Applied Numerical Mathematics 67 (2013) 4–16

Contents lists available at ScienceDirect

Applied Numerical Mathematics

www.elsevier.com/locate/apnum

Solving linear equations with a stabilized GPBiCG method

Kuniyoshi Abe [a,*], Gerard L.G. Sleijpen [b]

[a] *Faculty of Economics and Information, Gifu Shotoku University, Nakauzura, Gifu 500-8288, Japan*
[b] *Department of Mathematics, Utrecht University, P.O. Box 80.010, 3508 TA Utrecht, The Netherlands*

ARTICLE INFO

Article history:
Available online 22 June 2011

Keywords:
Linear systems
Krylov subspace method
Bi-CG
Generalized product-type Bi-CG method
Hybrid Bi-CG
Lanczos-type method

ABSTRACT

Any residual polynomial of hybrid Bi-Conjugate Gradient (Bi-CG) methods, as Bi-CG STABilized (Bi-CGSTAB), BiCGstab(ℓ), Generalized Product-type Bi-CG (GPBiCG), and BiCG $\times$ MR2, can be expressed as the product of a Lanczos polynomial and a so-called stabilizing polynomial. The stabilizing polynomials of GPBiCG have originally been built by coupled two-term recurrences, but, as in BiCG $\times$ MR2, they can also be constructed by a three-term recurrence similar to the one for the Lanczos polynomials. In this paper, we propose to use this three-term recurrence and to combine it with a slightly modified version of the coupled two-term recurrences for Bi-CG. The modifications appear to lead to more accurate Bi-CG coefficients. We consider two combinations. The recurrences of the resulting two algorithms are different from those of the original GPBiCG, BiCG $\times$ MR2, and other variants in literature. Specifically in cases where the convergence has a long stagnation phase, the convergence seems to rely on the underlying Bi-CG process. We therefore also propose a "stabilization" strategy that allows the Bi-CG coefficients in our variants to be more accurately computed. Numerical experiments show that our two new variants are less affected by rounding errors, and a GPBiCG method with the stabilization strategy is more effective.

1. Introduction

Bi-Conjugate Gradient (Bi-CG) [2] is a well-known method for solving linear systems

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b},$$

for $\boldsymbol{x}$, where $\mathbf{A}$ is a given n-by-n matrix, and $\boldsymbol{b}$ is a given n-vector. Typically, the dimension n of the linear system is high and the matrix $\mathbf{A}$ is sparse. A number of hybrid Bi-CG methods such as Conjugate Gradient Squared (CGS) [12], Bi-CG STABilized (Bi-CGSTAB) [14], BiCGStab2 [3], BiCGstab(ℓ) [9], Generalized Product-type Bi-CG (GPBiCG) [15], and BiCG $\times$ MR2 [4,1] have been developed to improve the convergence of Bi-CG and to avoid multiplication by the transpose $\mathbf{A}^T$ of $\mathbf{A}$, or, in case of a complex system, by the conjugate transpose $\mathbf{A}^*$ of $\mathbf{A}$.

The residual polynomials of hybrid Bi-CG methods can be expressed as the product of the bi-Lanczos polynomials [13] and other, so-called *stabilizing polynomials* [10], say P_k, with $P_k(0) = 1$ and degree k equal to the degree of the Lanczos polynomials. In GPBiCG and BiCG$\times$MR2, these stabilizing polynomials can be generated by a three-term recurrence similar to the Lanczos polynomials but with different recurrence coefficients. As the Lanczos polynomials, they can also be obtained by coupled two-term recurrences. GPBiCG exploits the coupled two-term version, while BiCG$\times$MR2 exploits the three-term variant. These two methods are mathematically equivalent, but show different stability behavior. A variant has been

* Corresponding author.
E-mail address: abe@gifu.shotoku.ac.jp (K. Abe).

0168-9274/$30.00 © 2011 IMACS. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.apnum.2011.06.010

proposed by Röllin and Gutknecht in [6] to improve the stability of the original GPBiCG method. As in GPBiCG, the stabilizing polynomials of this variant also rely on the coupled two-term recurrences. We therefore refer to this variant as the RG variant of GPBiCG, even though its derivation is based on the one in [4] rather than on the less transparent one in [15].

As in BiCG $\times$ MR2, we propose to redesign the recurrences of GPBiCG by coupling the coupled two-term recurrences of Bi-CG with the three-term recurrence for the stabilizing. However, the coupled two-term recurrences of Bi-CG that we use differ slightly from the standard ones. This modification appears to lead to slightly more accurate Bi-CG coefficients and allows a more elegant derivation of GPBiCG, that is, of an algorithm based on Zhang's idea [15]. Moreover, we consider two different ways of combining the Bi-CG recurrences and the recurrences for the stabilizing polynomials. Specifically in cases where the convergence has a long stagnation phase, the convergence seems to rely on the underlying Bi-CG process. Then, it is important to have Bi-CG coefficients that are as accurate as possible. Following [11], we therefore also propose a strategy to select the coefficients of the stabilizing polynomials such that the angle between the vectors in the inner products for the Bi-CG coefficients is as small as possible. This *stabilization* strategy "stabilizes" the computation of the Bi-CG coefficients: it leads to more accurate Bi-CG coefficients. It may avoid stagnation and lead to faster computation. Other aspects of effects of rounding errors have been analyzed in [6].

We compare the convergence of the original GPBiCG, BiCG $\times$ MR2, the RG variant of GPBiCG, and our two new variants. These variants are mathematically equivalent, but our numerical experiments show that our new variants are less affected by rounding errors than the other three. Numerical results also show that GPBiCG with the stabilization strategy is more effective.

We first outline the original GPBiCG method (Section 2). Then, in Section 3, we derive the alternative implementations of GPBiCG/BiCG $\times$ MR2. In Section 4, we propose the stabilization strategy for improving the accuracy of the Bi-CG coefficients. Numerical experiments in Section 5 show that our proposed variants and the variants with the stabilization strategy are more effective than the others. In Appendix A, we propose two other variants of GPBiCG/BiCG $\times$ MR2 by using the two-term recurrences of Rutishauser [7] for the stabilizing and combining it with the slightly modified version of the coupled two-term recurrences for Bi-CG.

2. Outline of the original GPBiCG method

In this section, we give an outline of the original GPBiCG method to show that the original GPBiCG method has been derived from the coupled two-term recurrences.

The residuals $\boldsymbol{r}_k^{\mathrm{bcg}}$ of Bi-CG are generated by the two-term recurrences

$$\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k \mathbf{A}\boldsymbol{u}_k^{\mathrm{bcg}}, \tag{1}$$

$$\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k \boldsymbol{u}_k^{\mathrm{bcg}}, \tag{2}$$

where, for each k, the recurrence coefficients α_k and β_k are such that

$$\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \mathbf{A}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} \perp \tilde{\boldsymbol{r}}_k. \tag{3}$$

The n-vector $\tilde{\boldsymbol{r}}_0$ is selected at the initialization of Bi-CG, the n-vectors $\tilde{\boldsymbol{r}}_0, \tilde{\boldsymbol{r}}_1, \ldots, \tilde{\boldsymbol{r}}_{k-1}$ are such that they span the kth Krylov subspace $\mathcal{K}_k(\mathbf{A}^*; \tilde{\boldsymbol{r}}_0)$ for each k. An induction argument shows that $\boldsymbol{r}_k^{\mathrm{bcg}} \perp \mathcal{K}_k(\mathbf{A}^*; \tilde{\boldsymbol{r}}_0)$.

The residuals $\mathbf{r}_k$ of GPBiCG can be expressed as

$$\mathbf{r}_k \equiv P_k(\mathbf{A})\boldsymbol{r}_k^{\mathrm{bcg}},$$

where the polynomials $P_k(\lambda)$ satisfy the three-term recurrence relation ([15]).

$$\begin{cases} P_0(\lambda) = 1, \qquad P_1(\lambda) = (1 - \zeta_0\lambda)P_0(\lambda), \\ P_{k+1}(\lambda) = (1 + \eta_k - \zeta_k\lambda)P_k(\lambda) - \eta_k P_{k-1}(\lambda), \quad k = 1, 2, \ldots. \end{cases} \tag{4}$$

The polynomial $P_k(\lambda)$ is of degree k. It is referred to as the *stabilizing polynomial* [10]. The recurrence coefficients ζ_k and η_k are computed to minimize the residual in 2-norm as a function of ζ_k and η_k. However, in the original GPBiCG method, the three-term recurrence (4) has not been used for the actual construction of the polynomial P_k. An auxiliary polynomial G_k of degree k,

$$G_k(\lambda) \equiv \big(P_k(\lambda) - P_{k+1}(\lambda)\big)/\lambda, \quad k = 0, 1, \ldots \tag{5}$$

was introduced to transform the three-term recurrences (4) into the coupled two-term recurrences

$$G_{-1}(\lambda) \equiv 0, \qquad P_0(\lambda) = 1,$$

$$G_k(\lambda) = \zeta_k P_k(\lambda) + \eta_k G_{k-1}(\lambda), \tag{6}$$

$$P_{k+1}(\lambda) = P_k(\lambda) - \lambda G_k(\lambda), \quad k = 0, 1, 2, \ldots. \tag{7}$$

We now indicate how in [15] a combination of the coupled two-term recurrences (6)–(7) with (1)–(2) led to the recurrences of residual vectors $\mathbf{r}_k$ and approximate solutions $\mathbf{x}_k$ of the original GPBiCG algorithm, where $\mathbf{r}_k = \boldsymbol{b} - \mathbf{A}\mathbf{x}_k$.

The update formulas for the residuals $\mathbf{r}_k$ read as

$$\begin{aligned}
&\mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k \boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k \mathbf{A}\mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}},\\
&\mathbf{A}\mathbf{G}_{k-1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_{k-1} \boldsymbol{r}_k^{\mathrm{bcg}} - \mathbf{P}_k \boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k \mathbf{A}\mathbf{P}_{k-1} \boldsymbol{u}_k^{\mathrm{bcg}} + \alpha_k \mathbf{A}\mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}},\\
&\mathbf{P}_{k+1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \eta_k \mathbf{A}\mathbf{G}_{k-1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \zeta_k \mathbf{A}\mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}},\\
&\mathbf{A}\mathbf{G}_k \boldsymbol{u}_k^{\mathrm{bcg}} = \zeta_k \mathbf{A}\mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}} + \eta_k \big(\mathbf{P}_{k-1} \boldsymbol{r}_k^{\mathrm{bcg}} - \mathbf{P}_k \boldsymbol{r}_k^{\mathrm{bcg}} - \beta_{k-1} \mathbf{A}\mathbf{G}_{k-1} \boldsymbol{u}_{k-1}^{\mathrm{bcg}}\big),\\
&\mathbf{P}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_{k+1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k \mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}} + \beta_k \mathbf{A}\mathbf{G}_k \boldsymbol{u}_k^{\mathrm{bcg}},\\
&\mathbf{A}\mathbf{P}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k \mathbf{A}\mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}}.
\end{aligned}$$

Here, we simply put $\mathbf{P}_k$ and $\mathbf{G}_k$ for the matrix polynomials $P_k(\mathbf{A})$ and $G_k(\mathbf{A})$, respectively. The first formula is obtained by multiplying (1) by $\mathbf{P}_k$. But the combination required to obtain some other formulas may be less obvious. For instance, the second formula is a combination of (1) multiplied by $\mathbf{A}\mathbf{G}_{k-1}$ and (5). For more details, however, we refer to [15].

By introducing the auxiliary vectors

$$\begin{aligned}
&\mathbf{t}_k \equiv \mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \qquad \mathbf{y}_k \equiv \mathbf{A}\mathbf{G}_{k-1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \qquad \mathbf{p}_k \equiv \mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}},\\
&\mathbf{w}_k \equiv \mathbf{A}\mathbf{P}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \qquad \mathbf{v}_k \equiv \mathbf{A}\mathbf{G}_k \boldsymbol{u}_k^{\mathrm{bcg}},
\end{aligned}$$

we obtain the recurrences of the original GPBiCG for updating the residual:

$$\mathbf{t}_k = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k, \tag{8}$$

$$\mathbf{y}_k = \mathbf{t}_{k-1} - \mathbf{r}_k - \alpha_k \mathbf{w}_{k-1} + \alpha_k \mathbf{A}\mathbf{p}_k, \tag{9}$$

$$\mathbf{r}_{k+1} = \mathbf{t}_k - \eta_k \mathbf{y}_k - \zeta_k \mathbf{A}\mathbf{t}_k, \tag{10}$$

$$\mathbf{v}_k = \zeta_k \mathbf{A}\mathbf{p}_k + \eta_k (\mathbf{t}_{k-1} - \mathbf{r}_k - \beta_{k-1} \mathbf{v}_{k-1}), \tag{11}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} - \beta_k (\mathbf{p}_k - \mathbf{v}_k), \tag{12}$$

$$\mathbf{w}_k = \mathbf{A}\mathbf{t}_k + \beta_k \mathbf{A}\mathbf{p}_k. \tag{13}$$

We now indicate the update formulas for the approximate solution $\mathbf{x}_k$. Combine (1) and (7) to see that the residual $\mathbf{r}_{k+1}$ can also be expressed by

$$\mathbf{P}_{k+1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k \boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k \mathbf{A}\mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}} - \mathbf{A}\mathbf{G}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}}.$$

With the auxiliary vector $\mathbf{z}_k \equiv \mathbf{G}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, this reads as

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k - \mathbf{A}\mathbf{z}_k.$$

Since $\mathbf{r}_k \equiv \boldsymbol{b} - \mathbf{A}\mathbf{x}_k$, we see that

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k + \mathbf{z}_k. \tag{14}$$

This formula requires a recurrence for updating $\mathbf{z}_k$. Combining (1) with (6) gives

$$\mathbf{G}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \zeta_k \mathbf{P}_k \boldsymbol{r}_k^{\mathrm{bcg}} + \eta_k \mathbf{G}_{k-1} \boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k \mathbf{A}\mathbf{G}_k \boldsymbol{u}_k^{\mathrm{bcg}},$$

and, therefore,

$$\mathbf{z}_k = \zeta_k \mathbf{r}_k + \eta_k \mathbf{z}_{k-1} - \alpha_k \mathbf{v}_k. \tag{15}$$

The recurrences of the original GPBiCG for updating the residual and the approximate solution are given by (8)–(15).

We finally describe how the recurrence coefficients α_k, β_k, ζ_k and η_k can be computed. The recurrence coefficients α_k and β_k of the original GPBiCG method are mathematically equivalent to those of Bi-CG, and by transforming those of Bi-CG, the coefficients are computed (see [15]) as follows:

$$\alpha_k = \frac{\mathbf{r}_k^* \tilde{\boldsymbol{r}}_0}{(\mathbf{A}\mathbf{p}_k)^* \tilde{\boldsymbol{r}}_0}, \qquad \beta_k = \frac{\alpha_k}{\zeta_k} \frac{\mathbf{r}_{k+1}{}^* \tilde{\boldsymbol{r}}_0}{\mathbf{r}_k{}^* \tilde{\boldsymbol{r}}_0}.$$

The recurrence coefficients ζ_k and η_k of the original GPBiCG method are selected such that the residual norm $\|\mathbf{r}_{k+1}\|_2 = \|\mathbf{t}_k - \eta_k \mathbf{y}_k - \zeta_k \mathbf{A}\mathbf{t}_k\|_2$ is minimized as the function of ζ_k and η_k. With $\mathbf{Q}_k \equiv [\mathbf{A}\mathbf{t}_k, \mathbf{y}_k]$, $(\zeta_k, \eta_k)^{\mathrm{T}} = (\mathbf{Q}_k^* \mathbf{Q}_k)^{-1} \mathbf{Q}_k^* \mathbf{t}_k$.

The RG variant of GPBiCG as mentioned in Section 1, is mathematically equivalent to the original GPBiCG method, and, as in the original GPBiCG, the recurrences of the RG variant have also been obtained by combining those of Bi-CG with coupled two-term recurrences; for details, we refer to [6].

3. Alternative implementations of the GPBiCG method

We derive alternative implementations of GPBiCG by combining the recurrences of Bi-CG and the three-term recurrence (4). One of these variants (variant 1) is closely related to BiCG × MR2.

Note that, for n-vectors $\boldsymbol{r}, \boldsymbol{c}$ and $\tilde{\boldsymbol{r}}$, the orthogonality condition $\boldsymbol{r} - \beta\boldsymbol{c} \perp \tilde{\boldsymbol{r}}$ implicitly defines the scalar β: $\beta = \tilde{\boldsymbol{r}}^*\boldsymbol{c}/\tilde{\boldsymbol{r}}^*\boldsymbol{r}$. Following [10], we will use expressions as $\boldsymbol{r}' = \boldsymbol{r} - \beta\boldsymbol{c} \perp \tilde{\boldsymbol{r}}$ not only to define the updated vector $\boldsymbol{r}'$, but also to define the scalar β.

The recurrences (1) and (2), and the orthogonality (3) can be expressed by

$$\boldsymbol{r}_{k+1}^{\text{bcg}} = \boldsymbol{r}_k^{\text{bcg}} - \alpha_k \mathbf{A}\boldsymbol{u}_k^{\text{bcg}} \perp \tilde{\boldsymbol{r}}_k, \tag{16}$$

$$\mathbf{A}\boldsymbol{u}_{k+1}^{\text{bcg}} = \mathbf{A}\boldsymbol{r}_{k+1}^{\text{bcg}} - \beta_k \mathbf{A}\boldsymbol{u}_k^{\text{bcg}} \perp \tilde{\boldsymbol{r}}_k, \tag{17}$$

$$\boldsymbol{u}_{k+1}^{\text{bcg}} = \boldsymbol{r}_{k+1}^{\text{bcg}} - \beta_k \boldsymbol{u}_k^{\text{bcg}}. \tag{18}$$

Note that $\mathbf{A}\boldsymbol{u}_{k+1}^{\text{bcg}}$ is obtained here by a vector update, whereas in the standard Bi-CG approach (18) is used to compute $\boldsymbol{u}_{k+1}^{\text{bcg}}$ and then $\mathbf{A}\boldsymbol{u}_{k+1}^{\text{bcg}}$ is obtained by multiplying $\boldsymbol{u}_{k+1}^{\text{bcg}}$ by $\mathbf{A}$. Hence, this modified approach requires an additional vector update. However, the computation of the Bi-CG coefficient β_k is more directly now and appears to lead to better stability.

Since $\tilde{\boldsymbol{r}}_k$ can be expressed by $\bar{P}_k(\mathbf{A}^*)\tilde{\boldsymbol{r}}_0$, (16)–(18) can be converted to ([10])

$$\mathbf{P}_k\boldsymbol{r}_{k+1}^{\text{bcg}} = \mathbf{P}_k\boldsymbol{r}_k^{\text{bcg}} - \alpha_k \mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\text{bcg}} \perp \tilde{\boldsymbol{r}}_0, \tag{19}$$

$$\mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\text{bcg}} = \mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\text{bcg}} - \beta_k \mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\text{bcg}} \perp \tilde{\boldsymbol{r}}_0, \tag{20}$$

$$\mathbf{P}_k\boldsymbol{u}_{k+1}^{\text{bcg}} = \mathbf{P}_k\boldsymbol{r}_{k+1}^{\text{bcg}} - \beta_k \mathbf{P}_k\boldsymbol{u}_k^{\text{bcg}} \tag{21}$$

with, as before, $\mathbf{P}_k$ shorthand for $P_k(\mathbf{A})$. These update formulas require the vectors $\mathbf{P}_k\boldsymbol{r}_k^{\text{bcg}}$, $\mathbf{P}_k\boldsymbol{u}_k^{\text{bcg}}$ and $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\text{bcg}}$ as input. Note that only one matrix vector multiplication (MV) is required, namely one to form $\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\text{bcg}}$ in (20): the vector $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\text{bcg}}$ is obtained by a vector update.

To obtain the next GPBiCG residual $\mathbf{r}_{k+1} = \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\text{bcg}}$, the recurrence (4) can be used:

$$\mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\text{bcg}} = (1+\eta_k)\mathbf{P}_k\boldsymbol{r}_{k+1}^{\text{bcg}} - \zeta_k \mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\text{bcg}} - \eta_k \mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\text{bcg}}, \tag{22}$$

with ζ_k and η_k scalars to be specified later. Note, however, that this formula requires $\mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\text{bcg}}$ which is not available yet. Multiplying (1) by the matrix polynomial $\mathbf{P}_{k-1}$ gives

$$\mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\text{bcg}} = \mathbf{P}_{k-1}\boldsymbol{r}_k^{\text{bcg}} - \alpha_k \mathbf{A}\mathbf{P}_{k-1}\boldsymbol{u}_k^{\text{bcg}}. \tag{23}$$

Note that both vectors $\mathbf{P}_{k-1}\boldsymbol{r}_k^{\text{bcg}}$ and $\mathbf{A}\mathbf{P}_{k-1}\boldsymbol{u}_k^{\text{bcg}}$ are available as intermediate quantities from the preceding step. Therefore, to complete the update step, we only have to explain how to obtain $\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\text{bcg}}$ and $\mathbf{A}\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\text{bcg}}$.

We can follow the above strategy for obtaining $\mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\text{bcg}}$ also for obtaining $\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\text{bcg}}$: use (4) and multiply (2) by $\mathbf{P}_{k-1}$ to find

$$\mathbf{P}_{k-1}\boldsymbol{u}_{k+1}^{\text{bcg}} = \mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\text{bcg}} - \beta_k \mathbf{P}_{k-1}\boldsymbol{u}_k^{\text{bcg}}, \tag{24}$$

$$\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\text{bcg}} = (1+\eta_k - \zeta_k\mathbf{A})\mathbf{P}_k\boldsymbol{u}_{k+1}^{\text{bcg}} - \eta_k \mathbf{P}_{k-1}\boldsymbol{u}_{k+1}^{\text{bcg}}. \tag{25}$$

Note that all vectors on the right-hand side of (24) and (25) are available from the preceding step.

As an alternative, (2) can be multiplied by $\mathbf{P}_{k+1}$. This requires the vector $\mathbf{P}_{k+1}\boldsymbol{u}_k^{\text{bcg}}$ which can be computed using (4). Hence,

$$\mathbf{P}_{k+1}\boldsymbol{u}_k^{\text{bcg}} = (1+\eta_k - \zeta_k\mathbf{A})\mathbf{P}_k\boldsymbol{u}_k^{\text{bcg}} - \eta_k \mathbf{P}_{k-1}\boldsymbol{u}_k^{\text{bcg}}, \tag{26}$$

$$\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\text{bcg}} = \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\text{bcg}} - \beta_k \mathbf{P}_{k+1}\boldsymbol{u}_k^{\text{bcg}}. \tag{27}$$

Also here, all vectors on the right-hand side are available.

None of the two alternatives for computing $\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\text{bcg}}$ requires an MV or inner products, only vector updates (AXPYs) are required. To obtain $\mathbf{A}\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\text{bcg}}$ we multiply $\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\text{bcg}}$ by $\mathbf{A}$.

Following [3,4], we introduce auxiliary vectors:

$$\mathbf{r}_k \equiv \mathbf{P}_k\boldsymbol{r}_k^{\text{bcg}}, \qquad \mathbf{u}_k \equiv \mathbf{P}_k\boldsymbol{u}_k^{\text{bcg}}, \qquad \mathbf{c}_k \equiv \mathbf{A}\mathbf{u}_k,$$
$$\mathbf{r}_k' \equiv \mathbf{P}_k\boldsymbol{r}_{k+1}^{\text{bcg}}, \qquad \mathbf{u}_k' \equiv \mathbf{P}_k\boldsymbol{u}_{k+1}^{\text{bcg}}, \qquad \mathbf{c}_k' \equiv \mathbf{A}\mathbf{u}_k', \qquad \mathbf{r}_k'' \equiv \mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\text{bcg}},$$

```
Function  $[\zeta,\eta] = \texttt{PolCoef}(\mathbf{r},\mathbf{s},\delta r)$
    $\mu_1 = \delta r^*\delta r,\quad \nu = \delta r^*\mathbf{s},\quad \mu_2 = \mathbf{s}^*\mathbf{s},\quad \omega_1 = \delta r^*\mathbf{r},\quad \omega_2 = \mathbf{s}^*\mathbf{r}$
    $\delta = \mu_1\mu_2 - |\nu|^2,\quad \zeta = (\mu_1\omega_2 - \bar{\nu}\omega_1)/\delta,\quad \eta = (\mu_2\omega_1 - \nu\omega_2)/\delta$
    (If $k = 0$, then $\zeta = \omega_2/\mu_2,\quad \eta = 0$)
```

Algorithm 1. Computation of the ζ and η coefficients of the stabilizing polynomial.

and Eqs. (19)–(21) are rewritten as

$$\mathbf{r}'_k = \mathbf{r}_k - \alpha_k\mathbf{c}_k \perp \tilde{\mathbf{r}}_0, \tag{28}$$

$$\mathbf{c}'_k = \mathbf{A}\mathbf{r}'_k - \beta_k\mathbf{c}_k \perp \tilde{\mathbf{r}}_0, \tag{29}$$

$$\mathbf{u}'_k = \mathbf{r}'_k - \beta_k\mathbf{u}_k. \tag{30}$$

The orthogonality conditions determine the recurrence coefficients α_k and β_k:

$$\alpha_k = \left(\tilde{\mathbf{r}}_0^*\mathbf{r}_k\right)/\sigma, \qquad \beta_k = \left(\tilde{\mathbf{r}}_0^*\mathbf{A}\mathbf{r}'_k\right)/\sigma, \quad \text{with } \sigma = \tilde{\mathbf{r}}_0^*\mathbf{c}_k. \tag{31}$$

Eqs. (22) and (23) read as

$$\mathbf{r}_{k+1} = (1+\eta_k)\mathbf{r}'_k - \zeta_k\mathbf{A}\mathbf{r}'_k - \eta_k\mathbf{r}''_k, \tag{32}$$

$$\mathbf{r}''_k = \mathbf{r}'_{k-1} - \alpha_k\mathbf{c}'_{k-1}, \tag{33}$$

respectively.

The recurrence coefficients ζ_k and η_k can be computed to minimize the residual norm $\|\mathbf{r}_{k+1}\|_2 = \|\mathbf{r}'_k - \zeta_k\mathbf{A}\mathbf{r}'_k - \eta_k(\mathbf{r}''_\mathbf{k} - \mathbf{r}'_k)\|_2$:

$$(\zeta_k, \eta_k)^{\mathrm{T}} = \left(\tilde{\mathbf{Q}}_k^*\tilde{\mathbf{Q}}_k\right)^{-1}\tilde{\mathbf{Q}}_k^*\mathbf{r}'_k, \quad \text{where } \tilde{\mathbf{Q}}_k \equiv \left[\mathbf{A}\mathbf{r}'_k, \mathbf{r}''_k - \mathbf{r}'_k\right]. \tag{34}$$

The algorithm for the computation of the coefficients ζ_k and η_k can be found in Algorithm 1 when expressing $\mathbf{r}'_k$, $\mathbf{A}\mathbf{r}'_k$ and $\mathbf{r}''_k - \mathbf{r}'_k$ as $\mathbf{r}$, $\mathbf{s}$ and δr, respectively.

For the computation of $\mathbf{u}_{k+1}$ we have two alternatives

$$\mathbf{w}_k = \mathbf{r}''_k - \beta_k\mathbf{u}'_{k-1}, \tag{35}$$

$$\mathbf{u}_{k+1} = (1+\eta_k)\mathbf{u}'_k - \zeta_k\mathbf{c}'_k - \eta_k\mathbf{w}_k \tag{36}$$

from (24) and (25), and

$$\mathbf{w}_\mathbf{k} = (1+\eta_k)\mathbf{u}_k - \zeta_k\mathbf{c}_k - \eta_k\mathbf{u}'_{k-1}, \tag{37}$$

$$\mathbf{u}_{k+1} = \mathbf{r}_{k+1} - \beta_k\mathbf{w}_k \tag{38}$$

from (26) and (27) with $\mathbf{w}_k$ appropriate (different) auxiliary vectors.

By viewing $\mathbf{r}'_k$ and $\mathbf{r}''_k$ also as residuals, $\mathbf{r}_k \equiv \boldsymbol{b} - \mathbf{A}\mathbf{x}_k$, $\mathbf{r}'_k \equiv \boldsymbol{b} - \mathbf{A}\mathbf{x}'_k$, and $\mathbf{r}''_k \equiv \boldsymbol{b} - \mathbf{A}\mathbf{x}''_k$, Eqs. (33), (28) and (32) lead to

$$\mathbf{x}''_k = \mathbf{x}'_{k-1} + \alpha_k\mathbf{u}'_{k-1}, \tag{39}$$

$$\mathbf{x}'_k = \mathbf{x}_k + \alpha_k\mathbf{u}_k, \tag{40}$$

$$\mathbf{x}_{k+1} = (1+\eta_k)\mathbf{x}'_k + \zeta_k\mathbf{r}'_k - \eta_k\mathbf{x}''_k, \tag{41}$$

respectively, which allows us to update the approximate solution.

Summarizing, we propose two new variants of GPBiCG. The algorithm stated by (28)–(34), and (37)–(41) is referred to as *variant 2* of GPBiCG. The implementation, which is updated by (35) and (36) instead of (37) and (38), is referred to as *variant 1* of GPBiCG. Our *variant 2* of GPBiCG can be found in Algorithm 2.

Note that the Bi-CG scalars α_k and β_k are such that both the vectors $\mathbf{r}'_k$ and $\mathbf{c}'_k$, formed in (28) and (29), respectively, are orthogonal to $\tilde{\mathbf{r}}_0$. The way β_k and $\mathbf{c}'_k = \mathbf{A}\mathbf{u}'_k$ are computed here differs from those in the variants in the literature. The other recurrences of our variants are the same as those of BiCG $\times$ MR2 except for the ones that form $\mathbf{w}_k$ and $\mathbf{u}_{k+1}$ in our variant 2.

Table 1 summarizes the computational costs and the memory requirements of the original GPBiCG method, BiCG $\times$ MR2, the RG variant of GPBiCG and our proposed variants per iteration. Here, we used the fact that δr_k can be stored at the location of $\mathbf{r}''_k$ and that $\mathbf{r}''_k$ and $\mathbf{x}''_k$ can be stored in the same location as $\mathbf{r}'_{k-1}$ and $\mathbf{x}'_{k-1}$, respectively, and $\mathbf{w}_k$ on the location of $\mathbf{c}_k$. Updates of the form $\alpha\mathbf{u}$ and $\mathbf{r}'' - \mathbf{r}'$ are counted as 0.5 AXPY. For the readability, we did not replace $\mathbf{w}$ and δr in Algorithm 2. Note that the expressions (37) and (38) can be combined to save 0.5 AXPY. The computation of $\|\mathbf{r}_k\|_2$ for the

```
Select an $\mathbf{x}$ and an $\tilde{\mathbf{r}}_0$. Compute $\mathbf{r} = \boldsymbol{b} - \mathbf{A}\mathbf{x}$
$\mathbf{u} = \mathbf{r}, \quad \mathbf{r}' = \mathbf{x}' = \mathbf{u}' = \mathbf{c}' = \mathbf{0}$
while $\|\mathbf{r}_k\|_2 >$ tol do
    $\mathbf{c} = \mathbf{A}\mathbf{u}, \quad \sigma = \tilde{\mathbf{r}}_0^*\mathbf{c}, \quad \alpha = \tilde{\mathbf{r}}_0^*\mathbf{r}/\sigma$
    $\mathbf{r}' = \mathbf{r}' - \alpha\mathbf{c}',$                    $\mathbf{x}' = \mathbf{x}' + \alpha\mathbf{u}'$
    $\mathbf{r} = \mathbf{r} - \alpha\mathbf{c},$                      $\mathbf{x} = \mathbf{x} + \alpha\mathbf{u}$
    $\mathbf{s} = \mathbf{A}\mathbf{r}, \quad \beta = \tilde{\mathbf{r}}_0^*\mathbf{s}/\sigma$
    $\mathbf{c}' = \mathbf{s} - \beta\mathbf{c}$
    $\delta r = \mathbf{r}' - \mathbf{r}$
    $[\zeta, \eta] =$ PolCoef($\mathbf{r}, \mathbf{s}, \delta r$) (see Algorithm 1)
    $\mathbf{r}' = \mathbf{r} - \zeta\mathbf{s} - \eta\delta r,$       $\mathbf{x}' = (1+\eta)\mathbf{x} + \zeta\mathbf{r} - \eta\mathbf{x}'$
    swap($\mathbf{r}, \mathbf{r}'$)                        swap($\mathbf{x}, \mathbf{x}'$)
    $\mathbf{w} = (1+\eta)\mathbf{u} - \zeta\mathbf{c} - \eta\mathbf{u}'$
    $\mathbf{u}' = \mathbf{r}' - \beta\mathbf{u}$
    $\mathbf{u} = \mathbf{r} - \beta\mathbf{w}$
end
```

Algorithm 2. Our proposed variant 2 of GPBiCG.

Table 1
Computational costs and memory requirements of GPBiCG and the variants per iteration.

	MV	Dot	AXPY	Memory
GPBiCG	2	8	14	11
RG variant	2	8	14	12
BiCG × MR2	2	8	15.5	10
Our variant 1	2	9(8)	14.5	10
Our variant 2	2	9(8)	14	10

termination criterion requires 1 Dot per step. $\tilde{\boldsymbol{r}}_0$ has to be stored. We did not count storage for $\boldsymbol{b}$ (which will be needed if the norm $\|\boldsymbol{b} - \mathbf{A}\mathbf{x}_k\|_2$ of the true residual is to be computed at termination).

Note that, since $\mathbf{r}_{k+1} = \mathbf{r}'_k - \zeta_k\mathbf{A}\mathbf{r}'_k - \eta_k(\mathbf{r}''_k - \mathbf{r}'_k)$ and $\mathbf{r}'_k, (\mathbf{r}''_k - \mathbf{r}'_k) \perp \tilde{\mathbf{r}}_0$, we have that $\tilde{\mathbf{r}}_0^*\mathbf{r}_{k+1} = -\zeta_k\tilde{\mathbf{r}}_0^*(\mathbf{A}\mathbf{r}'_k)$. This relation saves the computation of one inner product per step.

Moreover, we will derive variants of GPBiCG by combining the recurrence of Bi-CG and the coupled two-term recurrences of Rutishauser in Appendix A.

4. Toward more accurate Bi-CG coefficients

Specifically in cases where the convergence has a long stagnation phase, the convergence seems to rely completely on the underlying Bi-CG process. In such cases, it is important to have Bi-CG coefficients that are as accurate as possible. One of the ingredients that determines the accuracy of the Bi-CG coefficients is the accuracy with which the inner products are computed. For, for instance, α_{k+1}, the inner product $\tilde{\mathbf{r}}_0^*\mathbf{r}_{k+1}$ has to be computed. We first focus on this inner product. At the end of this section we comment on the accuracy of the other two inner products that form the Bi-CG coefficients α_k and β_k. Other aspects of effects of rounding errors have been analyzed in [6].

In our analysis, we follow [11], and take $\frac{|\mathbf{y}^*\mathbf{x}|}{\|\mathbf{y}\|\|\mathbf{x}\|}$, the cosine of the angle between $\mathbf{x}$ and $\mathbf{y}$, as a measure for the loss of accuracy in the inner product between $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{x}$ and $\mathbf{y}$ are n-vectors. In our experiments, we display the more precise quantity $\frac{|\mathbf{y}^*\mathbf{x}|}{|\mathbf{y}|^*|\mathbf{x}|}$. We apply the measure to the computation of $\tilde{\mathbf{r}}_0^*\mathbf{r}_k$. The Bi-CG coefficient α_k may not be expected to have any digit of accuracy if the cosine of the angle between $\tilde{\mathbf{r}}_0$ and $\mathbf{r}_k$,

$$\frac{|\tilde{\mathbf{r}}_0^*\mathbf{r}_k|}{\|\tilde{\mathbf{r}}_0\|\,\|\mathbf{r}_k\|}, \tag{42}$$

is less than machine precision (see [11]). Then, the computations relying on the underlying Bi-CG process do not have any meaning and, in practice, it is observed that the convergence stagnates from that point on (see Fig. 1 in Section 5). To avoid this situation, we are interested in a strategy that produces residuals $\mathbf{r}_k$ with small norm for which also the cosine in (42) is "as large as possible": this strategy aims for stabilizing the computation of the Bi-CG coefficients.

The residual $\mathbf{r}_k$ depends on P_k: $\mathbf{r}_k = \mathbf{P}_k\boldsymbol{r}_k^{\text{bcg}}$. Except for the initialization, we cannot control the Bi-CG residuals $\boldsymbol{r}_k^{\text{bcg}}$, but, by selecting ζ_k and η_k, we have some control on the polynomial P_k. For the smallest effect of rounding errors in the inner product, it is desirable to have a polynomial P_k for which the quantity in (42) is as large as possible. If θ_k is the leading coefficient of the polynomial P_k, i.e., $P_k(\lambda) - \theta_k\lambda^k$ is of degree $< k$, then $|\tilde{\mathbf{r}}_0^*\mathbf{r}_k| = \theta_k|\tilde{\mathbf{r}}_0^*\mathbf{A}^k\mathbf{r}_0|$. Note that $|\tilde{\mathbf{r}}_0^*\mathbf{A}^k\mathbf{r}_0|$ is independent of P_k. Therefore, the largest quantity in (42) is obtained by the polynomial P_k for which

$$\frac{1}{|\theta_k|}\|\mathbf{r}_k\| \tag{43}$$

is as small as possible. For computational reasons, we want to determine the polynomial P_k in this paper by a three term recurrence relation. But even for polynomials that can be obtained in this way, it is impossible to determine the polynomial P_k, i.e., the ζ_j and η_j ($j \leqslant k$), for which the scaled residual norm in (43) is minimal. This is for the same reasons as for which it is impossible to find the polynomial P_k for which the residual norm $\|\mathbf{r}_k\|$ is minimal.

As an alternative, we will follow the same strategy as for finding residuals with small norm. We simply minimize the scaled residual norm (43) 'locally' and hope that it also has a good global effect: we hope to reduce the negative effects of rounding errors in the inner products by selecting ζ_k and η_k such that

$$\frac{1}{|\zeta_k|}\left\|\mathbf{r}'_k - \zeta_k \mathbf{A}\mathbf{r}'_k - \eta_k \delta r_k\right\|$$

is as small as possible, or, equivalently,

$$\mathbf{r}_{k+1} = \mathbf{r}'_k - \zeta_k \mathbf{A}\mathbf{r}'_k - \eta_k \delta r_k \perp \mathbf{r}'_k, \delta r_k. \tag{44}$$

Here, we used that

$$(\alpha, \beta) = \operatorname*{arg\,min}_{\tilde{\alpha}, \tilde{\beta}} \|\boldsymbol{x} - \tilde{\alpha}\boldsymbol{a} - \tilde{\beta}\boldsymbol{b}\| \quad \text{if and only if} \quad \boldsymbol{x} - \alpha\boldsymbol{a} - \beta\boldsymbol{b} \perp \boldsymbol{a}, \boldsymbol{b}, \tag{45}$$

which we applied with $\boldsymbol{x} = \mathbf{A}\mathbf{r}'_k$, $\boldsymbol{a} = \mathbf{r}'_k$, $\boldsymbol{b} = \delta r_k$, $\alpha = 1/\zeta_k$ and $\beta = -\eta_k/\zeta_k$. The vectors $\mathbf{r}'_k$ and $\mathbf{r}''_k$ are available and $\delta r_k \equiv \mathbf{r}''_k - \mathbf{r}'_k$. To compare with the standard approach, recall that for (local) minimal residual norm, the ζ_k and η_k are such that

$$\mathbf{r}_{k+1} = \mathbf{r}'_k - \zeta_k \mathbf{A}\mathbf{r}'_k - \eta_k \delta r_k \perp \mathbf{A}\mathbf{r}'_k, \delta r_k \tag{46}$$

(now use (45) with $\boldsymbol{x} = \mathbf{r}'_k$, $\boldsymbol{a} = \mathbf{A}\mathbf{r}'_k$ and $\boldsymbol{b} = \delta r_k$).

To facilitate computation and selection, we first orthogonalize against δr_k since this is required for both (44) and (46).

$$\tilde{\mathbf{r}} \equiv \mathbf{r}'_k - \gamma_1 \delta r_k \quad \text{with } \gamma_1 \equiv \frac{\delta r_k{}^* \mathbf{r}'_k}{\delta r_k{}^* \delta r_k} \tag{47}$$

and

$$\tilde{\mathbf{s}} \equiv \mathbf{A}\mathbf{r}'_k - \gamma_2 \delta r_k \quad \text{with } \gamma_2 \equiv \frac{\delta r_k{}^* \mathbf{A}\mathbf{r}'_k}{\delta r_k{}^* \delta r_k}. \tag{48}$$

Now, with $\eta_k = \gamma_1 - \zeta_k \gamma_2$, (44) is equivalent to

$$\mathbf{r}_{k+1} = \tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}} \perp \tilde{\mathbf{r}}, \tag{49}$$

while (46) is equivalent to

$$\mathbf{r}_{k+1} = \tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}} \perp \tilde{\mathbf{s}}. \tag{50}$$

With

$$c \equiv \frac{\tilde{\mathbf{s}}^* \tilde{\mathbf{r}}}{\|\tilde{\mathbf{s}}\|\,\|\tilde{\mathbf{r}}\|}, \qquad s \equiv \sqrt{1 - |c|^2}, \qquad t \equiv \frac{s}{|c|},$$

$\zeta_k = c\,\frac{\|\tilde{\mathbf{r}}\|}{\|\tilde{\mathbf{s}}\|}$ solves (46), and $\|\tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}}\| = s\,\|\tilde{\mathbf{r}}\|$, $\frac{1}{|\zeta_k|}\|\tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}}\| = t\,\|\tilde{\mathbf{s}}\|$, while $\zeta_k = \frac{1}{c}\frac{\|\tilde{\mathbf{r}}\|}{\|\tilde{\mathbf{s}}\|}$ solves (44), and then $\|\tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}}\| = t\|\tilde{\mathbf{r}}\|$, $\frac{1}{|\zeta_k|}\|\tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}}\| = s\|\tilde{\mathbf{s}}\|$.

If $|c|$ is close to 1, then $t \approx s$ and there is not much difference between the two choices. If $|c| > \frac{1}{2}\sqrt{2}$, then $s < \frac{1}{2}\sqrt{2}$ and $t < 1$: neither of the choices lead to amplification of the residual norm nor of inaccuracy in the inner product. If $|c|$ is small, then t is large and the first choice may lead to large inaccuracies in the inner product, whereas the second choice may lead to large amplification of the residual norm. We want to avoid both situations.

If $|c| \leqslant \Omega$, with, say, $\Omega = \frac{1}{2}\sqrt{2}$, then the alternative choice $\zeta_k = \frac{c}{|c|}\Omega\|\tilde{\mathbf{r}}\|/\|\tilde{\mathbf{s}}\|$ implies that $\|\tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}}\| = \sqrt{s^2 + (\Omega - |c|)^2}\|\tilde{\mathbf{r}}\| \leqslant \sqrt{1 + \Omega^2}\|\tilde{\mathbf{r}}\|$, $\frac{1}{|\zeta_k|}\|\tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}}\| \leqslant \sqrt{1 + \Omega^{-2}}\|\tilde{\mathbf{s}}\|$. Both quantities, the residual norm as well as the scaled residual norm, still can get amplified, but now only moderately.

We therefore suggest to use

$$\zeta_k = \frac{c}{|c|} \max\bigl(|c|, \Omega\bigr) \frac{\|\tilde{\mathbf{r}}\|}{\|\tilde{\mathbf{s}}\|}.$$

The computational steps for the stabilization strategy for obtaining more accurate Bi-CG coefficients can be found in Algorithm 3.

```
Function  $[\zeta, \eta] = \texttt{PolCoef}(\mathbf{r}, \mathbf{s}, \delta r)$
  If  $k = 0$,  then  $\gamma_1 = \gamma_2 = 0$
  else
    $\mu = \delta r^* \delta r$,  $\nu = \delta r^* \mathbf{s}$,  $\omega = \delta r^* \mathbf{r}$,
    $\gamma_1 = \frac{\omega}{\mu}$,  $\gamma_2 = \frac{\nu}{\mu}$,  $\mathbf{r} = \mathbf{r} - \gamma_1 \delta r$,  $\mathbf{s} = \mathbf{s} - \gamma_2 \delta r$
  end
  $\rho = \frac{\mathbf{s}^* \mathbf{r}}{\|\mathbf{s}\| \, \|\mathbf{r}\|}$,  $\zeta = \frac{\rho}{|\rho|} \max(|\rho|, \Omega) \frac{\|\mathbf{r}\|}{\|\mathbf{s}\|}$,  $\eta = \gamma_1 - \zeta \gamma_2$
```

Algorithm 3. Computation of the ζ and η coefficients of the stabilizing polynomial avoiding inaccurate Bi-CG coefficients.

Note that with $\Omega = 0$, we have the standard local minimal residual approach. The choice $\Omega = \frac{1}{2}\sqrt{2}$ is inspired by the fact that $t > 1$ if c is less than $\frac{1}{2}\sqrt{2}$. However, even if $t > 1$ and $\zeta_k = c\|\tilde{\mathbf{r}}\|/\|\tilde{\mathbf{s}}\|$ (and $c \not\ll 1$), it is not clear how this will affect the scaled residual norm: it also depends on $\|\tilde{\mathbf{s}}\|$ and on the cumulative effect of the local steps. It is impossible to predict what the cumulative effect is. Nevertheless, we have good experience in practice with the value $\Omega = \frac{1}{2}\sqrt{2}$ (see Section 5).

From experiments, we learned that the cumulative effect of the local steps is very important. In practice, we often observed that with the local minimal residual approach (i.e., $\Omega = 0$), the cosine in (42) decreases in a consecutive number of steps. It may decrease to machine precision even if the decrease in all steps is moderate. With the above stabilization strategy (with $\Omega > 0$) of 'locally' avoiding significant amplifications in the scaled residual norm, the decrease of the cosine is much less, and a decrease to machine precision can be avoided more often. This may happen even though the local effects may only be small: a phenomenon that is similar to obtaining a small residual using a local residual approach.

We now comment on the accuracy of the inner products $\tilde{\mathbf{r}}_0^* \mathbf{c}_k$ and $\tilde{\mathbf{r}}_0^* \mathbf{s}_k$. The inaccuracy in computing $\tilde{\mathbf{r}}_0^* \mathbf{c}_k$ is determined by $\|\mathbf{c}_k\|/|\tilde{\mathbf{r}}_0^* \mathbf{c}_k|$ (this quantity should not be large). Use (28) to note that

$$\frac{\|\mathbf{c}_k\|}{|\tilde{\mathbf{r}}_0^* \mathbf{c}_k|} = \frac{\|\alpha_k \mathbf{c}_k\|}{|\tilde{\mathbf{r}}_0^* \mathbf{r}_k|} = \frac{\|\mathbf{r}_k - \mathbf{r}'_k\|}{|\tilde{\mathbf{r}}_0^* \mathbf{r}_k|} \leqslant \frac{\|\mathbf{r}_k\|}{|\tilde{\mathbf{r}}_0^* \mathbf{r}_k|} \left(1 + \frac{\|\mathbf{r}'_k\|}{\|\mathbf{r}_k\|}\right).$$

Our stabilization strategy is directed toward having $\|\mathbf{r}_k\|/|\tilde{\mathbf{r}}_0^* \mathbf{r}_k|$ as small as possible. If the quantity $\|\mathbf{r}'_k\|/\|\mathbf{r}_k\|$ is large, then this is visible in the convergence curve as a peak. Therefore, if the convergence curve does not exhibit a large peak at step k, then we expect that our stabilization strategy also avoids large negative effects of rounding errors in the computation of $\sigma_k = \tilde{\mathbf{r}}_0^* \mathbf{c}_k$. Note that peaks at preceding steps do not have an effect on the accuracy of σ_k.

Note that $\tilde{\mathbf{r}}_0^* \mathbf{s}_k = \tilde{\mathbf{r}}_0^* \tilde{\mathbf{s}}$, because $\tilde{\mathbf{r}}_0 \perp \delta r_k$. If the second inner product for the Bi-CG coefficient β_k is computed as $\tilde{\mathbf{r}}_0^* \tilde{\mathbf{s}}$, then the inaccuracy is determined by $\|\tilde{\mathbf{s}}\|/|\tilde{\mathbf{r}}_0^* \tilde{\mathbf{s}}|$ (this quantity should not be large). Since, by (32), $\mathbf{r}_{k+1} = \tilde{\mathbf{r}} - \zeta_k \tilde{\mathbf{s}}$ (cf. (49) and (50)), we have that

$$\frac{\|\tilde{\mathbf{s}}\|}{|\tilde{\mathbf{r}}_0^* \tilde{\mathbf{s}}|} = \frac{\|\zeta_k \tilde{\mathbf{s}}\|}{|\tilde{\mathbf{r}}_0^* \mathbf{r}_{k+1}|} = \frac{\|\mathbf{r}_{k+1}\|}{|\tilde{\mathbf{r}}_0^* \mathbf{r}_{k+1}|} \frac{\|\zeta_k \tilde{\mathbf{s}}\|}{\|\mathbf{r}_{k+1}\|} \leqslant \frac{\|\mathbf{r}_{k+1}\|}{|\tilde{\mathbf{r}}_0^* \mathbf{r}_{k+1}|} \left(1 + \frac{\|\tilde{\mathbf{r}}\|}{\|\mathbf{r}_{k+1}\|}\right).$$

Hence, the conclusion for the accuracy of computing $\tilde{\mathbf{r}}_0^* \mathbf{s}_k$ is similar to the one for computing σ_k. Note, however that the quantity $\|\tilde{\mathbf{r}}\|/\|\mathbf{r}_{k+1}\| \leqslant \|\mathbf{r}'_k\|/\|\mathbf{r}_{k+1}\|$ (cf. (47)) is large, if the polynomial step at step k is effective in reducing the residual norm (i.e., if $\|\mathbf{r}_{k+1}\| \ll \|\tilde{\mathbf{r}}\|$). If $|c|$ is small (this is the situation where the standard strategy ($\Omega = 0$) is expected to be problematic), then for the choice $\Omega > 0$ the quantity $\|\tilde{\mathbf{r}}\|/\|\mathbf{r}_{k+1}\|$ is moderate (less than $1/\Omega$). If $|c| \approx 1$, then (for $\Omega < 1$) $\|\tilde{\mathbf{r}}\|/\|\mathbf{r}_{k+1}\| = 1/s$ is large. But in this case we expect a significant reduction (by a factor of order t) in $\frac{\|\mathbf{r}_j\|}{|\tilde{\mathbf{r}}_0^* \mathbf{r}_j|}$ when moving from $j = k$ to $j = k + 1$.

5. Numerical experiments

In this section, we present numerical experiments on model problems with nonsymmetric matrices, and reveal the merit of our proposed GPBiCG variants and our stabilization strategy for obtaining more accurate Bi-CG coefficients. The examples show the typical convergence behavior of these variants for situations where the convergence is slow (with a long stagnation phase). In case of fast convergence, all methods converge equally fast with the same accuracy. In other examples for which the convergence of GPBiCG exhibits a long stagnation phase (not reported here), we observed a similar behavior.

Throughout this section numerical calculations were carried out in double-precision floating-point arithmetic on a PC with an Intel Pentium M 2.1 GHz processor equipped with a Matlab 7.1. The iterations start with the zero vector (i.e., $\mathbf{x}_0 = (0, \ldots, 0)^{\mathrm{T}}$). The initial shadow residual $\tilde{\mathbf{r}}_0$ is set to random (with seed 16) vector. The iterations are terminated when the reduction of residual norms (i.e., $\|\mathbf{r}_k\|_2/\|\mathbf{r}_0\|_2$) become $\leqslant 10^{-10}$.

As shown in [8], applying 5-point central differences to the two-dimensional convection–diffusion equation

$$-u_{xx} - u_{yy} + 1000(xu_x + yu_y) + 10u = f(x, y)$$

over the unit square $\Omega = (0, 1) \times (0, 1)$ with the Dirichlet boundary conditions $u|_{\partial u} = 0$ yields a linear system with a nonsymmetric matrix. The mesh size is chosen as 64 $(= M + 1)$ in both directions of Ω, so that the resulting system has an $M^2 \times M^2$ coefficient matrix. The right-hand side vector $\boldsymbol{b}$ is determined by substituting a solution vector $\hat{\boldsymbol{x}} = (1, \ldots, 1)^{\mathrm{T}}$

Table 2
Number of MVs and explicitly computed relative residual norms.

	MVs (ratio)	True res.
Original GPBiCG	n.c. (–)	–
RG variant	n.c. (–)	–
BiCG × MR2	2432 (1.15)	2.1e−09
Our variant 1	2100 (1.00)	4.2e−11
Our variant 2	2640 (1.25)	7.7e−11

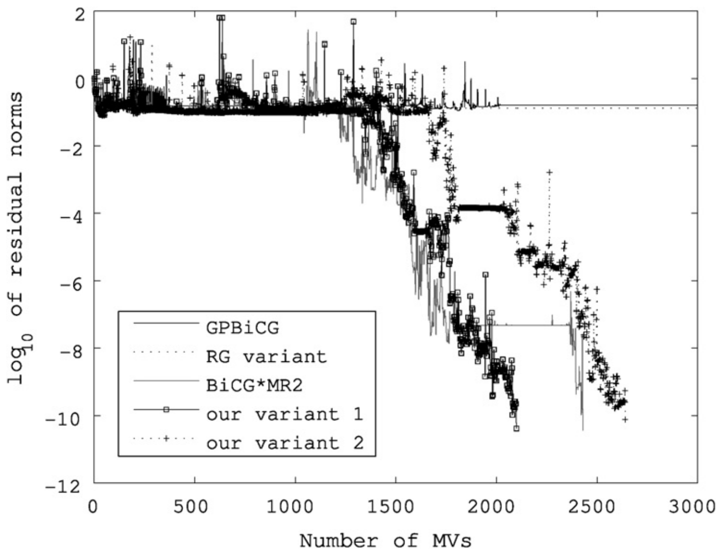


Fig. 1. Convergence history of the original GPBiCG, BiCG × MR2, the RG variant and our variants.

into the equation $\boldsymbol{b} = \mathbf{A}\hat{\boldsymbol{x}}$. The linear system with the coefficient matrix is solved with the original GPBiCG, BiCG × MR2, the RG variant of GPBiCG and our two variants, and then the convergence behaviors among the original GPBiCG, BiCG × MR2, the RG variant of GPBiCG and our variants are compared. Table 2 shows the number of matrix-vector products (*MVs*) and the explicitly computed relative residual 2-norms $\log_{10}(\|\boldsymbol{b} - \mathbf{A}\mathbf{x}_k\|_2/\|\mathbf{r}_0\|_2)$ (abbreviated as *True res.*) at termination. The convergence behaviors are displayed in Fig. 1. The number of MVs (on the horizontal axis) is plotted versus the $\log_{10}$ of the relative residual 2-norms.

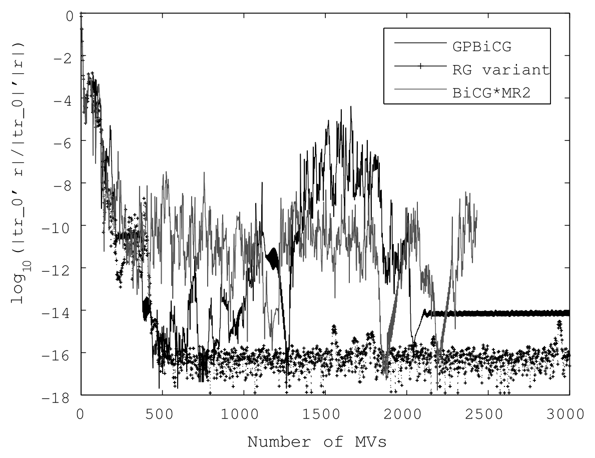
From Table 2 and Fig. 1, we learn that the residual norms of the original GPBiCG and the RG variant of GPBiCG do not converge, i.e., have a long stagnation. Our proposed variants and BiCG × MR2 converge. The number of MVs of our variant 1 is 15% less than that of BiCG × MR2.

We show the quantity $\frac{|\tilde{\mathbf{r}}_0^* \mathbf{r}|}{|\tilde{\mathbf{r}}_0|^*|\mathbf{r}|}$ to examine the loss of accuracy in the inner product between $\tilde{\mathbf{r}}_0$ and $\mathbf{r}$. We display the $\log_{10}$ of this quantity for the original GPBiCG method, BiCG × MR2 and the RG variant of GPBiCG in the left panel of Fig. 2, and the $\log_{10}$ of the quantity $\frac{|\tilde{\mathbf{r}}_0^* \mathbf{s}|}{|\tilde{\mathbf{r}}_0|^*|\mathbf{s}|}$ for our variants in the right panel of Fig. 2, respectively, to show the effect of using the slightly different coupled two-term recurrences for Bi-CG from the standard ones for deriving our variants.

From Fig. 2, we can observe that the quantity calculated by the original GPBiCG, BiCG × MR2 and the RG variant reach less than 10^{-16}, but that done by our variant 1 becomes about 10^{-12}, i.e., three digits are accurate for our variant 1. Some of the Bi-CG coefficients of the original GPBiCG, BiCG × MR2 and the RG variant cannot be expected to have any digit of accuracy.

Next, we apply our proposed stabilization strategy of Algorithm 3 in Section 4 to our variants. The parameter Ω is set to $\sqrt{2}/2$. We display the convergence behaviors in the left panel of Fig. 3. Table 3 shows MVs and the true residual at termination. Moreover, we display the quantity $\frac{|\tilde{\mathbf{r}}_0^* \mathbf{s}|}{|\tilde{\mathbf{r}}_0|^*|\mathbf{s}|}$ for our variants with the stabilization strategy in the right panel of Fig. 3.

From Table 3 and the left panel of Fig. 3, we learn that the residual norms of our variants with the stabilization strategy converge nicely and the number of MVs of our variants with the stabilization strategy is much less than that of our variants without this strategy. From the right panel of Fig. 3, we can observe that the quantity $\frac{|\tilde{\mathbf{r}}_0^* \mathbf{s}|}{|\tilde{\mathbf{r}}_0|^*|\mathbf{s}|}$ calculated by our variants with the stabilization strategy is more than about 10^{-8}. The Bi-CG coefficients of our variants with the stabilization strategy are more accurate than those of our variants without this strategy. Therefore, we may conclude that our proposed stabilization strategy is effective in maintaining sufficiently accurate Bi-CG coefficients, which has a beneficial effect on the convergence. Setting the parameter Ω to $\sqrt{2}/2$ seems to be effective.



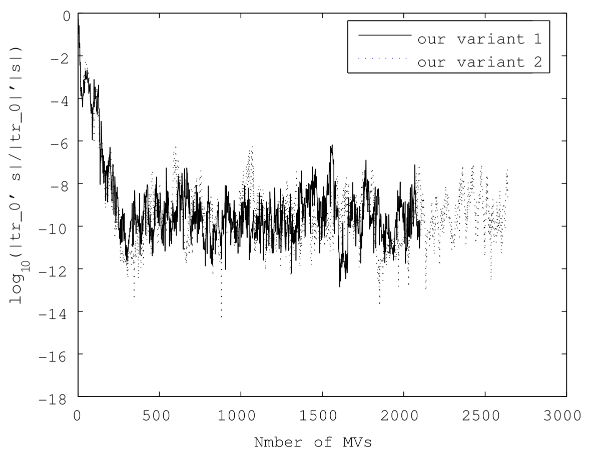
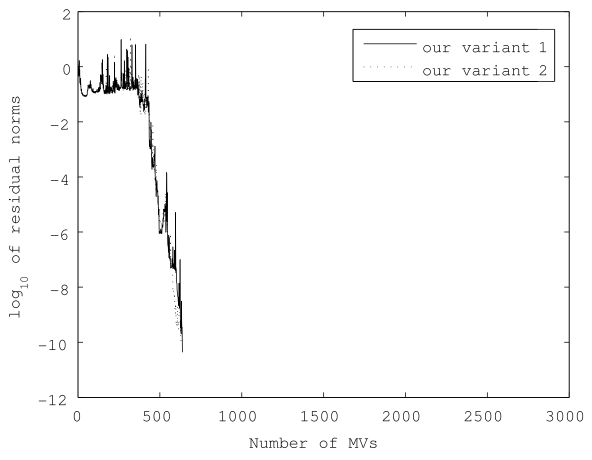


Fig. 2. The $\log_{10}$ of the quantity $\frac{|\tilde{\mathbf{r}}_0^*\mathbf{r}|}{|\tilde{\mathbf{r}}_0|^*|\mathbf{r}|}$ for the original GPBiCG method, BiCG × MR2 and the RG variant (on the left) and the $\log_{10}$ of the quantity $\frac{|\tilde{\mathbf{r}}_0^*\mathbf{s}|}{|\tilde{\mathbf{r}}_0|^*|\mathbf{s}|}$ for our variants (on the right).



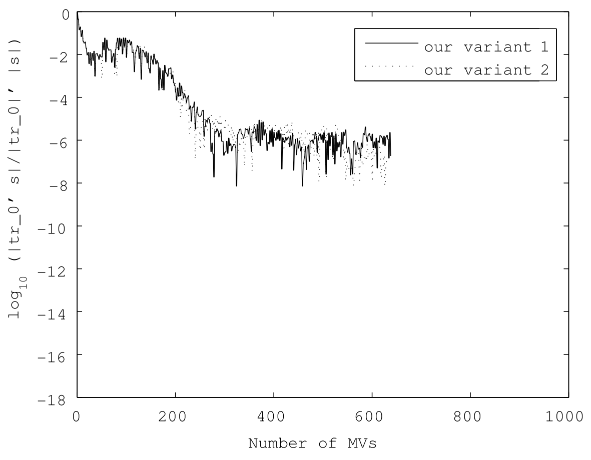


Fig. 3. Convergence history (on the left) and the $\log_{10}$ of the quantity $\frac{|\tilde{\mathbf{r}}_0^*\mathbf{s}|}{|\tilde{\mathbf{r}}_0|^*|\mathbf{s}|}$ (on the right) for our variants with the stabilization strategy.

Table 3
Number of MVs and explicitly computed relative residual norms for our variants with the stabilization strategy.

	MVs (ratio)	True res.
Our variant 1	638 (1.00)	4.3e−11
Our variant 2	630 (1.00)	8.6e−11

Note that our proposed stabilization strategy can also be applied to the original GPBiCG, BiCG × MR2 and the RG variant.

6. Concluding remarks

We designed variants of GPBiCG/BiCG × MR2. The original GPBiCG method and the RG variant of GPBiCG were derived by combining the recurrences of Bi-CG with coupled two-term recurrences for the stabilizing polynomials, while our proposed variants are derived by combining slightly modified recurrences of Bi-CG with three-term recurrences for the stabilizing polynomials. This modification seems to allow a more accurate computation of the Bi-CG coefficients. Although these variants are mathematically equivalent, the computation of one of the Bi-CG coefficients and of the recurrences of our variants are also (partly) different from those of other variants of GPBiCG as BiCG × MR2. Numerical experiments show that our variants are more stable, leading to faster convergence and more accurate results typically in cases where the convergence is slow. Moreover, we proposed a strategy to select the coefficients of the stabilizing polynomials such that Bi-CG coefficients are more accurately computed. Numerical experiments show that GPBiCG with this stabilization strategy is effective.

Appendix A

We also propose variants of GPBiCG by combining the recurrence of Bi-CG and the coupled two-term recurrences of Rutishauser [7].

With $\zeta'_k \equiv \alpha_k$ and $\eta'_k \equiv -\beta_{k-1}\frac{\alpha_k}{\alpha_{k-1}}$, the coupled two-term recurrences for Bi-CG can be rewritten into a three-term recurrence for the Bi-CG residuals:

$$\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \left(1+\eta'_k-\zeta'_k\mathbf{A}\right)\boldsymbol{r}_k^{\mathrm{bcg}} - \eta'_k\boldsymbol{r}_{k-1}^{\mathrm{bcg}},$$

or, in terms of the Bi-CG residual polynomials P'_k,

$$P'_{k+1}(\lambda) = \left(1-\eta'_k-\zeta'_k\lambda\right)P'_k(\lambda) + \eta'_k P'_{k-1}(\lambda).$$

This scaled version of the three-term recurrence for the Lanczos polynomials has been exploited in the ORTHODIR algorithm [5] for symmetric systems: CG can be viewed as a coupled two-term variant of ORTHODIR. To stabilize ORTHODIR, Rutishauser [7] suggested an alternative coupled two-term variant, that in the polynomial version reads

$$\begin{aligned}\tilde{G}'_{k+1}(\lambda) &= \zeta'_k\lambda P'_k(\lambda) + \eta'_k\tilde{G}'_k(\lambda),\\ P'_{k+1}(\lambda) &= P'_k(\lambda) - \tilde{G}'_{k+1}(\lambda).\end{aligned}$$

Applying Rutishauser's suggestion to (4) gives the coupled two-term recurrence

$$\begin{aligned}\tilde{G}_{k+1}(\lambda) &= \zeta_k\lambda P_k(\lambda) + \eta_k\tilde{G}_k(\lambda),\\ P_{k+1}(\lambda) &= P_k(\lambda) - \tilde{G}_{k+1}(\lambda),\end{aligned} \tag{51}$$

where the Bi-CG formulation corresponds to

$$\begin{aligned}\hat{G}_k(\lambda) &= P_k(\lambda) + \eta_k\frac{\zeta_{k-1}}{\zeta_k}\hat{G}_{k-1}(\lambda),\\ P_{k+1}(\lambda) &= P_k(\lambda) - \zeta_k\lambda\hat{G}_k(\lambda).\end{aligned} \tag{52}$$

The G_k of (6) and (7), $\tilde{G}_k$ of (51) and $\hat{G}_k$ of (52) are related as $\tilde{G}_{k+1}(\lambda) = \lambda G_k(\lambda)$ and $G_k = \zeta_k\hat{G}_k$.

In our GPBiCG variant that we will derive here, we will combine the Bi-CG recurrences and the polynomial recurrences (51). Note that the polynomials G_k in the original GPBiCG can be viewed as corresponding to the auxiliary polynomials of Bi-CG that define $\boldsymbol{u}_k^{\mathrm{bcg}}$: the difference is an insignificant scaling. Whereas in our variant the difference, formulated in the vector setting, is substantial: it is a multiplication by $\mathbf{A}$.

Combining (16)–(18) and (51) gives the following update formulas for the residual

$$\begin{array}{ll} & \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}} \perp \tilde{\boldsymbol{r}}_0,\\ & \tilde{\mathbf{G}}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \tilde{\mathbf{G}}_k\boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k\mathbf{A}\tilde{\mathbf{G}}_k\boldsymbol{u}_k^{\mathrm{bcg}},\\ \mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, & \tilde{\mathbf{G}}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \zeta_k\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} + \eta_k\tilde{\mathbf{G}}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}},\\ & \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \tilde{\mathbf{G}}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}.\end{array} \tag{53}$$

The computation of the vector $\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ requires a multiplication by $\mathbf{A}$, which we indicated by putting this vector in the left column. As input, we need the vectors $\mathbf{P}_k\boldsymbol{r}_k^{\mathrm{bcg}}$, $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}$, $\tilde{\mathbf{G}}_k\boldsymbol{r}_k^{\mathrm{bcg}}$, and $\mathbf{A}\tilde{\mathbf{G}}_k\boldsymbol{u}_k^{\mathrm{bcg}}$, and, as we will learn below, also $\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}$. At the end of (53), $\mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ and $\tilde{\mathbf{G}}_{k+1}\mathbf{r}_{k+1}$ are available.

For the update vector $\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}$ we have

$$\begin{array}{ll} & \mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}} \perp \tilde{\boldsymbol{r}}_0,\\ & \mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}},\\ & \tilde{\mathbf{G}}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \tilde{\mathbf{G}}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\tilde{\mathbf{G}}_k\boldsymbol{u}_k^{\mathrm{bcg}},\\ & \tilde{\mathbf{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \zeta_k\mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} + \eta_k\tilde{\mathbf{G}}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}},\\ \mathbf{A}\tilde{\mathbf{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, & \mathbf{A}\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \mathbf{A}\tilde{\mathbf{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}},\\ & \mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \tilde{\mathbf{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}.\end{array} \tag{54}$$

Now the vectors $\mathbf{AP}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, $\mathbf{A\tilde{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, and $\mathbf{P}_{k+1}\mathbf{u}_{k+1}$ are also available for the next step. Note that a multiplication by $\mathbf{A}$ is required to form $\mathbf{A\tilde{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, while, in the first line, $\mathbf{AP}_{k}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ is obtained as a vector update.

The vector $\mathbf{A\tilde{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ can also be computed as

$$\mathbf{AP}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \quad \mathbf{A\tilde{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{AP}_{k}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \mathbf{AP}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}: \tag{55}$$

(55) can replace the last but one line of (54). We expect the vectors $\mathbf{AP}_{k}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ and $\mathbf{AP}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ to be large and $\mathbf{A\tilde{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ to be small. In such case, (55) may not provide an accurate way of computing $\mathbf{A\tilde{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$. We, therefore, favor (54).

As an alternative to the third and fourth line in (54) we can change the order, first the Bi-CG update then the stabilizing polynomial update:

$$\begin{aligned}
&\mathbf{\tilde{G}}_{k+1}\boldsymbol{u}_{k}^{\mathrm{bcg}} = \zeta_k\mathbf{AP}_k\boldsymbol{u}_{k}^{\mathrm{bcg}} + \eta_k\mathbf{\tilde{G}}_k\boldsymbol{u}_{k}^{\mathrm{bcg}},\\
&\mathbf{\tilde{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{\tilde{G}}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{\tilde{G}}_{k+1}\boldsymbol{u}_{k}^{\mathrm{bcg}}.
\end{aligned}$$

We refer to this variant as variant 4 and to (54) as variant 3.

Introducing the vectors and auxiliary vectors

$$\begin{aligned}
&\mathbf{r}_k \equiv \mathbf{P}_k\boldsymbol{r}_k^{\mathrm{bcg}}, && \mathbf{r}'_k \equiv \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, && \delta r_k \equiv \mathbf{\tilde{G}}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, && \delta r'_k \equiv \mathbf{\tilde{G}}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}},\\
&\mathbf{u}_k \equiv \mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}, && \mathbf{u}'_k \equiv \mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, && \delta u_k \equiv \mathbf{\tilde{G}}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, && \delta u'_k \equiv \mathbf{\tilde{G}}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}},\\
&\mathbf{c}_k \equiv \mathbf{A}\mathbf{u}_k, && \mathbf{c}'_k \equiv \mathbf{A}\mathbf{u}'_k, && \mathbf{s}_k \equiv \mathbf{A}\mathbf{r}'_k, && \delta c_k \equiv \mathbf{A}\delta u_k,
\end{aligned}$$

these update formulas read as

$$\begin{array}{lll}
 & \mathbf{r}'_k = \mathbf{r}_k - \alpha_k\mathbf{c}_k \perp \tilde{\mathbf{r}}_0, & \mathbf{x}'_k = \mathbf{x}_k + \alpha_k\mathbf{u}_k,\\
 & \delta r'_k = \delta r_{k-1} - \alpha_k\delta c_{k-1}, & \delta x'_k = \delta x_{k-1} + \alpha_k\delta u_{k-1},\\
\mathbf{s}_k = \mathbf{A}\mathbf{r}'_k, & \delta r_k = \zeta_k\mathbf{s}_k + \eta_k\delta r'_k, & \delta x_k = -\zeta_k\mathbf{r}'_k + \eta_k\delta x'_k,\\
 & \mathbf{r}_{k+1} = \mathbf{r}'_k - \delta r_k, & \mathbf{x}_{k+1} = \mathbf{x}'_k - \delta x_k,
\end{array} \tag{56}$$

and (with variant 3 on the left side and variant 4 on the right side)

$$\begin{array}{lll}
 & \mathbf{c}'_k = \mathbf{s}_k - \beta_k\mathbf{c}_k \perp \tilde{\mathbf{r}}_0, & \\
 & \mathbf{u}'_k = \mathbf{r}'_k - \beta_k\mathbf{u}_k, & \\
 & \delta u'_k = \delta r'_k - \beta_k\delta u_{k-1}, & \mathbf{w}_k = \zeta_k\mathbf{c}_k + \eta_k\delta u_{k-1},\\
 & \delta u_k = \zeta_k\mathbf{c}'_k + \eta_k\delta u'_k, & \delta u_k = \delta r_k - \beta_k\mathbf{w}_k,\\
\delta c_k = \mathbf{A}\delta u_k, & \mathbf{c}_{k+1} = \mathbf{c}'_k - \delta c_k, & \\
 & \mathbf{u}_{k+1} = \mathbf{u}'_k - \delta u_k. &
\end{array} \tag{57}$$

Note that the computation of β_k requires $\mathbf{s}_k$ which is available after the first two lines of (56). Then, for variant 3, the other lines of (57) can be performed followed by the remaining two lines of (56). Variant 4 requires δr_k. For this variant, computation of (57) can be performed after finishing the first three updates in (56).

The update formulas for the approximate solution in (56) have been obtained from the ones for the residual using that $\mathbf{r}_k = \boldsymbol{b} - \mathbf{A}\mathbf{x}_k$, $\mathbf{r}'_k = \boldsymbol{b} - \mathbf{A}\mathbf{x}'_k$, $\delta r_k = -\mathbf{A}\delta x_k$, and $\delta r'_k = -\mathbf{A}\delta x'_k$.

As before, the ζ_k and η_k on the third line of (56) can be computed to minimize $\|\mathbf{r}_{k+1}\|_2$: $(\zeta_k, \eta_k)^T = (\tilde{\mathbf{Q}}^*\tilde{\mathbf{Q}})^{-1}(\tilde{\mathbf{Q}}^*\mathbf{r}'_k)$, where $\tilde{\mathbf{Q}} = [\mathbf{s}_k, \delta r'_k]$.

In variant 4, δu_k can be computed with explicitly computing $\mathbf{w}_k$:

$$\delta u_k = \delta r_k - (\beta_k\zeta_k)\mathbf{c}_k - (\beta_k\eta_k)\delta u_{k-1}.$$

This saves 0.5 AXPY.

Note that $\mathbf{r}_k$, $\mathbf{r}'_k$ and $\mathbf{r}_{k+1}$ can be stored at the same location. Similarly δr_{k-1}, $\delta r'_k$ and δr_k can be stored at the same location. A similar observation applies to the u-vectors and the x-vectors. When properly rearranged, the vectors δc_{k-1} can be replaced by, subsequently, $\mathbf{s}_k$, while $\mathbf{c}_k$ can be replaced by $\mathbf{c}'_k$ and $\mathbf{c}_{k+1}$. Therefore, 9 vectors have to be stored (including $\tilde{\mathbf{r}}_0$). Algorithms 4 and 5 incorporates these observation. For the readability, we did not replace δc by $\mathbf{s}$, but the reader can easily check that such a change does not affect the algorithm.

Note that, since $\mathbf{r}_{k+1} = \mathbf{r}'_k - \zeta_k\mathbf{s}_k - \eta_k\delta r'_k$ and $\mathbf{r}'_k, \delta r'_k \perp \tilde{\mathbf{r}}_0$, we have that $\tilde{\mathbf{r}}_0^*\mathbf{r}_{k+1} = -\zeta_k\tilde{\mathbf{r}}_0^*\mathbf{s}_k$. This relation saves the computation of one inner product per step.

```
Select an $\mathbf{x}$ and an $\tilde{\boldsymbol{r}}_0$. Compute $\mathbf{r} = \boldsymbol{b} - \mathbf{A}\mathbf{x}$
$\mathbf{u} = \mathbf{r}$,   $\mathbf{c} = \mathbf{A}\mathbf{r}$,   $\delta r = \delta c = \delta x = \delta u = \mathbf{0}$
while $\|\mathbf{r}\|_2 >$ tol do
    $\sigma = \tilde{\mathbf{r}}_0^*\mathbf{c}$,   $\alpha = \tilde{\mathbf{r}}_0^*\mathbf{r}/\sigma$
    $\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}$,                                   $\mathbf{x} = \mathbf{x} + \alpha\mathbf{u}$
    $\delta r = \delta r - \alpha\delta c$,                              $\delta x = \delta x + \alpha\delta u$
    $\mathbf{s} = \mathbf{A}\mathbf{r}$,   $[\zeta, \eta] =$ PolCoef($\mathbf{r}, \mathbf{s}, \delta r$) (see Algorithm 1)
    $\delta r = \zeta\mathbf{s} + \eta\delta r$,                           $\delta x = -\zeta\mathbf{r} + \eta\delta x$
    $\beta = \tilde{\mathbf{r}}_0^*\mathbf{s}/\sigma$,
    $\delta u = \delta r - (\beta\eta)\delta u - (\beta\zeta)\mathbf{c}$,
    $\mathbf{c} = \mathbf{s} - \beta\mathbf{c}$,   $\mathbf{u} = \mathbf{r} - \beta\mathbf{u}$
    $\delta c = \mathbf{A}\delta u$,   $\mathbf{c} = \mathbf{c} - \delta c$,   $\mathbf{u} = \mathbf{u} - \delta u$,
    $\mathbf{r} = \mathbf{r} - \delta r$,                                   $\mathbf{x} = \mathbf{x} - \delta x$,
end
```

Algorithm 4. Our proposed variant 4 of GPBiCG.

```
Select an $\mathbf{x}$ and an $\tilde{\boldsymbol{r}}_0$. Compute $\mathbf{r} = \boldsymbol{b} - \mathbf{A}\mathbf{x}$
$\mathbf{u} = \mathbf{r}$,   $\mathbf{c} = \mathbf{A}\mathbf{r}$,   $\delta r = \delta c = \delta x = \delta u = \mathbf{0}$
while $\|\mathbf{r}\|_2 >$ tol do
    $\sigma = \tilde{\mathbf{r}}_0^*\mathbf{c}$,   $\alpha = \tilde{\mathbf{r}}_0^*\mathbf{r}/\sigma$
    $\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}$,                                   $\mathbf{x} = \mathbf{x} + \alpha\mathbf{u}$
    $\delta r = \delta r - \alpha\delta c$,                              $\delta x = \delta x + \alpha\delta u$
    $\mathbf{s} = \mathbf{A}\mathbf{r}$,   $[\zeta, \eta] =$ PolCoef($\mathbf{r}, \mathbf{s}, \delta r$) (see Algorithm 1)
    $\beta = \tilde{\mathbf{r}}_0^*\mathbf{s}/\sigma$,
    $\mathbf{c} = \mathbf{s} - \beta\mathbf{c}$,   $\mathbf{u} = \mathbf{r} - \beta\mathbf{u}$,   $\delta u = \delta r - \beta\delta u$,
    $\delta u = \zeta\mathbf{c} + \eta\delta u$,   $\delta r = \zeta\mathbf{s} + \eta\delta r$,      $\delta x = -\zeta\mathbf{r} + \eta\delta x$
    $\mathbf{r} = \mathbf{r} - \delta r$,                                   $\mathbf{x} = \mathbf{x} - \delta x$,
    $\delta c = \mathbf{A}\delta u$,   $\mathbf{c} = \mathbf{c} - \delta c$,   $\mathbf{u} = \mathbf{u} - \delta u$,
end
```

Algorithm 5. Our proposed variant 3 of GPBiCG.

Acknowledgements

This research was partly supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), (B) and on Innovative Areas, 22560067, 21300017 and 22104004, 2010.

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, Lecture Notes in Math. 506 (1976) 73–89.
[3] M.H. Gutknecht, Variants of BiCGStab for matrices with complex spectrum, SIAM J. Sci. Comput. 14 (1993) 1020–1033.
[4] M.H. Gutknecht, Lanczos-type solvers for nonsymmetric linear systems of equations, Acta Numer. 6 (1997) 217–397.
[5] K.C. Jea, D.M. Young, Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods, Linear Algebra Appl. 34 (1980) 159–194.
[6] S. Röllin, M.H. Gutknecht, Variations of Zhang's Lanczos-type product method, Appl. Numer. Math. 41 (2002) 119–133.
[7] H. Rutishauser, Theory of gradient method, in: Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Value Problems, in: Mitt. Inst. Angew. Math. ETH Zürich, Birkhäuser, Basel, 1959, pp. 24–49.
[8] Y. Saad, A flexible inner–outer preconditioned GMRES algorithm, SIAM J. Sci. Stat. Comput. 14 (1993) 461–469.
[9] G.L.G. Sleijpen, D.R. Fokkema, BiCGstab(l) for solving linear equations involving unsymmetric matrices with complex spectrum, ETNA 1 (1993) 11–31.
[10] G.L.G. Sleijpen, P. Sonneveld, M.B. van Gijzen, Bi-CGSTAB as induced dimension reduction method, Appl. Numer. Math. 60 (2010) 1100–1114.
[11] G.L.G. Sleijpen, H.A. van der Vorst, Maintaining convergence properties of BiCGstab methods in finite precision arithmetic, Numer. Algorithms 10 (1995) 203–223.
[12] P. Sonneveld, CGS, A fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Comput. 10 (1989) 36–52.
[13] E.L. Stiefel, Kernel polynomial in linear algebra and their numerical applications, in: Further Contributions to the Determination of Eigenvalues, NBS Appl. Math. Ser. 49 (1958) 1–22.
[14] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 13 (1992) 631–644.
[15] S.L. Zhang, GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 18 (1997) 537–551.