berkeley.

C. H. Tong (1994), 'A family of quasi-minimal residual methods for nonsymmetric linear systems', *SIAM J. Sci. Comput.* **15**, 89–105.

C. H. Tong and Q. Ye (1995), 'Analysis of the finite precision bi-conjugate gradient algorithm for nonsymmetric linear systems'. Preprint.

C. H. Tong and Q. Ye (1996), 'A linear system solver based on a modified Krylov subspace method for breakdown recovery', *Numerical Algorithms* **12**, 233–251.

A. van der Sluis (1992), The convergence behaviour of conjugate gradients and Ritz values in various circumstances, in *Iterative Methods in Linear Algebra* (R. Beauwens and P. de Groen, eds), Elsevier (North-Holland), Proceedings IMACS Symposium, Brussels, 1991, pp. 49–66.

A. van der Sluis and H. A. van der Vorst (1986), 'The rate of convergence of conjugate gradients', *Numer. Math.* **48**, 543–560.

A. van der Sluis and H. A. van der Vorst (1987), 'The convergence behavior of Ritz values in the presence of close eigenvalues', *Linear Algebra Appl.* **88/89**, 651–694.

H. A. van der Vorst (1992), 'Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems', *SIAM J. Sci. Statist. Comput.* **13**, 631–644.

H. A. van der Vorst and C. Vuik (1993), 'The superlinear convergence behaviour of GMRES', *J. Comput. Appl. Math.* **48**, 327–341.

H. F. Walker (1988), 'Implementation of the GMRES method using Householder transformations', *SIAM J. Sci. Statist. Comput.* **9**, 152–163.

H. F. Walker (1995), 'Residual smoothing and peak/plateau behavior in Krylov subspace methods', *Appl. Numer. Math.* **19**, 279–286.

R. Weiss (1990), Convergence behavior of generalized conjugate gradient methods, PhD thesis, University of Karlsruhe.

R. Weiss (1994), 'Properties of generalized conjugate gradient methods', *J. Numer. Linear Algebra Appl.* **1**, 45–63.

J. H. Wilkinson (1965), *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford.

Q. Ye (1991), 'A convergence analysis for nonsymmetric Lanczos algorithms', *Math. Comp.* **56**, 677–691.

Q. Ye (1994), 'A breakdown-free variation of the nonsymmetric Lanczos algorithms', *Math. Comp.* **62**, 179–207.

S.-L. Zhang (1997), 'GPBI-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems', *SIAM J. Sci. Comput.* **18**, 537–551.

L. Zhou and H. F. Walker (1994), 'Residual smoothing techniques for iterative methods', *SIAM J. Sci. Comput.* **15**, 297–312.