ELSEVIER

Applied Numerical Mathematics 41 (2002) 119–133

APPLIED NUMERICAL MATHEMATICS

www.elsevier.com/locate/apnum

Variations of Zhang's Lanczos-type product method

Stefan Röllin [a,*], Martin H. Gutknecht [b]

[a] *Integrated Systems Laboratory, ETH-Zentrum ETZ, CH-8092 Zürich, Switzerland*
[b] *Seminar for Applied Mathematics, ETH-Zentrum HG, CH-8092 Zürich, Switzerland*

Dedicated to the memory of Rüdiger Weiss

Abstract

Among the Lanczos-type product methods, which are characterized by residual polynomials $p_n t_n$ that are the product of the Lanczos polynomial p_n and another polynomial t_n of exact degree n with $t_n(0) = 1$, Zhang's algorithm GPBiCG has the feature that the polynomials t_n are implicitly built up by a pair of coupled two-term recurrences whose coefficients are chosen so that the new residual is minimized in a 2-dimensional space. There are several ways to achieve this. We discuss here alternative algorithms that are mathematically equivalent (that is, produce in exact arithmetic the same results). The goal is to find one where the ultimate accuracy of the iterates $\boldsymbol{x}_n$ is guaranteed to be high and the cost is at most slightly increased. © 2001 IMACS. Published by Elsevier Science B.V. All rights reserved.

Keywords: Krylov space method; Biconjugate gradients; Lanczos-type product method; BiCG × MR2; GPBi-CG

1. Introduction

Krylov space methods based on the Lanczos process are particularly efficient tools for solving large sparse non-symmetric systems of linear equations. In contrast to competing methods based on orthogonal or minimal residual, they feature short recurrences (that is, three-term or coupled two-term recurrences) for generating the approximations (or iterates) $\boldsymbol{x}_n$ and the corresponding residuals $\boldsymbol{r}_n :\equiv \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n$. However, the classical biconjugate gradient (BiCG) method of Lanczos [9] (reformulated by Fletcher [2]) has also a number of shortcomings:

(i) BiCG may break down (even for well-conditioned problems),
(ii) BiCG requires matrix–vector products with the transpose of the matrix,
(iii) BiCG needs two matrix–vector products to gain one dimension in the search space,
(iv) the convergence often appears to be very erratic,

[*] Corresponding author.
E-mail addresses: roellin@iis.ee.ethz.ch (S. Röllin), mhg@sam.math.ethz.ch (M.H. Gutknecht).

(v) roundoff causes loss of biorthogonality, inaccurate recurrence coefficients (and, hence, inaccurate eigenvalue approximations), and, possibly, low ultimate accuracy of the approximate solution of the linear system.

For all these shortcomings, there are at least partial remedies. Breakdowns can be overcome by *look-ahead*; see [7] and references given there. The transpose and the second matrix–vector product can be avoided by "squaring" BiCG, as suggested by Sonneveld with his (bi)conjugate gradient squared (CGS) algorithm [12]. This algorithm still needs two matrix–vector products per iteration step but we gain two dimensions in the search space, in contrast to BiCG. These benefits of CGS persist in Van der Vorst's BiCGStab [13], which additionally smoothes the very erratic convergence behavior of CGS somewhat and has become the model for a whole family of *Lanczos-type product methods* (*LTPMs*). The smoothing effect results from an incorporated one-dimensional local residual minimization.

This paper is devoted to a set of algorithms that realize a particular LTPM first proposed in 1993 by Zhang [15], where a 2-dimensional minimization is incorporated in each step, and which was therefore called BiCG×MR2 in [7]. In one version, which independently was also proposed by Cao [1] and Gutknecht [6], the implementation only requires a one-line modification of BiCGStab2 [5], which does such a minimization in every other step. But Zhang also proposed a version called GPBiCG that is fully based on coupled two-term recursions and can therefore be expected to have better roundoff behavior. It was recently shown in [8] that under certain assumptions Krylov solvers based on a pair of three-term recurrences attain typically a lower ultimate accuracy than those based on two-term recurrences. However, Zhang's algorithm GPBiCG is more complicated and does not fit into the simple patterns compared in [8]. Our aim was to find a version that is easier to analyze and can be shown to have high ultimate accuracy at the same cost as GPBiCG.

2. Lanczos-type product methods based on two pairs of coupled two-term recurrences

Starting point for all LTPMs is the classical method BiCG of Lanczos [9] reformulated by Fletcher [2]. It is given as Algorithm 1.

Algorithm 1 (BiCG). *For computing $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ choose an initial approximation $\boldsymbol{x}_0$ and let $\boldsymbol{v}_0 := \boldsymbol{r}_0 := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$. Choose $\widetilde{\boldsymbol{r}}_0$ such that $\delta_0 := \langle \widetilde{\boldsymbol{r}}_0, \boldsymbol{r}_0\rangle \neq 0$ and $\delta_0' := \langle \widetilde{\boldsymbol{r}}_0, \boldsymbol{A}\boldsymbol{v}_0\rangle \neq 0$. Then, compute for $n = 0, 1, \ldots$*

$$\begin{aligned}
\omega_n &:= \delta_n/\delta_n', \\
\boldsymbol{x}_{n+1} &:= \boldsymbol{x}_n + \boldsymbol{v}_n\omega_n, \\
\boldsymbol{r}_{n+1} &:= \boldsymbol{r}_n - \boldsymbol{A}\boldsymbol{v}_n\omega_n, \\
\widetilde{\boldsymbol{r}}_{n+1} &:= \widetilde{\boldsymbol{r}}_n - \boldsymbol{A}^\star\widetilde{\boldsymbol{v}}_n\omega_n, \\
\delta_{n+1} &:= \langle \widetilde{\boldsymbol{r}}_{n+1}, \boldsymbol{r}_{n+1}\rangle, \\
\psi_n &:= -\delta_{n+1}/\delta_n, \\
\boldsymbol{v}_{n+1} &:= \boldsymbol{r}_{n+1} - \boldsymbol{v}_n\psi_n, \\
\widetilde{\boldsymbol{v}}_{n+1} &:= \widetilde{\boldsymbol{r}}_{n+1} - \widetilde{\boldsymbol{v}}_n\psi_n, \\
\widetilde{\delta}_n' &:= \langle \widetilde{\boldsymbol{v}}_{n+1}, \boldsymbol{A}\boldsymbol{v}_{n+1}\rangle.
\end{aligned}$$

The basic idea behind BiCG is to generate residuals $\boldsymbol{r}_n$ and a second set of vectors $\widetilde{\boldsymbol{r}}_n$ with the properties

$$\boldsymbol{r}_n \in \mathcal{K}_n \equiv \mathcal{K}_n(\boldsymbol{A}, \boldsymbol{r}_0) := \operatorname{span}\bigl(\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{r}_0\bigr), \tag{1}$$

$$\widetilde{\boldsymbol{r}}_n \in \widetilde{\mathcal{K}}_n \equiv \widetilde{\mathcal{K}}_n(\boldsymbol{A}^\star, \widetilde{\boldsymbol{r}}_0) := \operatorname{span}\bigl(\widetilde{\boldsymbol{r}}_0, \boldsymbol{A}^\star\widetilde{\boldsymbol{r}}_0, \ldots, (\boldsymbol{A}^\star)^{n-1}\widetilde{\boldsymbol{r}}_0\bigr), \tag{2}$$

and

$$\boldsymbol{r}_n \perp \widetilde{\mathcal{K}}_n, \qquad \widetilde{\boldsymbol{r}}_n \perp \mathcal{K}_n. \tag{3}$$

In other words, we build two sequences of biorthogonal vectors $\boldsymbol{r}_n$ and $\widetilde{\boldsymbol{r}}_n$, which lie in the Krylov spaces $\mathcal{K}_n$ and $\widetilde{\mathcal{K}}_n$, respectively. Analogous properties hold for the direction vectors $\boldsymbol{v}_n$ and the vectors $\widetilde{\boldsymbol{v}}_n$. They are not biorthogonal but biconjugate, which means that they satisfy

$$\boldsymbol{v}_n \in \mathcal{K}_n, \qquad \widetilde{\boldsymbol{v}}_n \in \widetilde{\mathcal{K}}_n, \tag{4}$$

and

$$\boldsymbol{v}_n \perp \boldsymbol{A}^\star\widetilde{\mathcal{K}}_n, \qquad \widetilde{\boldsymbol{v}}_n \perp \boldsymbol{A}\mathcal{K}_n. \tag{5}$$

Since the residuals $\boldsymbol{r}_n$ and the direction vectors $\boldsymbol{v}_n$ lie in the Krylov space $\mathcal{K}_n$, they can be expressed as

$$\boldsymbol{r}_n = p_n(\boldsymbol{A})\boldsymbol{r}_0, \tag{6a}$$

$$\boldsymbol{v}_n = \widehat{p}_n(\boldsymbol{A})\boldsymbol{r}_0, \tag{6b}$$

where $p_n(\zeta)$ and $\widehat{p}_n(\zeta)$ are polynomials of degree n. From Algorithm 1 (BiCG) and (6) we can derive two coupled recurrences for these two polynomials:

$$p_{n+1}(\zeta) := p_n(\zeta) - \omega_n\zeta\,\widehat{p}_n(\zeta), \tag{7a}$$

$$\widehat{p}_{n+1}(\zeta) := p_{n+1}(\zeta) - \psi_n\widehat{p}_n(\zeta) \tag{7b}$$

(to be started with $p_0(\zeta) = 1$, $\widehat{p}_0(\zeta) = 1$).

In LTPMs the idea is to use $t_n(\boldsymbol{A})p_n(\boldsymbol{A})\boldsymbol{r}_0$ as the nth residual of the method. The polynomials $p_n(\zeta)$ are those of BiCG as defined above, and the polynomials $t_l(\zeta)$ are arbitrary polynomials of respective exact degree l satisfying the consistency condition $t_l(0) = 1$. There are various strategies to choose $t_l(\zeta)$. For example, setting $t_l(\zeta) = p_l(\zeta)$ gives the CGS method of Sonneveld [12]. In van der Vorst's BiCGStab [13] the polynomials $t_l(\zeta)$ are defined by the recurrence

$$t_{l+1}(\zeta) := (1 - \chi_{l+1}\zeta)t_l(\zeta), \tag{8}$$

where χ_{l+1} is chosen such that $\|\boldsymbol{r}_{l+1}\| = \|t_{l+1}(\boldsymbol{A})p_{l+1}(\boldsymbol{A})\boldsymbol{r}_0\|$ is minimized. Note that in the case of real data the polynomial t_l is real-valued and therefore all its zeros $1/\chi_k$ $(k = 1, \ldots, l)$ are real, although the spectrum of $\boldsymbol{A}$ may be complex.

In contrast, in BiCGStab2 [5] in every other step two new zeros are chosen by a two-dimensional residual minimization rule, and these zeros can be complex-conjugate. The recurrences, which impose automatically the normalization $t_{l+1}(0) = 1$, are

$$t_{l+1}(\zeta) := (1 - \chi_{l+1}\zeta)t_l(\zeta), \quad \text{if } l \text{ is even}, \tag{9}$$

$$t_{l+1}(\zeta) := (\xi_l + \eta_l\zeta)t_l(\zeta) + (1 - \xi_l)t_{l-1}(\zeta), \quad \text{if } l \text{ is odd}. \tag{10}$$

Here, χ_{l+1} is again chosen to minimize $\|\boldsymbol{r}_{l+1}\|$ when l is even, but its choice has at least in theory no consequence on $\boldsymbol{r}_n$ for $n > l+1$ as long as $\chi_{l+1} \neq 0$. (If $|\chi_{l+1}| \approx 0$, serious roundoff effects have to be expected, however.) The pair (ξ_l, η_l) is chosen to minimize $\|\boldsymbol{r}_{l+1}\|$ if l is odd, and since we have now two coefficients that appear linearly in the recurrence, this leads to a least squares problem with a linear 2-by-2 system of normal equations; see [5].

Alternatively, we could also use (10) for every step and enforce a two-dimensional minimization in every step; see [6,7,15]. However, in this paper we focus on a different implementation of this goal and, following [15], we assume that also the polynomials $t_l(\zeta)$ satisfy instead of the three-term recurrence (10) an equivalent pair of coupled two-term recurrences,

$$t_{l+1}(\zeta) := t_l(\zeta) - \widetilde{\omega}_l \zeta \widehat{t}_l(\zeta), \tag{11a}$$

$$\widehat{t}_{l+1}(\zeta) := t_{l+1}(\zeta) - \widetilde{\psi}_l \widehat{t}_l(\zeta) \tag{11b}$$

with $t_0(\zeta) = 1$, $\widehat{t}_0(\zeta) = 1$. Obviously these recurrences are fully analogous to those in (7) for the Lanczos polynomials p_n and the corresponding direction polynomials $\widehat{p}_n$, and they therefore involve a fourth set of polynomials $\widehat{t}_l$. The coefficients $\widetilde{\omega}_l$ and $\widetilde{\psi}_l$ will be chosen such that in each step the 2-norm of the residual is minimized in a 2-dimensional space.

For the residuals $t_n(\boldsymbol{A})p_n(\boldsymbol{A})\boldsymbol{r}_0$ and some other vectors that will be used as intermediate quantities the following notation is introduced:

$$\boldsymbol{w}_n^l :\equiv t_l(\boldsymbol{A})p_n(\boldsymbol{A})\boldsymbol{r}_0, \tag{12a}$$

$$\widehat{\boldsymbol{w}}_n^l :\equiv t_l(\boldsymbol{A})\widehat{p}_n(\boldsymbol{A})\boldsymbol{r}_0, \tag{12b}$$

$$\boldsymbol{u}_n^l :\equiv \widehat{t}_l(\boldsymbol{A})p_n(\boldsymbol{A})\boldsymbol{r}_0, \tag{12c}$$

$$\widehat{\boldsymbol{u}}_n^l :\equiv \widehat{t}_l(\boldsymbol{A})\widehat{p}_n(\boldsymbol{A})\boldsymbol{r}_0. \tag{12d}$$

Although these vectors are here defined for all nonnegative indices l and n, only some where $n - l$ is small will actually be needed and computed; in particular, of course, the residuals $\boldsymbol{w}_n^n$ of the method. The challenge is to find an efficient and stable way to get from $\boldsymbol{w}_n^n$ to the next residual $\boldsymbol{w}_{n+1}^{n+1}$. At the same time, the Lanczos recurrence coefficients ω_n, ψ_n and the coefficients $\widetilde{\omega}_n$, $\widetilde{\psi}_n$ have to be computed. For the moment, we postpone the question of how to compute the corresponding iterates $\boldsymbol{x}_n^n$ and $\boldsymbol{x}_{n+1}^{n+1}$.

By multiplying Eqs. (7) with $t_l(\zeta)$ and $\widehat{t}_l(\zeta)$ as well as multiplying (11) with $p_n(\zeta)$ and $\widehat{p}_n(\zeta)$, and by recalling that a factor ζ in (7a) or (11a) gives rise to a factor $\boldsymbol{A}$ when we map polynomials in Krylov space vectors, we obtain in the notation (12)

$$\boldsymbol{w}_{n+1}^l = \boldsymbol{w}_n^l - \boldsymbol{A}\widehat{\boldsymbol{w}}_n^l \omega_n, \tag{13a}$$

$$\boldsymbol{u}_{n+1}^l = \boldsymbol{u}_n^l - \boldsymbol{A}\widehat{\boldsymbol{u}}_n^l \omega_n, \tag{13b}$$

$$\widehat{\boldsymbol{w}}_{n+1}^l = \boldsymbol{w}_{n+1}^l - \widehat{\boldsymbol{w}}_n^l \psi_n, \tag{13c}$$

$$\widehat{\boldsymbol{u}}_{n+1}^l = \boldsymbol{u}_{n+1}^l - \widehat{\boldsymbol{u}}_n^l \psi_n, \tag{13d}$$

$$\boldsymbol{w}_n^{l+1} = \boldsymbol{w}_n^l - \boldsymbol{A}\boldsymbol{u}_n^l \widetilde{\omega}_l, \tag{13e}$$

$$\widehat{\boldsymbol{w}}_n^{l+1} = \widehat{\boldsymbol{w}}_n^l - \boldsymbol{A}\widehat{\boldsymbol{u}}_n^l \widetilde{\omega}_l, \tag{13f}$$

$$\boldsymbol{u}_n^{l+1} = \boldsymbol{w}_n^{l+1} - \boldsymbol{u}_n^l \widetilde{\psi}_l, \tag{13g}$$

$$\widehat{\boldsymbol{u}}_n^{l+1} = \widehat{\boldsymbol{w}}_n^{l+1} - \widehat{\boldsymbol{u}}_n^l \widetilde{\psi}_l. \tag{13h}$$

These eight equations show the relations between the vectors $\boldsymbol{w}_n^l$, $\widehat{\boldsymbol{w}}_n^l$, $\boldsymbol{u}_n^l$ and $\widehat{\boldsymbol{u}}_n^l$. For a visualization of these relations we arrange the vectors in a table. Vectors with the same indices are grouped in a block. We let the l-axis point to the right and the n-axis point downwards. The result is shown in Fig. 1. As mentioned above, $\boldsymbol{w}_n^n$ will be the nth residual and therefore the vectors in the upper left corner of the diagonal blocks are the vectors we really aim at. Comparing Eqs. (13) with Fig. 1, we see that three consecutive vectors in a row or a column are related. Now, suitable relations can easily be found to compute the vectors of a diagonal block, provided the vectors from the previous diagonal block are known. One possibility is the following:

$$\widehat{\boldsymbol{w}}_n^{n+1} := \widehat{\boldsymbol{w}}_n^n - \boldsymbol{A}\widehat{\boldsymbol{u}}_n^n\widetilde{\omega}_n, \tag{14a}$$

$$\boldsymbol{u}_{n+1}^n := \boldsymbol{u}_n^n - \boldsymbol{A}\widehat{\boldsymbol{u}}_n^n\omega_n, \tag{14b}$$

$$\boldsymbol{w}_{n+1}^{n+1} := \boldsymbol{w}_n^n - \boldsymbol{A}\big(\widehat{\boldsymbol{w}}_n^n\omega_n + \boldsymbol{u}_{n+1}^n\widetilde{\omega}_n\big), \tag{14c}$$

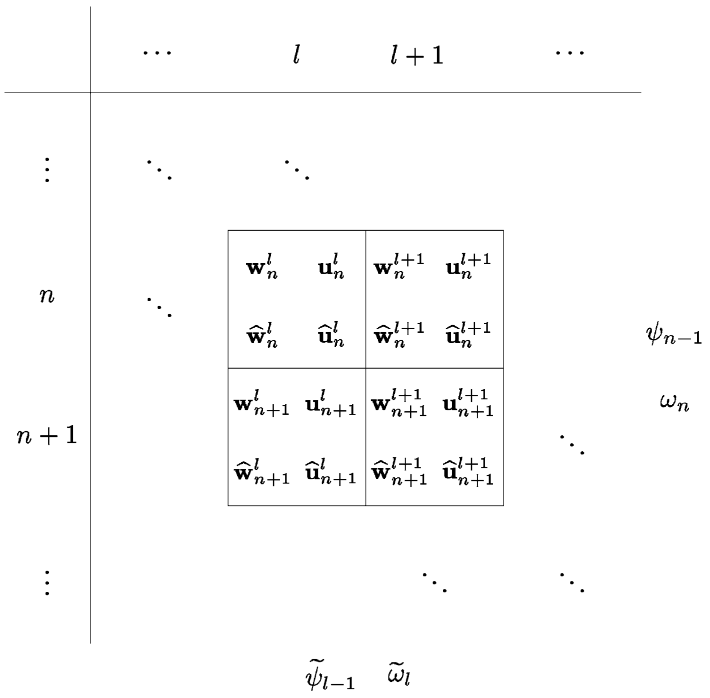
$$\boldsymbol{u}_{n+1}^{n+1} := \boldsymbol{w}_{n+1}^{n+1} - \boldsymbol{u}_{n+1}^n\widetilde{\psi}_n, \tag{14d}$$

$$\widehat{\boldsymbol{u}}_{n+1}^n := \boldsymbol{u}_{n+1}^n - \widehat{\boldsymbol{u}}_n^n\psi_n, \tag{14e}$$

$$\widehat{\boldsymbol{w}}_{n+1}^{n+1} := \boldsymbol{w}_{n+1}^{n+1} - \widehat{\boldsymbol{w}}_n^{n+1}\psi_n, \tag{14f}$$

$$\widehat{\boldsymbol{u}}_{n+1}^{n+1} := \widehat{\boldsymbol{w}}_{n+1}^{n+1} - \widehat{\boldsymbol{u}}_{n+1}^n\widetilde{\psi}_n, \tag{14g}$$

where we have rearranged the relations (13) suitably.



$\cdots$ l $l+1$ $\cdots$
n
$n+1$
$\mathbf{w}_n^l$ $\mathbf{u}_n^l$ $\mathbf{w}_n^{l+1}$ $\mathbf{u}_n^{l+1}$
$\widehat{\mathbf{w}}_n^l$ $\widehat{\mathbf{u}}_n^l$ $\widehat{\mathbf{w}}_n^{l+1}$ $\widehat{\mathbf{u}}_n^{l+1}$
$\mathbf{w}_{n+1}^l$ $\mathbf{u}_{n+1}^l$ $\mathbf{w}_{n+1}^{l+1}$ $\mathbf{u}_{n+1}^{l+1}$
$\widehat{\mathbf{w}}_{n+1}^l$ $\widehat{\mathbf{u}}_{n+1}^l$ $\widehat{\mathbf{w}}_{n+1}^{l+1}$ $\widehat{\mathbf{u}}_{n+1}^{l+1}$
ψ_{n-1}
ω_n
$\widetilde{\psi}_{l-1}$ $\widetilde{\omega}_l$

Fig. 1. The table of an LTPM based on two coupled two-term recurrences.

Determination of ω_n and ψ_n in LTPMs

Up to now, we saw neither how to compute the coefficients of the BiCG polynomial $p_n(\zeta)$ nor the coefficients of the arbitrary polynomial $t_n(\zeta)$.

The BiCG residual $\boldsymbol{r}_n = p_n(\boldsymbol{A})\boldsymbol{r}_0$ fulfills $\boldsymbol{r}_n \perp \widetilde{\mathcal{K}}_n$, where

$$\widetilde{\mathcal{K}}_n \equiv \widetilde{\mathcal{K}}_n(\boldsymbol{A}^\star, \widetilde{\boldsymbol{y}}_0) := \operatorname{span}(\widetilde{\boldsymbol{y}}_0, \boldsymbol{A}^\star\widetilde{\boldsymbol{y}}_0, \dots, (\boldsymbol{A}^\star)^{n-1}\widetilde{\boldsymbol{y}}_0), \tag{15}$$

and $\widetilde{\boldsymbol{y}}_0$ is an arbitrary vector such that $\langle \boldsymbol{r}_0, \widetilde{\boldsymbol{y}}_0\rangle \neq 0$. Especially, for $l < n$, $\overline{t_l}(\boldsymbol{A}^\star)\widetilde{\boldsymbol{y}}_0$ is an element of $\widetilde{\mathcal{K}}_n$ and therefore orthogonal to $\boldsymbol{r}_n$:

$$0 = \langle \overline{t_l}(\boldsymbol{A}^\star)\widetilde{\boldsymbol{y}}_0, \boldsymbol{r}_n\rangle = \langle \widetilde{\boldsymbol{y}}_0, t_l(\boldsymbol{A})p_n(\boldsymbol{A})\boldsymbol{r}_0\rangle = \langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{w}_n^l\rangle, \quad l < n. \tag{16a}$$

For the direction vectors $\boldsymbol{v}_n = \widehat{p}_n(\boldsymbol{A})\boldsymbol{r}_0$ of BiCG the orthogonality $\boldsymbol{A}\boldsymbol{v}_n \perp \widetilde{\mathcal{K}}_n$ holds, which implies that

$$0 = \langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{A}\widehat{\boldsymbol{w}}_n^l\rangle, \quad l < n. \tag{16b}$$

Replacing $t_l(\zeta)$ by $\widehat{t_l}(\zeta)$ in these derivations gives two additional orthogonality conditions:

$$\langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{u}_n^l\rangle = 0, \quad \langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{A}\widehat{\boldsymbol{u}}_n^l\rangle = 0, \qquad l < n. \tag{16c}$$

Taking the orthogonality conditions (16) into account in the recursions (13) for $l = n$ and defining

$$\widetilde{\delta}_n :\equiv \langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{w}_n^n\rangle, \qquad \widetilde{\delta}'_n :\equiv \langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{A}\widehat{\boldsymbol{w}}_n^n\rangle \tag{17a}$$

leads to

$$\omega_n := \frac{\widetilde{\delta}_n}{\widetilde{\delta}'_n}, \qquad \psi_n := -\frac{\widetilde{\delta}_{n+1}}{\widetilde{\delta}'_n\widetilde{\omega}_n}. \tag{17b}$$

For the last equation we used

$$\langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{A}\boldsymbol{w}_{n+1}^n\rangle = -\frac{1}{\widetilde{\omega}_n}\langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{w}_{n+1}^{n+1}\rangle, \tag{18}$$

which can be derived from the conditions (16) and equations (13). As we see from (17), we can obtain the Lanczos coefficients ω_n and ψ_n by computing just two inner products.

Except for the way of computing $\widehat{\boldsymbol{u}}_{n+1}^{n+1}$ the assignments (14) and (17) correspond to those of the general CGS (GCGS) algorithm of Fokkema et al. [3]. Several particular algorithms can be deduced from them. If we let $t_n(\zeta) = p_n(\zeta)$, Sonneveld's CGS is retrieved. The choice $t_n(\zeta) = (1 - \mu\zeta)p_{n-1}(\zeta)$ yields SHIFTED CGS of [3], where simply $\widetilde{\omega}_n := \omega_{n-1}$ and $\widetilde{\psi}_n := \psi_{n-1}$. However, if the aim is to capitalize on the free parameters $\widetilde{\omega}_n$ and $\widetilde{\psi}_n$ for a local 2-dimensional residual norm minimization as in BiCGSTAB2 and GPBiCG, the recursions (14) need to be modified.

Determination of $\widetilde{\omega}_n$ and $\widetilde{\psi}_{n-1}$ in BiCG×MR2

Given $\widehat{t}_{n-1}$ and t_n, we need according to the coupled recurrences (7) both $\widetilde{\psi}_{n-1}$ and $\widetilde{\omega}_n$ in order to compute $\widehat{t}_n$ and t_{n+1}. Recall that both are implicitly needed to compute the new residual $\boldsymbol{w}_{n+1}^{n+1}$ and that we want to choose the two coefficients such that the norm of this new residual is as small as possible. How to attain this goal is not so straightforward here. First, we need to express $\boldsymbol{w}_{n+1}^{n+1}$ as a function of the two coefficients, and then the challenge is to find alternative recursions that provide $\boldsymbol{w}_{n+1}^{n+1}$ and formulas for computing the two coefficients $\widetilde{\psi}_{n-1}$ and $\widetilde{\omega}_n$ so that still only two matrix–vector products are required per step. Zhang [15] succeeded in solving this problem. Here, we present a slightly different solution, one among several that were explored in [11].

First, a suitable expression for $\boldsymbol{w}_{n+1}^{n+1}$ is

$$\boldsymbol{w}_{n+1}^{n+1} = \boldsymbol{w}_{n+1}^{n} - \boldsymbol{A}\boldsymbol{w}_{n+1}^{n}\widetilde{\omega}_n + \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}\widetilde{\psi}_{n-1}\widetilde{\omega}_n, \tag{19}$$

which leads to the minimization problem

$$\|\boldsymbol{w}_{n+1}^{n+1}\| = \min_{\widetilde{\omega}_n, \widetilde{\psi}_{n-1}} \|\boldsymbol{w}_{n+1}^{n} - \boldsymbol{A}\boldsymbol{w}_{n+1}^{n}\widetilde{\omega}_n + \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}\widetilde{\psi}_{n-1}\widetilde{\omega}_n\| \tag{20}$$

in order to minimize the norm of the residual $\boldsymbol{w}_{n+1}^{n+1}$ with respect to $\widetilde{\omega}_n$ and $\widetilde{\psi}_{n-1}$. This minimization problem is a standard least square problem. The solution can be found by solving the 2×2 system

$$\begin{pmatrix} \|\boldsymbol{A}\boldsymbol{w}_{n+1}^{n}\|_2^2 & \langle \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}, \boldsymbol{A}\boldsymbol{w}_{n+1}^{n}\rangle \\ \hline \langle \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}, \boldsymbol{A}\boldsymbol{w}_{n+1}^{n}\rangle & \|\boldsymbol{A}\boldsymbol{u}_{n+1}^{n}\|_2^2 \end{pmatrix} \begin{pmatrix} \widetilde{\omega}_n \\ \chi \end{pmatrix} = \begin{pmatrix} \langle \boldsymbol{w}_{n+1}^{n}, \boldsymbol{A}\boldsymbol{w}_{n+1}^{n}\rangle \\ \langle \boldsymbol{w}_{n+1}^{n}, \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}\rangle \end{pmatrix} \tag{21}$$

and by solving $\chi = -\widetilde{\psi}_{n-1}\widetilde{\omega}_n$ for $\widetilde{\psi}_{n-1}$ afterwards. It requires computing five inner products. For $n = 0$ we let $\widetilde{\psi}_{-1} := 0$ and determine $\widetilde{\omega}_0$ by a 1-dimensional minimization. In the following, the determination of the coefficients $\widetilde{\omega}_n$ and $\widetilde{\psi}_{n-1}$ by solving (20) is indicated by the function

$$f : \boldsymbol{w}_{n+1}^{n}, \boldsymbol{A}\boldsymbol{w}_{n+1}^{n}, \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1} \mapsto [\widetilde{\omega}_n, \widetilde{\psi}_{n-1}]. \tag{22}$$

The solution of (21) requires the two matrix–vector products $\boldsymbol{A}\boldsymbol{w}_{n+1}^{n}$ and $\boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}$, which do not appear in (14) and which are not easily expressed by other known matrix–vector products. Nevertheless it is possible to find recursions so that a total of two matrix–vector products per iteration are enough. The following Algorithm 2 achieves this.

Algorithm 2. *For computing* $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ *choose an initial approximation* $\boldsymbol{x}_0$ *and let* $\widehat{\boldsymbol{w}}_0^0 := \boldsymbol{w}_0^0 := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$. *Set* $\boldsymbol{u}_1^{-1} := \boldsymbol{A}\boldsymbol{u}_1^{-1} := \boldsymbol{A}\widehat{\boldsymbol{u}}_0^{-1} := 0$. *Choose* $\widetilde{\boldsymbol{y}}_0$ *such that* $\widetilde{\delta}_0 := \langle \widetilde{\boldsymbol{y}}_0, \widehat{\boldsymbol{w}}_0^0\rangle \neq 0$ *and* $\langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{A}\widehat{\boldsymbol{w}}_0^0\rangle \neq 0$. *Then, compute for* $n = 0, 1, \ldots$

$$\begin{aligned}
\widetilde{\delta}'_n &:= \langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{A}\widehat{\boldsymbol{w}}_n^n\rangle, \\
\omega_n &:= \widetilde{\delta}_n / \widetilde{\delta}'_n, \\
\boldsymbol{u}_{n+1}^{n-1} &:= \boldsymbol{u}_n^{n-1} - \boldsymbol{A}\widehat{\boldsymbol{u}}_n^{n-1}\omega_n \qquad \textit{if } n \geqslant 1, \\
\boldsymbol{w}_{n+1}^{n-1} &:= \boldsymbol{w}_n^{n-1} - \boldsymbol{A}\widehat{\boldsymbol{w}}_n^{n-1}\omega_n \qquad \textit{if } n \geqslant 1, \\
\boldsymbol{w}_{n+1}^{n} &:= \boldsymbol{w}_n^{n} - \boldsymbol{A}\widehat{\boldsymbol{w}}_n^{n}\omega_n, \\
\boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1} &:= \frac{1}{\widetilde{\omega}_{n-1}}\big(\boldsymbol{w}_{n+1}^{n-1} - \boldsymbol{w}_{n+1}^{n}\big) \quad \textit{if } n \geqslant 1, \\
[\widetilde{\omega}_n, \widetilde{\psi}_{n-1}] &:= f\big(\boldsymbol{w}_{n+1}^{n}, \boldsymbol{A}\boldsymbol{w}_{n+1}^{n}, \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}\big), \\
\boldsymbol{u}_{n+1}^{n} &:= \boldsymbol{w}_{n+1}^{n} - \boldsymbol{u}_{n+1}^{n-1}\widetilde{\psi}_{n-1}, \\
\boldsymbol{A}\boldsymbol{u}_{n+1}^{n} &:= \boldsymbol{A}\boldsymbol{w}_{n+1}^{n} - \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}\widetilde{\psi}_{n-1}, \\
\boldsymbol{w}_{n+1}^{n+1} &:= \boldsymbol{w}_{n+1}^{n} - \boldsymbol{A}\boldsymbol{u}_{n+1}^{n}\widetilde{\omega}_n, \\
\widetilde{\delta}_{n+1} &:= \langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{w}_{n+1}^{n+1}\rangle,
\end{aligned}$$

$$\begin{aligned}
\psi_n &:= -\widetilde{\delta}_{n+1}/\big(\widetilde{\delta}'_n\widetilde{\omega}_n\big),\\
\widehat{\boldsymbol{w}}^{\,n}_{n+1} &:= \boldsymbol{w}^{\,n}_{n+1} - \widehat{\boldsymbol{w}}^{\,n}_{n}\psi_n,\\
\boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n}_{n+1} &:= \boldsymbol{A}\boldsymbol{w}^{\,n}_{n+1} - \boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n}_{n}\psi_n,\\
\boldsymbol{A}\widehat{\boldsymbol{u}}^{\,n}_{n} &:= \boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n}_{n} - \boldsymbol{A}\widehat{\boldsymbol{u}}^{\,n-1}_{n}\widetilde{\psi}_{n-1},\\
\boldsymbol{A}\widehat{\boldsymbol{u}}^{\,n}_{n+1} &:= \boldsymbol{A}\boldsymbol{u}^{\,n}_{n+1} - \boldsymbol{A}\widehat{\boldsymbol{u}}^{\,n}_{n}\psi_n,\\
\widehat{\boldsymbol{w}}^{\,n+1}_{n+1} &:= \widehat{\boldsymbol{w}}^{\,n}_{n+1} - \boldsymbol{A}\widehat{\boldsymbol{u}}^{\,n}_{n+1}\widetilde{\omega}_n.
\end{aligned}$$

Not yet given are recursions for the approximate solutions of the given system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. We actually obtain two per iteration: $\boldsymbol{x}^{\,n-1}_{n}$ and $\boldsymbol{x}^{\,n}_{n}$ are implicitly defined by

$$\boldsymbol{w}^{\,l}_{n} \equiv: \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{\,l}_{n} \quad (l = n-1, n). \tag{23}$$

From (13a) and (13e) we see by subtracting $\boldsymbol{b}$, multiplying by $-\boldsymbol{A}^{-1}$, and inserting (23) that

$$\boldsymbol{x}^{\,n}_{n+1} := \boldsymbol{x}^{\,n}_{n} + \widehat{\boldsymbol{w}}^{\,n}_{n}\omega_n, \qquad \boldsymbol{x}^{\,n+1}_{n+1} := \boldsymbol{x}^{\,n}_{n+1} + \boldsymbol{u}^{\,n}_{n+1}\widetilde{\omega}_n. \tag{24}$$

Together with the recursions in Algorithm 2 this defines a first version of the method BiCG×MR2_2×2. It is similar to, but different from Zhang's GPBiCG.

Our error analysis suggested to modify it further. To be precise, we modified the recursions for $\boldsymbol{A}\boldsymbol{u}^{\,n-1}_{n+1}$ and $\boldsymbol{u}^{\,n-1}_{n+1}$, for reasons that will become clear in Section 4. We replace the corresponding relations in Algorithm 2 by

$$\boldsymbol{A}\boldsymbol{u}^{\,n-1}_{n+1} := \boldsymbol{A}\boldsymbol{u}^{\,n-1}_{n} + \frac{\omega_n}{\widetilde{\omega}_{n-1}}\big(\boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n}_{n} - \boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n-1}_{n}\big), \tag{25a}$$

$$\boldsymbol{u}^{\,n-1}_{n+1} := \boldsymbol{u}^{\,n-1}_{n} + \frac{\omega_n}{\widetilde{\omega}_{n-1}}\big(\widehat{\boldsymbol{w}}^{\,n}_{n} - \widehat{\boldsymbol{w}}^{\,n-1}_{n}\big). \tag{25b}$$

Due to this replacement, other equations can be rearranged or become redundant. Further, we combine both equations in (24) to get a new recursion for the iterates $\boldsymbol{x}_n \equiv \boldsymbol{x}^{\,n}_{n}$:

$$\boldsymbol{x}_{n+1} := \boldsymbol{x}_n + \widehat{\boldsymbol{w}}^{\,n}_{n}\omega_n + \boldsymbol{u}^{\,n}_{n+1}\widetilde{\omega}_n. \tag{26}$$

Altogether we obtain the following Algorithm 3.

Algorithm 3 (BiCG×MR2_2×2). *For computing $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ choose an initial approximation $\boldsymbol{x}_0$ and let $\widehat{\boldsymbol{w}}^{\,0}_{0} := \boldsymbol{w}^{\,0}_{0} := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$. Set $\boldsymbol{u}^{-1}_{1} := \boldsymbol{A}\boldsymbol{u}^{-1}_{1} := \boldsymbol{A}\widehat{\boldsymbol{u}}^{-1}_{0} := 0$. Choose $\widetilde{\boldsymbol{y}}_0$ such that $\widetilde{\delta}_0 := \langle\widetilde{\boldsymbol{y}}_0, \widehat{\boldsymbol{w}}^{\,0}_{0}\rangle \neq 0$ and $\langle\widetilde{\boldsymbol{y}}_0, \boldsymbol{A}\widehat{\boldsymbol{w}}^{\,0}_{0}\rangle \neq 0$. Then, compute for $n = 0, 1, \ldots$*

$$\begin{aligned}
\widetilde{\delta}'_n &:= \langle\widetilde{\boldsymbol{y}}_0, \boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n}_{n}\rangle,\\
\omega_n &:= \widetilde{\delta}_n/\widetilde{\delta}'_n,\\
\boldsymbol{w}^{\,n}_{n+1} &:= \boldsymbol{w}^{\,n}_{n} - \boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n}_{n}\omega_n,\\
\boldsymbol{u}^{\,n-1}_{n+1} &:= \boldsymbol{u}^{\,n-1}_{n} + \frac{\omega_n}{\widetilde{\omega}_{n-1}}\big(\widehat{\boldsymbol{w}}^{\,n}_{n} - \widehat{\boldsymbol{w}}^{\,n-1}_{n}\big) \qquad \textit{if } n \geqslant 1,\\
\boldsymbol{A}\boldsymbol{u}^{\,n-1}_{n+1} &:= \boldsymbol{A}\boldsymbol{u}^{\,n-1}_{n} + \frac{\omega_n}{\widetilde{\omega}_{n-1}}\big(\boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n}_{n} - \boldsymbol{A}\widehat{\boldsymbol{w}}^{\,n-1}_{n}\big) \quad \textit{if } n \geqslant 1,\\
[\widetilde{\omega}_n, \widetilde{\psi}_{n-1}] &:= f\big(\boldsymbol{w}^{\,n}_{n+1}, \boldsymbol{A}\boldsymbol{w}^{\,n}_{n+1}, \boldsymbol{A}\boldsymbol{u}^{\,n-1}_{n+1}\big),
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{u}_{n+1}^{n} &:= \boldsymbol{w}_{n+1}^{n} - \boldsymbol{u}_{n+1}^{n-1}\widetilde{\psi}_{n-1},\\
\boldsymbol{A}\boldsymbol{u}_{n+1}^{n} &:= \boldsymbol{A}\boldsymbol{w}_{n+1}^{n} - \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}\widetilde{\psi}_{n-1},\\
\boldsymbol{x}_{n+1} &:= \boldsymbol{x}_{n} + \widehat{\boldsymbol{w}}_{n}^{n}\omega_n + \boldsymbol{u}_{n+1}^{n}\widetilde{\omega}_n,\\
\boldsymbol{w}_{n+1}^{n+1} &:= \boldsymbol{w}_{n+1}^{n} - \boldsymbol{A}\boldsymbol{u}_{n+1}^{n}\widetilde{\omega}_n,\\
\widetilde{\delta}_{n+1} &:= \langle \widetilde{\boldsymbol{y}}_0, \boldsymbol{w}_{n+1}^{n+1}\rangle,\\
\psi_n &:= -\widetilde{\delta}_{n+1}/\bigl(\widetilde{\delta}_n'\widetilde{\omega}_n\bigr),\\
\widehat{\boldsymbol{w}}_{n+1}^{n} &:= \boldsymbol{w}_{n+1}^{n} - \widehat{\boldsymbol{w}}_{n}^{n}\psi_n,\\
\boldsymbol{A}\widehat{\boldsymbol{w}}_{n+1}^{n} &:= \boldsymbol{A}\boldsymbol{w}_{n+1}^{n} - \boldsymbol{A}\widehat{\boldsymbol{w}}_{n}^{n}\psi_n,\\
\boldsymbol{A}\widehat{\boldsymbol{u}}_{n+1}^{n} &:= \boldsymbol{A}\boldsymbol{u}_{n+1}^{n} - \bigl(\boldsymbol{A}\widehat{\boldsymbol{w}}_{n}^{n} - \boldsymbol{A}\widehat{\boldsymbol{u}}_{n}^{n-1}\widetilde{\psi}_{n-1}\bigr)\psi_n,\\
\widehat{\boldsymbol{w}}_{n+1}^{n+1} &:= \widehat{\boldsymbol{w}}_{n+1}^{n} - \boldsymbol{A}\widehat{\boldsymbol{u}}_{n+1}^{n}\widetilde{\omega}_n.
\end{aligned}$$

3. Comparison of BiCG×MR2_2×2 and GPBiCG

Two questions arise when comparing our algorithm BiCG×MR2_2×2 with Zhang's GPBiCG: are they mathematically equivalent and, if equivalent, what are the differences between them?

The answer to the first question was given in [11]:

Theorem 1. *The algorithms* BiCG×MR2_2×2 *and* GPBiCG *generate in exact arithmetic with identical starting values $\boldsymbol{x}_0$ and $\widetilde{\boldsymbol{y}}_0$ the same iterates $\boldsymbol{x}_n$, $n = 1, \ldots$.*

The idea of the proof is to show that the residuals are the same and thus also the iterates. In both algorithms, the residuals are defined by a product of two polynomials. So it is sufficient to prove the identity of the underlying polynomials, which can be achieved by induction.

One small difference between the two algorithms is in the requirements in operations and memory usage. Table 1 shows the average costs per Krylov space dimension for different methods. Our Algorithm 2 uses the same amount of vectors as GPBiCG, whereas BiCG×MR2_2×2 requires one additional vector. The number of operations per Krylov space dimension is slightly lower in both our algorithms.

Although the algorithms BiCG×MR2_2×2 and GPBiCG realize the same method in exact arithmetic, they behave differently in finite arithmetic due to different recurrences. An example is shown in Fig. 2 with the matrix "Sherman5" from the Matrix Market [http://math.nist.gov/MatrixMarket/]. Our algorithm attains a slightly higher ultimate accuracy. However, numerical experiments show that the behavior is different for other starting values. Thus, for each matrix we did ten different experiments and show the averages in Table 2. We report the dimension n and the number of entries of the matrices, the number of iterations n_{12} to reduce the residual norm by a factor of 10^{12} and the ultimate (relative) accuracy where the residual norm stagnates. All matrices can be retrieved from the Matrix Market. As can be seen, there are not any significant differences between the algorithms. The differences in n_{12} are less than 5%. The ultimate accuracy of BiCG×MR2_2×2 is mostly slightly higher than the one of GPBiCG.

Table 1
Average costs per Krylov space dimension

method	MV(S)	AXPY	DOT	memory
BiCG	2	6.5	2	7
CGS	1	3.25	1	7
BiCGStab	1	3	2	7
BiCGStab2	1	5.5	2.75	10
BiCGStab(2)	1	3.75	2.25	9
GPBiCG	1	7.5	3.5	11
Algorithm 2	1	7	3.5	11
BiCG×MR2_2×2	1	7	3.5	12

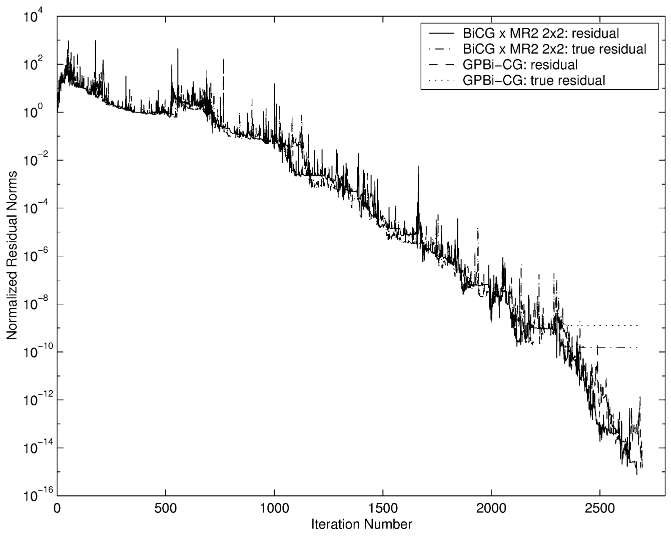


Fig. 2. Sherman5.

4. Error analysis of BiCG×MR2_2×2

In exact arithmetic the residuals $\boldsymbol{w}_n^n$ and the iterates $\boldsymbol{x}_n$ satisfy the equation

$$\boldsymbol{w}_n^n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n. \tag{27}$$

However, in finite arithmetic, this equation does no longer hold, in particular if we compute $\boldsymbol{w}_n^n$ by recursions, as we do here. Therefore we call $\boldsymbol{w}_n^n$ the recursive residual and the right-hand side of Eq. (27)

Table 2
Comparison of BiCG×MR2_2×2 und GPBiCG for various matrices from the Matrix Market. The reported values are averages for ten different starting values

matrix	n	nnz	BiCG×MR2_2×2		GPBiCG	
			ult.acc.	n_{12}	ult.acc.	n_{12}
fs6802	680	2424	1.7e−14	1075	1.2e−14	1056
nos3	960	627	3.3e−15	223	3.3e−15	222
nos6	675	1965	7.9e−13	2565	1.3e−12	2650
1138bus	1138	2596	1.1e−12	2766	1.3e−12	2809
saylr3	1000	3750	4.1e−15	450	4.5e−15	448
saylr4	3564	22316	1.0e−12	2273	6.3e−13	2254
gre115	115	421	1.0e−14	100	4.9e−15	97
gre185	185	1005	5.0e−11	675	7.4e−11	671
e05r0000	236	5856	5.3e−11	508	1.6e−10	525
sherman1	1000	3750	6.2e−13	494	6.2e−13	491
sherman3	5005	20033	3.9e−09	6524	3.6e−09	6546
sherman4	1104	3786	6.4e−13	122	6.5e−13	122
sherman5	3312	20793	3.3e−10	2589	7.9e−10	2648

the true residual (actually we should say “true residual of the recursively computed $\boldsymbol{x}_n$”). We define the gap $\boldsymbol{e}_n$ between these residuals as

$$\boldsymbol{e}_n := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n - \boldsymbol{w}_n^n. \tag{28}$$

All vectors appearing on the right-hand side in (28) denote the values from BiCG×MR2_2×2 (Algorithm 3) computed in floating-point arithmetic.

Recently it has been shown that for three-term recurrences [8], the gap between the recursive and the true residual has the form:

$$\begin{aligned}\boldsymbol{e}_{n+1} = \boldsymbol{e}_0 &- \sum_{j=0}^{n} \boldsymbol{l}_j \\ &- \boldsymbol{l}_0\left(\frac{\beta_0}{\gamma_1} + \cdots + \frac{\beta_0\cdots\beta_{n-1}}{\gamma_1\cdots\gamma_n}\right) \\ &- \boldsymbol{l}_1\left(\frac{\beta_1}{\gamma_2} + \cdots + \frac{\beta_1\cdots\beta_{n-1}}{\gamma_2\cdots\gamma_n}\right) \\ &\;\;\vdots \\ &- \boldsymbol{l}_{n-1}\frac{\beta_{n-1}}{\gamma_n}.\end{aligned} \tag{29}$$

The vectors $\boldsymbol{l}_n$ are the local errors due to the roundoff in step n. In contrast, for 2-term recurrences the gap is just a sum of local errors $\boldsymbol{l}_j^G$ [4]:

$$\boldsymbol{e}_{n+1}^G = \boldsymbol{e}_0 - \sum_{j=0}^{n} \boldsymbol{l}_j^G. \tag{30}$$

Since the local errors $\boldsymbol{l}_j$ and $\boldsymbol{l}_j^G$ are of comparable size, the level of accuracy for 2-term recursions is usually better than the one of 3-term recursions. The difference may be very large if some of the quotients in (29) are large, which may even happen if the conjugate gradient method is applied to a symmetric positive definite matrix. We refer the reader to [8] for further information.

In the following, we will derive a formula for the gap $\boldsymbol{e}_n$ resulting from our algorithm BiCG×MR2_2×2. Since the equations in this algorithm are not exactly fulfilled in finite arithmetic, we add in each equation an error term $\boldsymbol{f}_n^x$, $x \in \{a, \dots, p\}$, which is either a scalar or a vector depending on the equation. As an example, the third equation gives:

$$\boldsymbol{w}_{n+1}^n = \boldsymbol{w}_n^n - \underline{\boldsymbol{A}\widehat{\boldsymbol{w}}_n^n}\omega_n + \boldsymbol{f}_n^c. \tag{31}$$

The vector $\underline{\boldsymbol{A}\widehat{\boldsymbol{w}}_n^n}$ denotes the matrix–vector product computed in finite arithmetic. On the contrary, $\boldsymbol{A}\widehat{\boldsymbol{w}}_n^n$ is the exact matrix–vector product of $\boldsymbol{A}$ with $\widehat{\boldsymbol{w}}_n^n$ from BiCG×MR2_2×2. Thus, we have to replace each matrix–vector product in this algorithm by its underlined equivalent. Additionally, we introduce the vectors

$$\widetilde{\boldsymbol{f}}_n^1 := \underline{\boldsymbol{A}\widehat{\boldsymbol{w}}_n^n} - \boldsymbol{A}\widehat{\boldsymbol{w}}_n^n, \tag{32}$$

$$\widetilde{\boldsymbol{f}}_n^2 := \underline{\boldsymbol{A}\boldsymbol{w}_{n+1}^n} - \boldsymbol{A}\boldsymbol{w}_{n+1}^n. \tag{33}$$

They express the errors resulting from the computation of the matrix–vector products $\boldsymbol{A}\widehat{\boldsymbol{w}}_n^n$ and $\boldsymbol{A}\boldsymbol{w}_{n+1}^n$ in floating point arithmetic. Moreover, we define the vectors

$$\overline{\boldsymbol{e}}_n := \underline{\boldsymbol{A}\boldsymbol{u}_{n+1}^n} - \boldsymbol{A}\boldsymbol{u}_{n+1}^n, \tag{34}$$

$$\widetilde{\boldsymbol{e}}_n := \underline{\boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}} - \boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1} \tag{35}$$

to simplify the notation.

Starting point for the following derivation is the definition (28), where we substitute equations for $\boldsymbol{x}_{n+1}$, $\boldsymbol{w}_{n+1}^{n+1}$ and $\boldsymbol{w}_{n+1}^n$ from BiCG×MR2_2×2:

$$\boldsymbol{e}_{n+1} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{n+1} - \boldsymbol{w}_{n+1}^{n+1} = \boldsymbol{e}_n + \overline{\boldsymbol{e}}_n\widetilde{\omega}_n + \widetilde{\boldsymbol{f}}_n^1\omega_n - \boldsymbol{A}\boldsymbol{f}_n^i - \boldsymbol{f}_n^j - \boldsymbol{f}_n^c. \tag{36}$$

In the same way we get for $\overline{\boldsymbol{e}}_n$:

$$\overline{\boldsymbol{e}}_n = -\widetilde{\boldsymbol{e}}_n\widetilde{\psi}_{n-1} + \widetilde{\boldsymbol{f}}_n^2 + \boldsymbol{f}_n^h - \boldsymbol{A}\boldsymbol{f}_n^g \tag{37}$$

and for $\widetilde{\boldsymbol{e}}_n$:

$$\widetilde{\boldsymbol{e}}_n = \overline{\boldsymbol{e}}_{n-1} - \frac{\omega_n}{\widetilde{\omega}_{n-1}}\left(\widetilde{\boldsymbol{f}}_n^1 + \widetilde{\boldsymbol{f}}_{n-1}^2 - \widetilde{\boldsymbol{f}}_{n-1}^1\psi_{n-1} + \boldsymbol{f}_{n-1}^n - \boldsymbol{A}\boldsymbol{f}_{n-1}^m\right) + \boldsymbol{f}_n^e - \boldsymbol{A}\boldsymbol{f}_n^d. \tag{38}$$

Without the modified equations for $\boldsymbol{u}_{n+1}^{n-1}$ and $\boldsymbol{A}\boldsymbol{u}_{n+1}^{n-1}$ in BiCG×MR2_2×2, the derivation for (38) would not be possible.

Combining equations (37) and (38) leads to a recursion for $\overline{\boldsymbol{e}}_n$:

$$\overline{\boldsymbol{e}}_n = -\overline{\boldsymbol{e}}_{n-1}\widetilde{\psi}_{n-1} + \overline{\boldsymbol{l}}_{n-1} = \left(\prod_{k=0}^{n-1}\widetilde{\psi}_k\right)\overline{\boldsymbol{e}}_0 + \sum_{i=0}^{n-2}\left(\prod_{k=i+1}^{n-1}\widetilde{\psi}_k\right)\overline{\boldsymbol{l}}_i + \overline{\boldsymbol{l}}_{n-1}, \tag{39}$$

where

$$\begin{aligned}\overline{\boldsymbol{l}}_{n-1} = {} & \widetilde{\boldsymbol{f}}_n^{2} + \boldsymbol{f}_n^{h} - \boldsymbol{A}\boldsymbol{f}_n^{g} \\ & + \widetilde{\psi}_{n-1}\left(\frac{\omega_n}{\widetilde{\omega}_{n-1}}\left(\widetilde{\boldsymbol{f}}_n^{1} + \widetilde{\boldsymbol{f}}_{n-1}^{2} - \widetilde{\boldsymbol{f}}_{n-1}^{1}\psi_{n-1} + \boldsymbol{f}_{n-1}^{n} - \boldsymbol{A}\boldsymbol{f}_{n-1}^{m}\right) + \boldsymbol{f}_n^{e} - \boldsymbol{A}\boldsymbol{f}_n^{d}\right).\end{aligned} \tag{40}$$

In summary we have the following formula for the gap $\boldsymbol{e}_n$:

$$\boldsymbol{e}_{n+1} = \boldsymbol{e}_n + \overline{\boldsymbol{e}}_n\widetilde{\omega}_n + \boldsymbol{l}_n = \boldsymbol{e}_0 + \sum_{i=0}^{n}\left(\overline{\boldsymbol{e}}_i\widetilde{\omega}_i + \boldsymbol{l}_i\right), \tag{41}$$

with the local error

$$\boldsymbol{l}_n = \widetilde{\boldsymbol{f}}_n^{1}\omega_n - \boldsymbol{A}\boldsymbol{f}_n^{i} - \boldsymbol{f}_n^{j} - \boldsymbol{f}_n^{c}. \tag{42}$$

As expected, (41) looks like formula (30) for 2-term recursions. Unfortunately, $\overline{\boldsymbol{e}}_n$ is part of $\boldsymbol{e}_n$, and its recursion is similar to the error of a 3-term recursion.

Table 3
Accuracy of BiCG×MR2_2×2 with two and three matrix–vector products per iteration. The reported values are averages for ten different starting values

matrix	BiCG×MR2_2×2 with 2mv		BiCG×MR2_2×2 with 3mv	
	ult.acc	n_{12}	ult.acc	n_{12}
fs6802	3.5e−14	1051	2.6e−15	1020
nos3	5.0e−15	221	2.5e−15	222
nos6	3.2e−12	2629	2.0e−14	2384
1138bus	1.2e−12	2862	2.1e−14	2921
saylr3	6.8e−15	449	3.1e−15	442
saylr4	1.1e−12	2279	5.0e−14	2234
gre115	1.8e−15	97	6.6e−16	97
gre185	6.3e−11	760	3.4e−12	640
e05r0000	1.5e−10	534	1.6e−12	514
sherman1	2.5e−13	486	1.1e−13	485
sherman3	3.5e−09	6889	1.5e−10	6988
sherman4	6.5e−13	121	4.3e−13	122
sherman5	2.9e−10	2506	5.4e−11	2534

Next, we investigate the influence of $\overline{\boldsymbol{e}}_n$ on the error $\boldsymbol{e}_n$. First, we modify BiCG×MR2_2×2, by replacing the indirect computation of $\boldsymbol{Au}_{n+1}^{n-1}$ by the corresponding direct matrix–vector product. As a consequence, the error $\boldsymbol{e}_n$ has now the form

$$\boldsymbol{e}_{n+1}^{\mathrm{mod}} = \boldsymbol{e}_0 + \sum_{i=0}^{n} \big(\widetilde{\boldsymbol{f}}_i^{\,3} \widetilde{\omega}_i + \boldsymbol{l}_i \big), \tag{43}$$

where $\widetilde{\boldsymbol{f}}_n^{\,3} := \underline{\boldsymbol{Au}_{n+1}^{n}} - \boldsymbol{Au}_{n+1}^{n}$. In other words, the error $\boldsymbol{e}_{n+1}^{\mathrm{mod}}$ is now independent of $\overline{\boldsymbol{e}}_n$.

In Table 3, we compare BiCG×MR2_2×2 without and with the above modification. We list again the number of iterations n_{12} to reduce the residual norm by a factor of 10^{12} and the (relative) ultimate accuracy. All reported values are averages of ten different numerical experiments. Since the starting values in these experiments differ from those for Table 2, the reported ultimate accuracy and n_{12} differ also. We observe that the ultimate accuracy is higher for BiCG×MR2_2×2 with three matrix–vector products. The difference is especially significant if n_{12} is greater than the dimension of the matrix. This seems obvious, since the norm of the error $\overline{\boldsymbol{e}}_n$ is likely to grow with increasing n.

5. Conclusions

We have derived BiCG×MR2_2×2, an LTPM whose second sequence of polynomials is determined through a 2-dimensional minimization of the residual norm in each step, like in Zhang's GPBiCG. Both algorithms are mathematically equivalent and based on two pairs of coupled 2-term recurrences. However, BiCG×MR2_2×2 determines the recurrence coefficients of the second set of polynomials in a different way and does not use the same intermediate quantities as GPBiCG. Some of the recursions for residuals and iterates are simpler and are closer related to each other in order to make the gap between updated and true residuals as small as possible. We have analyzed this gap and have provided a formula for it. Despite the use of 2-term recurrences it has still some elements of a 3-term recurrence in it, but it is much easier to estimate than the one for GPBiCG. In our numerical examples of sizes up to 5005 the two algorithms turned out to be about equally accurate. Their behavior for very large problems with suitable preconditioners has still to be investigated.

The recurrences of our algorithm allow to improve it further. Approaches to avoid a significant loss of accuracy have been investigated and tested in [11]: it turns out that the adaptation of the correction scheme of van der Vorst and Ye [14] is quite complicated, while variations of Neumaier's scheme [10] can be implemented more easily. Finally, a look-ahead algorithm for our main version of BiCG×MR2_2×2 has been implemented in [11].

References

[1] Z.-H. Cao, On the QMR approach for iterative methods including coupled three-term recurrences for solving nonsymmetric linear systems, Appl. Numer. Math. 27 (1998) 123–140.

[2] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G.A. Watson (Ed.), Numerical Analysis, Dundee, 1975, Lecture Notes in Math., Vol. 506, Springer, Berlin, 1976, pp. 73–89.

[3] D.R. Fokkema, G.L.G. Sleijpen, H.A. van der Vorst, Generalized conjugate gradient squared, J. Comput. Appl. Math. 71 (1996) 125–146.

[4] A. Greenbaum, Estimating the attainable accuracy of recursively computed residual methods, SIAM J. Matrix Anal. Appl. 18 (3) (1997) 535–551.
[5] M.H. Gutknecht, Variants of BiCGSTAB for matrices with complex spectrum, SIAM J. Sci. Statist. Comput. 14 (5) (1993) 1020–1033.
[6] M.H. Gutknecht, Local minimum residual smoothing, Talk at Oberwolfach, Germany, April 1994.
[7] M.H. Gutknecht, Lanczos-type solvers for nonsymmetric linear systems of equations, Acta Numer. 6 (1997) 271–397.
[8] M.H. Gutknecht, Z. Strakoš, Accuracy of two three-term and three two-term recurrences for Krylov space solvers, SIAM J. Matrix Anal. Appl. 22 (1) (2000) 213–229.
[9] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Nat. Bureau Standards 49 (1952) 33–53.
[10] A. Neumaier, Iterative regularization for large-scale ill-conditioned linear systems, Talk at Oberwolfach, April 1994.
[11] S. Röllin, Auf 2-Term-Rekursionen beruhende Produktmethoden vom Lanczos-Typ, Diplomarbeit, Seminar für Angewandte Mathematik, ETH Zürich, 2000.
[12] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 10 (1989) 36–52.
[13] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems, SIAM J. Sci. Statist. Comput. 13 (2) (1992) 631–644.
[14] H.A. van der Vorst, Q. Ye, Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals, SIAM J. Sci. Comput. 22 (2000) 835–852.
[15] S.-L. Zhang, GPBI-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 18 (2) (1997) 537–551.