Applied Mathematics and Computation 218 (2012) 10889–10899

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Hybrid Bi-CG methods with a Bi-CG formulation closer to the IDR approach

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

Keywords:
Linear systems
Krylov subspace method
Bi-CG
Hybrid Bi-CG method
Lanczos-type method
Induced Dimension Reduction (IDR) method

ABSTRACT

The Induced Dimension Reduction(s) (IDR(s)) method has recently been developed. Sleijpen et al. have reformulated the Bi-Conjugate Gradient STABilized (BiCGSTAB) method to clarify the relationship between BiCGSTAB and IDR(s). The formulation of Bi-Conjugate Gradient (Bi-CG) part used in the reformulated BiCGSTAB is different from that of the original Bi-CG method; the Bi-CG coefficients are computed by a formulation that is closer to the IDR approach. In this paper, we will redesign variants of the Conjugate Gradient Squared method (CGS) method, BiCGSTAB and the Generalized Product-type method derived from Bi-CG (GPBiCG)/BiCG×MR2 by using the Bi-CG formulation that is closer to the IDR approach. Although our proposed variants are mathematically equivalent to their counterparts, the computation of one of the Bi-CG coefficients differs, and the recurrences of the variants are also partly different from those of the original hybrid Bi-CG methods. Numerical experiments show that the variants of BiCGSTAB and GPBiCG/BiCG×MR2 are more stable and lead to faster convergence typically for linear systems for which the methods converge slowly (long stagnation phase), and that the variants of CGS attain more accurate approximate solutions.

1. Introduction

We treat Krylov subspace method for solving a large sparse linear system

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

for $\boldsymbol{x}$, where $\mathbf{A}$ is a given n-by-n matrix, and $\boldsymbol{b}$ is a given n-vector. The Bi-Conjugate Gradient (Bi-CG) method [4] is well-known for solving the linear system. A number of hybrid Bi-CG methods such as Conjugate Gradient Squared (CGS) [16], Bi-CG STABilized (BiCGSTAB) [19], BiCGStab2 [7], BiCGstab(ℓ) [12], Generalized Product-type Bi-CG (GPBiCG) [20], BiCG×MR2 [8,2], and Generalized CGS (GCGS) [5] have been developed to improve the convergence of Bi-CG and to avoid multiplication by the conjugate transpose $\mathbf{A}^*$ of $\mathbf{A}$. The residuals of the hybrid Bi-CG method are expressed as the product of the residual of Bi-CG and a stabilization polynomial ([14]). The Induced Dimension Reduction(s) (IDR(s)) method [17] has recently been proposed, and it has been reported that IDR(s) is often more effective than the hybrid Bi-CG methods. IDR(s) can be considered as a block version of BiCGSTAB, and IDR(s) for $s = 1$ is equivalent to BiCGSTAB [14,13]. Sleijpen et al. have reformulated BiCGSTAB to clarify the relationship between BiCGSTAB and IDR(s) in [14,13]. The formulation of Bi-CG part used in the reformulated BiCGSTAB is different from the standard Bi-CG; the Bi-CG coefficients are computed by using the formulation that is closer to the IDR approach, which is referred to as a *Bi-CG*$_{(IDR)}$ formulation.

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

0096-3003/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2012.04.049

In this paper, therefore, we will redesign variants of CGS, BiCGSTAB and GPBiCG/BiCG×MR2 by using the *Bi-CG*$_{(IDR)}$ formulation, i.e., the formulation of Bi-CG described in [14,13]. Following [1], we derive four variants of GPBiCG/BiCG×MR2 by combining the *Bi-CG*$_{(IDR)}$ formulation and a three-term recurrence or coupled two-term recurrences for the stabilization polynomial since it has not been remarked in [1] that GPBiCG/BiCG×MR2 can profit from inside IDR. Two variants of BiCGSTAB are given by combining the *Bi-CG*$_{(IDR)}$ formulation and a two-term recurrence for the stabilization polynomial. One of our variants of BiCGSTAB coincides with the reformulated BiCGSTAB described in [14,13], but its advantage has not previously been described. We design two variants of CGS by combining the *Bi-CG*$_{(IDR)}$ formulation and the coupled two-term recurrences for Lanczos polynomials [18] with the same analogy as was used in GCGS. Although these variants are mathematically equivalent to their counterparts, the computation of one of the Bi-CG coefficients differs, and the recurrences of the variants are also partly different from those of the original hybrid Bi-CG methods. This modification leads to slightly more accurate Bi-CG coefficients and faster convergence for the variants of BiCGSTAB and GPBiCG/BiCG×MR2, and more accurate approximate solutions for the variants of CGS.

We compare the convergence between the original CGS, BiCGSTAB, GPBiCG and BiCG×MR2 methods, and our proposed variants. Numerical experiments show that the variants of BiCGSTAB and GPBiCG/BiCG×MR2 are more stable and lead to faster convergence typically for linear systems for which the methods converge slowly (long stagnation phase), and that the approximate solutions solved by the variants of CGS are more accurate. Following [15], we also examine the accuracy for the computation of an inner product computed in the Bi-CG coefficients.

We first outline the standard Bi-CG method and the *Bi-CG*$_{(IDR)}$ formulation in Section 2. In Section 3, we derive alternative implementations of the original CGS, BiCGSTAB and GPBiCG/BiCG×MR2. Numerical experiments in Section 4 show that the variants are more effective. The results imply that the hybrid Bi-CG methods profit from inside IDR.

2. Standard Bi-CG and an alternative Bi-CG

In this section, we describe the formulation of the standard Bi-CG method and the *Bi-CG*$_{(IDR)}$ formulation used in the reformulated BiCGSTAB.

The residuals $\boldsymbol{r}_k^{\mathrm{bcg}}$ of the standard Bi-CG method are generated by the coupled 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 Bi-CG coefficients α_k and β_k are determined 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, and the n-vectors $\tilde{\boldsymbol{r}}_0, \tilde{\boldsymbol{r}}_1, \ldots, \tilde{\mathrm{r}}_{k-1}$ are such that they span the kth Krylov subspace $\mathscr{K}_k(\boldsymbol{A}^*; \tilde{\boldsymbol{r}}_0)$ for each k. An induction argument shows that $\boldsymbol{r}_k^{\mathrm{bcg}} \perp \mathscr{K}_k(\boldsymbol{A}^*; \tilde{\boldsymbol{r}}_0)$.

The residuals $\boldsymbol{r}_k$ of the hybrid Bi-CG method can be expressed as

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

by combining a polynomial $P_k(\lambda)$ of degree k together with the residual of Bi-CG. The hybrid Bi-CG method finds the residuals in the kth Sonnoveld space $\{P_k(\mathbf{A})\boldsymbol{r}_k^{\mathrm{bcg}} | \boldsymbol{r}_k^{\mathrm{bcg}} \perp \mathscr{K}_k(A^*; \tilde{\boldsymbol{r}}_0)\}$ [14]. When the relation $P_k(\lambda)$ is valid for the Lanczos polynomials $R_k(\lambda)$, the residuals $R_k(\mathbf{A})\boldsymbol{r}_k^{\mathrm{bcg}}$ of CGS can be obtained. The residuals of BiCGSTAB are expressed by $Q_k(\mathbf{A})\boldsymbol{r}_k^{\mathrm{bcg}}$. Here, the polynomial $Q_k(\lambda)$ is the Generalized Minimal RESidual (1) (GMRES (1)) [11] or Generalized Conjugate Residual (1) (GCR (1)) [3] polynomial. $H_k(\mathbf{A})\boldsymbol{r}_k^{\mathrm{bcg}}$ stands for the residuals of GPBiCG/BiCG×MR2, where the polynomials $H_k(\lambda)$ satisfy a three-term recurrence formula similar to the one for the Lanczos polynomials ([20]).

Sleijpen et al. have reformulated BiCGSTAB to clarify the relationship between BiCGSTAB and IDR(s) in [14,13]. The formulation of *Bi-CG*$_{(IDR)}$ described in [14,13] is different from that of the standard Bi-CG, i.e., (1) and (2), and the orthogonality (3) have been rewritten by alternative recurrences

$$\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k \mathbf{A}\boldsymbol{u}_k^{\mathrm{bcg}} \perp \tilde{\boldsymbol{r}}_k, \tag{4}$$

$$\mathbf{A}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k \mathbf{A}\boldsymbol{u}_k^{\mathrm{bcg}} \perp \tilde{\boldsymbol{r}}_k, \tag{5}$$

$$\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k \boldsymbol{u}_k^{\mathrm{bcg}}. \tag{6}$$

Note that $\mathbf{A}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ is obtained here by a vector update, whereas in the standard Bi-CG (6) is used to compute and then $\mathbf{A}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ is obtained by explicitly multiplying $\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ by $\mathbf{A}$. We will redesign the algorithms of the hybrid Bi-CG method using the formulas (4)–(6).

3. Alternative implementations of the hybrid Bi-CG methods

In this section, we derive alternative implementations of the CGS, BiCGSTAB and GPBiCG/BiCG×MR2 methods.

3.1. Bi-CG part

Since $\tilde{\boldsymbol{r}}_k$ can be expressed by $\bar{P}_k(\mathbf{A}^*)\tilde{\boldsymbol{r}}_0$, Eqs. (4)–(6) can be converted to

$$\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}} \perp \tilde{\boldsymbol{r}}_0, \tag{7}$$

$$\mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}} \perp \tilde{\boldsymbol{r}}_0, \tag{8}$$

$$\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}} \tag{9}$$

with $\mathbf{P}_k$ shorthand for $P_k(\mathbf{A})$ ([14]). We use Eqs. (7)–(9) as the Bi-CG part to design the recurrences for updating the residuals of the variants of CGS, BiCGSTAB and GPBiCG/BiCG×MR2. The Bi-CG coefficients α_k and β_k are determined such that both the vectors $\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ and $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, formed in (7) and (8), respectively, are orthogonal to $\tilde{\boldsymbol{r}}_0$. The Bi-CG coefficients α_k and β_k are computed as follows:

$$\alpha_k = (\tilde{\boldsymbol{r}}_0^*(\mathbf{P}_k\boldsymbol{r}_k^{\mathrm{bcg}}))/\sigma, \quad \beta_k = (\tilde{\boldsymbol{r}}_0^*(\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}))/\sigma, \text{ with } \sigma = \tilde{\boldsymbol{r}}_0^*(\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}). \tag{10}$$

The way β_k and $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ computed here differs from those in the standard Bi-CG.

3.2. Variants of GPBiCG/BiCG×MR2

In this section, following [1], we redesign the algorithms of GPBiCG/BiCG×MR2.

To obtain the next GPBiCG/BiCG×MR2 residuals $\boldsymbol{r}_{k+1} \equiv \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, the polynomials, which satisfy the three-term recurrence relation,

$$\begin{cases} P_0(\lambda) = 1, \quad P_1(\lambda) = (1-\zeta_0\lambda)P_0(\lambda), \\ P_{k+1}(\lambda) = (1+\eta_k-\zeta_k\lambda)P_k(\lambda) - \eta_k P_{k-1}(\lambda), \quad k = 1,2,\ldots, \end{cases} \tag{11}$$

can be used, and the residuals $\mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ are updated by

$$\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \quad \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = (1+\eta_k)\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \zeta_k\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \eta_k\mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} \tag{12}$$

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

$$\mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_{k-1}\boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k\mathbf{A}\mathbf{P}_{k-1}\boldsymbol{u}_k^{\mathrm{bcg}}. \tag{13}$$

Note that $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ is obtained as a vector update, while a multiplication by $\mathbf{A}$ is required to form $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}$ in (7).

To complete the update step, we can obtain $\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ by two ways. We use (11) and multiply (2) by $\mathbf{P}_{k-1}$ to find

$$\mathbf{P}_{k-1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{P}_{k-1}\boldsymbol{u}_k^{\mathrm{bcg}}, \tag{14}$$

$$\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = (1+\eta_k-\zeta_k\mathbf{A})\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \eta_k\mathbf{P}_{k-1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}. \tag{15}$$

As an alternative, (2) can be multiplied by $\mathbf{P}_{k+1}$. This requires the vector $\mathbf{P}_{k+1}\boldsymbol{u}_k^{\mathrm{bcg}}$ which can be computed by using (11). Hence,

$$\mathbf{P}_{k+1}\boldsymbol{u}_k^{\mathrm{bcg}} = (1+\eta_k-\zeta_k\mathbf{A})\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}} - \eta_k\mathbf{P}_{k-1}\boldsymbol{u}_k^{\mathrm{bcg}}, \tag{16}$$

$$\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{P}_{k+1}\boldsymbol{u}_k^{\mathrm{bcg}}. \tag{17}$$

The recurrence coefficients ζ_k and η_k are selected such that the residual norms are minimized as the function of ζ_k and η_k. With $\mathbf{Q}_k \equiv [\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}},\ (\mathbf{P}_{k-1}-\mathbf{P}_k)\boldsymbol{r}_{k+1}^{\mathrm{bcg}}]$, $(\zeta_k,\eta_k)^T = (\mathbf{Q}_k^*\mathbf{Q}_k)^{-1}\mathbf{Q}_k^*(\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}})$.

Introducing auxiliary vectors $\boldsymbol{r}_k \equiv \mathbf{P}_k\boldsymbol{r}_k^{\mathrm{bcg}}, \boldsymbol{u}_k \equiv \mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}$, $\mathbf{c}_k \equiv \mathbf{A}\mathbf{u}_k$, $\boldsymbol{r}'_k \equiv \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, $\mathbf{s}_k \equiv \mathbf{A}\mathbf{r}'_k$, $\boldsymbol{u}'_k \equiv \mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, $\mathbf{c}'_k \equiv \mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, $\boldsymbol{r}''_k \equiv \mathbf{P}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ and $\mathbf{w}_k \equiv \mathbf{P}_{k-1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}/\mathbf{P}_{k+1}\boldsymbol{u}_k^{\mathrm{bcg}}$ gives the algorithms. The variant using (7)–(9) and (12)–(15) is referred to as the variant 1, and the recurrences of this variant eventually coincides with those of BiCG×MR2. We refer to the variant using (16) and (17) as the variant 2.

By viewing $\boldsymbol{r}'_k$ and $\boldsymbol{r}''_k$ also as residuals, $\boldsymbol{r}_k \equiv \mathbf{b}-\mathbf{A}\mathbf{x}_k, \boldsymbol{r}'_k \equiv \mathbf{b}-\mathbf{A}\mathbf{x}'_k$, and $\boldsymbol{r}''_k \equiv \mathbf{b}-\mathbf{A}\mathbf{x}''_k$, we can derive the formulas $\mathbf{x}'_k = \mathbf{x}_k + \alpha_k\boldsymbol{u}_k, \mathbf{x}_{k+1} = (1+\eta_k)\mathbf{x}'_k + \zeta_k\boldsymbol{r}'_k - \eta_k\mathbf{x}''_k$ and $\mathbf{x}''_k = \mathbf{x}'_{k-1} + \alpha_k\boldsymbol{u}'_{k-1}$ to update the approximate solution from (7), (12) and (13).

The *variant 2* of GPBiCG/BiCG×MR2 can be found in Algorithm 1, which is expressed such that the memory requirements are saved. Note that, for the readability, we did not replace δr.

Algorithm 1. The variant 2 of GPBiCG/BiCG×MR2.

Select an $\mathbf{x}$. Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
$\mathbf{u} = \mathbf{r}$, $\mathbf{r}' = \mathbf{x}' = \mathbf{u}' = \mathbf{c}' = \mathbf{0}$
while $\|\boldsymbol{r}_k\|_2 >$ tol do
$\mathbf{c} = \mathbf{A}\mathbf{u}$, $\sigma = \tilde{\mathbf{r}}_0^* \mathbf{c}$, $\alpha = \tilde{\mathbf{r}}_0^* \mathbf{r}/\sigma$
$\mathbf{r}' = \mathbf{r}' - \alpha \mathbf{c}'$, $\mathbf{x}' = \mathbf{x}' + \alpha \mathbf{u}'$
$\mathbf{r} = \mathbf{r} - \alpha \mathbf{c}$, $\mathbf{x} = \mathbf{x} + \alpha \mathbf{u}$
$\mathbf{s} = \mathbf{A}\mathbf{r}$, $\beta = \tilde{\mathbf{r}}_0^* \mathbf{s}/\sigma$
$\mathbf{c}' = \mathbf{s} - \beta \mathbf{c}$
$\boldsymbol{\delta r} = \mathbf{r}' - \mathbf{r}$
$(\zeta, \eta)^T = (\mathbf{Q}^* \mathbf{Q})^{-1} \mathbf{Q}^* \mathbf{r}$, where $\mathbf{Q} \equiv [\mathbf{s}, \boldsymbol{\delta r}]$
$\mathbf{r}' = \mathbf{r} - \zeta \mathbf{s} - \eta \boldsymbol{\delta r}$, $\mathbf{x}' = (1+\eta)\mathbf{x} + \zeta \mathbf{r} - \eta \mathbf{x}'$
swap($\mathbf{r}$, $\mathbf{r}'$), swap($\mathbf{x}$, $\mathbf{x}'$)
$\mathbf{u}' = \mathbf{r}' - \beta \mathbf{u}$
$\mathbf{u} = \mathbf{r} - (\beta + \beta\eta)\mathbf{u} + (\beta\zeta)\mathbf{c} + (\beta\eta)\mathbf{u}'$
end

Moreover, we will derive variants of GPBiCG/BiCG×MR2 by combining the *Bi-CG*$_{(IDR)}$ formulation and the coupled two-term recurrences of Rutishauser [9]:

$$\begin{aligned} \widetilde{G}_{k+1}(\lambda) &= \zeta_k \lambda P_k(\lambda) + \eta_k \widetilde{G}_k(\lambda), \\ P_{k+1}(\lambda) &= P_k(\lambda) - \widetilde{G}_{k+1}(\lambda). \end{aligned} \tag{18}$$

First, we use the *Bi-CG*$_{(IDR)}$ part (7)–(9). The scalars α_k and β_k are computed by (10). Next, multiplying (1) by the matrix polynomial $\widetilde{\mathbf{G}}_k$ and using (18) give the following update formulas for the residual

$$\begin{aligned} & & \widetilde{\mathbf{G}}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}} &= \widetilde{\mathbf{G}}_k \boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k \mathbf{A}\widetilde{\mathbf{G}}_k \boldsymbol{u}_k^{\mathrm{bcg}}, \\ \mathbf{A}\mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}}, & & \widetilde{\mathbf{G}}_{k+1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}} &= \zeta_k \mathbf{A}\mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}} + \eta_k \widetilde{\mathbf{G}}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \\ & & \mathbf{P}_{k+1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}} &= \mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \widetilde{\mathbf{G}}_{k+1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \end{aligned} \tag{19}$$

where $\mathbf{P}_k$ and $\widetilde{\mathbf{G}}_k$ shorthand for $P_k(\mathbf{A})$ and $\widetilde{G}_k(\mathbf{A})$ as before. 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. For the update vector $\mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}}$ we have

$$\begin{aligned} & & \widetilde{\mathbf{G}}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}} &= \widetilde{\mathbf{G}}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k \widetilde{\mathbf{G}}_k \boldsymbol{u}_k^{\mathrm{bcg}}, \\ & & \widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}} &= \zeta_k \mathbf{A}\mathbf{P}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}} + \eta_k \widetilde{\mathbf{G}}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \\ \mathbf{A}\widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}}, & & \mathbf{A}\mathbf{P}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}} &= \mathbf{A}\mathbf{P}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \mathbf{A}\widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \\ & & \mathbf{P}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}} &= \mathbf{P}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}}. \end{aligned} \tag{20}$$

Note that a multiplication by $\mathbf{A}$ is required to form $\mathbf{A}\widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, while $\mathbf{A}\mathbf{P}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ and $\mathbf{A}\mathbf{P}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ are obtained as vector updates. As an alternative to the first and second lines in (20) we can change the order; first the Bi-CG update then the stabilization polynomial update:

$$\begin{aligned} \widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_k^{\mathrm{bcg}} &= \zeta_k \mathbf{A}\mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}} + \eta_k \widetilde{\mathbf{G}}_k \boldsymbol{u}_k^{\mathrm{bcg}}, \\ \widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}} &= \widetilde{\mathbf{G}}_{k+1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k \widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_k^{\mathrm{bcg}}. \end{aligned} \tag{21}$$

As before, the ζ_k and η_k can be determined so as to minimize the residual norms $\|\boldsymbol{r}_{k+1}\|_2$.

Introducing auxiliary vectors $\boldsymbol{r}_k \equiv \mathbf{P}_k \boldsymbol{r}_k^{\mathrm{bcg}}$, $\boldsymbol{r}'_k \equiv \mathbf{P}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, $\delta \boldsymbol{r}_k \equiv \widetilde{\mathbf{G}}_{k+1} \boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, $\delta \boldsymbol{r}'_k \equiv \widetilde{\mathbf{G}}_k \boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, $\boldsymbol{u}_k \equiv \mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}}$, $\boldsymbol{u}'_k \equiv \mathbf{P}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, $\delta \boldsymbol{u}_k \equiv \widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, $\delta \boldsymbol{u}'_k \equiv \widetilde{\mathbf{G}}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}} / \widetilde{\mathbf{G}}_{k+1} \boldsymbol{u}_k^{\mathrm{bcg}}$, $\mathbf{c}_k \equiv \mathbf{A}\mathbf{P}_k \boldsymbol{u}_k^{\mathrm{bcg}}$, $\mathbf{c}'_k \equiv \mathbf{A}\mathbf{P}_k \boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, $\mathbf{s}_k \equiv \mathbf{A}\boldsymbol{r}'_k$ and $\delta \boldsymbol{c}_k \equiv \mathbf{A}\delta \boldsymbol{u}_k$ gives the algorithms. We refer to the variant using Eqs. (7)–(9), (19) and (20) as variant 3 and to that using (21) instead of the first and second lines in (20) as variant 4.

The update formulas $\mathbf{x}'_k = \mathbf{x}_k + \alpha_k \boldsymbol{u}_k$, $\delta \boldsymbol{x}'_k = \delta \boldsymbol{x}_{k-1} + \alpha_k \delta \boldsymbol{u}_{k-1}$, $\delta \boldsymbol{x}_k = -\zeta_k \boldsymbol{r}'_k + \eta_k \delta \boldsymbol{x}'_k$ and $\mathbf{x}_{k+1} = \mathbf{x}'_k - \delta \boldsymbol{x}_k$ for the approximate solutions can be obtained from (7) and (19) for the residual using that $\boldsymbol{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$, $\boldsymbol{r}'_k = \mathbf{b} - \mathbf{A}\mathbf{x}'_k$, $\delta \boldsymbol{r}_k = -\mathbf{A}\delta \boldsymbol{x}_k$, and $\delta \boldsymbol{r}'_k = -\mathbf{A}\delta \boldsymbol{x}'_k$.

The *variant 4* of GPBiCG/BiCG×MR2 can be found in Algorithm 2, which is expressed such that the memory requirements are saved. Note that, for the readability, we did not replace $\delta \boldsymbol{c}$.

Algorithm 2. The variant 4 of GPBiCG/BiCG×MR2.

Select an $\mathbf{x}$. Compute $r = \mathbf{b} - \mathbf{Ax}$
$\mathbf{u} = \mathbf{r}$, $\mathbf{c} = \mathbf{A}r$, $\delta r = \delta c = \delta x = \delta u = \mathbf{0}$
while $\|r\|_2 >$ tol do
$\sigma = \tilde{\mathbf{r}}_0^* \mathbf{c}$, $\alpha = \tilde{\mathbf{r}}_0^* \mathbf{r}/\sigma$
$\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}$, $\mathbf{x} = \mathbf{x} + \alpha\mathbf{r}$
$\delta r = \delta r - \alpha\delta c$, $\delta x = \delta x + \alpha\delta u$
$\mathbf{s} = \mathbf{Ar}$
$(\zeta, \eta)^T = (\mathbf{Q}^*\mathbf{Q})^{-1}\mathbf{Q}^*\mathbf{r}$, where $\mathbf{Q} \equiv [\mathbf{s}, \delta r]$
$\delta r = \zeta\mathbf{s} + \eta\delta r$, $\delta x = -\zeta\mathbf{r} + \eta\delta x$
$\beta = \tilde{r}_0^* \mathbf{s}/\sigma$
$\delta u = \delta r - (\beta\eta)\delta u - (\beta\zeta)\mathbf{c}$
$\mathbf{c} = \mathbf{s} - \beta\mathbf{c}$, $\mathbf{u} = \mathbf{r} - \beta\mathbf{r}$
$\delta c = \mathbf{A}\delta u$, $\mathbf{c} = \mathbf{c} - \delta c$, $\mathbf{u} = \mathbf{u} - \delta u$
$\mathbf{r} = \mathbf{r} - \delta r$, $\mathbf{x} = \mathbf{x} - \delta x$
end

Table 1 summarizes the computational costs and the memory requirements of the original GPBiCG method, BiCG×MR2 and the proposed variants per iteration. The computation of $\|\boldsymbol{r}_k\|_2$ for the termination criterion requires 1 Dot per step. $\tilde{\boldsymbol{r}}_0$ has to be stored. We did not count storage for $\mathbf{b}$. Note that, in the variants 1–4 of GPBiCG/BiCG×MR2 since $\boldsymbol{r} = \boldsymbol{r}' - \zeta\mathbf{s} - \eta\delta r$ and $\boldsymbol{r}', \delta r \perp \tilde{\boldsymbol{r}}_0$, we have that $\tilde{\boldsymbol{r}}_0^* \boldsymbol{r} = -\zeta\tilde{\boldsymbol{r}}_0^* \mathbf{s}$. This relation saves the computation of one inner product per step.

3.3. Variants of BiCGSTAB

Our variants of BiCGSTAB will be derived by combining the *Bi-CG*$_{(IDR)}$ formulation and the two-term recurrence for the stabilization polynomial, which is obtained by putting $\eta_k = 0$ in (11).

First, we use the *Bi-CG*$_{(IDR)}$ part (7)–(9). The scalars α_k and β_k are computed by (10). Next, using (11) with $\eta_k = 0$, we update the next BiCGSTAB residuals $\boldsymbol{r}_{k+1} \equiv \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ by

$$\mathbf{AP}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \quad \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \zeta_k\mathbf{AP}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}. \tag{22}$$

The computation of the vector $\mathbf{AP}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ requires a multiplication by $\mathbf{A}$, which we indicated by putting this vector in the left column. To complete the update step, we can obtain $\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ by two ways. Multiplying (11) with $\eta_k = 0$ by $\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ finds

$$\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \zeta_k\mathbf{AP}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}. \tag{23}$$

Note that $\mathbf{AP}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ is obtained as a vector update. We can also obtain $\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ by an alternative formula

$$\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{P}_{k+1}\boldsymbol{u}_k^{\mathrm{bcg}} = \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k(\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}} - \zeta_k\mathbf{AP}_k\boldsymbol{u}_k^{\mathrm{bcg}}). \tag{24}$$

Note that (9) is not needed for updating the residual when using (24), and that a multiplication by $\mathbf{A}$ is required to form $\mathbf{AP}_k\boldsymbol{u}_k^{\mathrm{bcg}}$ in (7).

The recurrence coefficient ζ_k is selected such that the residual norms $\|\boldsymbol{r}_{k+1}\|_2$ are minimized as the function of ζ_k. With $\mathbf{s}_k = \mathbf{AP}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \zeta_k = \mathbf{s}_k^*(\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}})/\mathbf{s}_k^*\mathbf{s}_k$.

We obtain the algorithms by introducing auxiliary vectors $\boldsymbol{r}_k \equiv \mathbf{P}_k\boldsymbol{r}_k^{\mathrm{bcg}}$, $\boldsymbol{u}_k \equiv \mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}$, $\mathbf{c}_k \equiv \mathbf{Au}_k$, $\boldsymbol{r}_k' \equiv \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, $\mathbf{s}_k \equiv \mathbf{Ar}_k'$, $\boldsymbol{u}_k' \equiv \mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$, and $\mathbf{c}_k' \equiv \mathbf{AP}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$. The variant using (7)–(9), (22) and (23) is referred to as the variant 1 of BiCGSTAB and coincides with the reformulated BiCGSTAB described in [14]. The variant using (7), (8), (22) and (24) is referred to as the variant 2 of BiCGSTAB. The recurrences of our variant 2 are the same as those of the original BiCGSTAB, but the coefficient β_k differs in that of the original.

By viewing $\boldsymbol{r}_k'$ also as residuals, $\boldsymbol{r}_k \equiv \mathbf{b} - \mathbf{Ax}_k$ and $\boldsymbol{r}_k' \equiv \mathbf{b} - \mathbf{Ax}_k'$, we can obtain the formulas $\mathbf{x}_k' = \mathbf{x}_k + \alpha_k\boldsymbol{u}_k$ and $\mathbf{x}_{k+1} = \mathbf{x}_k' + \zeta_k\boldsymbol{r}_k'$ to update the approximate solutions from (7) and (22).

The *variant 1* of BiCGSTAB can be found in Algorithm 3, which is expressed such that the memory requirements are saved.

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

	MV	Dot	AXPY	Memory
GPBiCG	2	8	14	11
BiCG×MR2	2	8	15.5	10
variant 1	2	9(8)	14.5	10
variant 2	2	9(8)	14	10
variant 3	2	9(8)	13	9
variant 4	2	9(8)	13.5	9

Algorithm 3. The variant 1 of BiCGSTAB.

Select an $\mathbf{x}$. Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
$\mathbf{u} = \mathbf{r}$
while $\|\mathbf{r}_k\|_2 >$ tol do
  $\mathbf{c} = \mathbf{A}\mathbf{u}, \quad \sigma = \tilde{\mathbf{r}}_0^*\mathbf{c}, \quad \alpha = \tilde{\mathbf{r}}_0^*\mathbf{r}/\sigma$
  $\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}, \quad \mathbf{x} = \mathbf{x} + \alpha\mathbf{u}$
  $\mathbf{s} = \mathbf{A}\mathbf{r}$
  $\beta = \tilde{\mathbf{r}}_0^*\mathbf{s}/\sigma$
  $\mathbf{c} = \mathbf{s} - \beta\mathbf{c}$
  $\mathbf{u} = \mathbf{r} - \beta\mathbf{u}$
  $\zeta = \mathbf{s}^*\mathbf{r}/\mathbf{s}^*\mathbf{s}$
  $\mathbf{r} = \mathbf{r} - \zeta\mathbf{s}, \quad \mathbf{x} = \mathbf{x} + \zeta\mathbf{r}$
  $\mathbf{u} = \mathbf{u} - \zeta\mathbf{c}$
end

Table 2 summarizes the computational costs and the memory requirements of the original BiCGSTAB method and the proposed variants per iteration. The computation of $\|\boldsymbol{r}_k\|_2$ for the termination criterion requires 1 Dot per step. $\tilde{\boldsymbol{r}}_0$ has to be stored. We did not count storage for $\mathbf{b}$. Note that, since $\boldsymbol{r} = \boldsymbol{r}' - \zeta\mathbf{s}$ and $\boldsymbol{r}' \perp \tilde{\boldsymbol{r}}_0$, we have that $\tilde{\boldsymbol{r}}_0^*\boldsymbol{r} = -\zeta\tilde{\boldsymbol{r}}_0^*\mathbf{s}$. This relation saves the computation of one inner product per step.

3.4. Variants of CGS

We will derive variants of CGS by combining the *Bi-CG*$_{(IDR)}$ formulation and the recurrences (25) and (26).
With $\zeta_k \equiv \alpha_k$ and $\eta_k \equiv -\beta_{k-1}\frac{\alpha_k}{\alpha_{k-1}}$, the polynomials $P_k(\lambda)$ also satisfy the coupled two-term recurrences

$$G_k(\lambda) = P_k(\lambda) - \beta_{k-1}G_{k-1}(\lambda), \tag{25}$$
$$P_{k+1}(\lambda) = P_k(\lambda) - \alpha_k\lambda G_k(\lambda), \tag{26}$$

which correspond to the Bi-CG formulation: the polynomials P_k and G_k of CGS can be viewed as the polynomials of Bi-CG that define $\boldsymbol{r}_k^{\mathrm{bcg}}$ and $\boldsymbol{u}_k^{\mathrm{bcg}}$, respectively, where $\mathbf{P}_k$ and $\mathbf{G}_k$ shorthand for $P_k(\mathbf{A})$ and $G_k(\mathbf{A})$ as before.

First, we use the *Bi-CG*$_{(IDR)}$ part (7)–(9). The scalars α_k and β_k are computed by (10). The next CGS residuals $\boldsymbol{r}_{k+1} \equiv \mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ are updated by

$$\mathbf{P}_{k+1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \alpha_k\mathbf{A}\mathbf{G}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}. \tag{27}$$

Multiplying Eqs. (1)–(2) by the matrix polynomial $\mathbf{A}\mathbf{G}_{k-1}$, and multiplying (25) by $\mathbf{A}\mathbf{r}_{k+1}^{\mathrm{bcg}}$ and $\mathbf{A}\mathbf{u}_{k+1}^{\mathrm{bcg}}$ give the following update formulas for the residual:

$$\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \quad \mathbf{A}\mathbf{G}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_{k-1}\mathbf{A}\mathbf{G}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \tag{28}$$
$$\mathbf{A}\mathbf{G}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\mathbf{G}_{k-1}\boldsymbol{r}_k^{\mathrm{bcg}} - \alpha_k\mathbf{A}(\mathbf{A}\mathbf{G}_{k-1}\boldsymbol{u}_k^{\mathrm{bcg}}), \tag{29}$$
$$\mathbf{A}\mathbf{G}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \beta_{k-1}\mathbf{A}\mathbf{G}_{k-1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \tag{30}$$
$$\mathbf{A}\mathbf{G}_{k-1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} = \mathbf{A}\mathbf{G}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{A}\mathbf{G}_{k-1}\boldsymbol{u}_k^{\mathrm{bcg}}. \tag{31}$$

Note that the computation of the vector $\mathbf{A}\mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$ requires a multiplication by $\mathbf{A}$, while $\mathbf{A}\mathbf{G}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, $\mathbf{A}\mathbf{G}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}$, $\mathbf{A}\mathbf{G}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ and $\mathbf{A}\mathbf{G}_{k-1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ are obtained as vector updates. For the update vectors $\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}$ and $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}$ we have

$$\begin{aligned} \mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} &= \mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \alpha_k\mathbf{A}\mathbf{G}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \\ \mathbf{A}(\mathbf{A}\mathbf{G}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}), \quad \mathbf{A}\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}} &= \mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} - \alpha_k\mathbf{A}(\mathbf{A}\mathbf{G}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}). \end{aligned} \tag{32}$$

Note that the vector $\mathbf{A}(\mathbf{A}\mathbf{G}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}})$ is obtained by multiplying the vector $\mathbf{A}\mathbf{G}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ by $\mathbf{A}$, while $\mathbf{A}\mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ and $\mathbf{A}\mathbf{P}_{k+1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}$ are obtained as vector updates. As alternatives to (30) and (31), we can also use the following formulas

Table 2
Computational costs and memory requirements of BiCGSTAB and the variants per iteration.

	MV	Dot	AXPY	Memory
BiCGSTAB	2	5	6	6
variant 1	2	6(5)	7	6
variant 2	2	6(5)	6	6

Table 3
Computational costs and memory requirements of CGS and the variants per iteration.

	MV	Dot	AXPY	Memory
CGS	2	3	6.5	7
variants 1 and 2	2	4(3)	14	9

$$\begin{aligned}\mathbf{AG}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}} &= \mathbf{AG}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}} - \beta_k\mathbf{AG}_k\boldsymbol{u}_k^{\mathrm{bcg}},\\ \mathbf{AG}_k\boldsymbol{u}_k^{\mathrm{bcg}} &= \mathbf{AP}_k\boldsymbol{u}_k^{\mathrm{bcg}} - \beta_{k-1}\mathbf{AG}_{k-1}\boldsymbol{u}_k^{\mathrm{bcg}}.\end{aligned} \tag{33}$$

Introducing auxiliary vectors $\boldsymbol{r}_k \equiv \mathbf{P}_k\boldsymbol{r}_k^{\mathrm{bcg}}, \boldsymbol{r}'_k \equiv \mathbf{P}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \delta\boldsymbol{r}_k \equiv \mathbf{AG}_k\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \delta\boldsymbol{r}'_k \equiv \mathbf{AG}_{k-1}\boldsymbol{r}_{k+1}^{\mathrm{bcg}}, \boldsymbol{u}_k \equiv \mathbf{P}_k\boldsymbol{u}_k^{\mathrm{bcg}}, \boldsymbol{u}'_k \equiv \mathbf{P}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \delta\boldsymbol{u}_k \equiv \mathbf{AG}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \delta\boldsymbol{u}'_k \equiv \mathbf{AG}_{k-1}\boldsymbol{u}_{k+1}^{\mathrm{bcg}}/\mathbf{AG}_k\boldsymbol{u}_k^{\mathrm{bcg}}, \mathbf{c}_k \equiv \mathbf{AP}_k\boldsymbol{u}_k^{\mathrm{bcg}}, \mathbf{c}'_k \equiv \mathbf{AP}_k\boldsymbol{u}_{k+1}^{\mathrm{bcg}}, \mathbf{s}'_k \equiv \mathbf{A}\boldsymbol{r}'_k$ and $\delta\boldsymbol{c}_k \equiv \mathbf{A}\delta\boldsymbol{u}_k$ gives the algorithms. We refer to the variant using Eqs. (7)–(9) and (27)–(32) as variant 1 and to that using (33) instead of (30) and (31) as variant 2.

The update formulas $\mathbf{x}'_k = \mathbf{x}_k + \alpha_k\boldsymbol{u}_k, \mathbf{x}_{k+1} = \mathbf{x}'_k - \alpha_k\delta\boldsymbol{x}_k, \delta\boldsymbol{x}_k = -\boldsymbol{r}'_k - \beta_{k-1}\delta\boldsymbol{x}'_k$ and $\delta\boldsymbol{x}'_k = \delta\boldsymbol{x}_{k-1} + \alpha_k\delta\boldsymbol{u}_{k-1}$ for the approximate solution can be obtained from (7) and (27)–(29) for the residual using that $\boldsymbol{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k, \boldsymbol{r}'_k = \mathbf{b} - \mathbf{A}\mathbf{x}'_k, \delta\boldsymbol{r}_k = -\mathbf{A}\delta\boldsymbol{x}_k$, and $\delta\boldsymbol{r}'_k = -\mathbf{A}\delta\boldsymbol{x}'_k$.

The *variant 1* of CGS can be found in Algorithm 4, which is expressed such that the memory requirements are saved.

Algorithm 4. *The variant 1 of CGS.*

Select an $\mathbf{x}$. Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
$\mathbf{u} = \mathbf{r}$, $\mathbf{c} = \mathbf{A}\mathbf{r}$, $\delta\mathbf{r} = \delta x = \delta u = \mathbf{0}$
$\alpha = \beta = 0$
while $\|\mathbf{r}\|_2 >$ tol do
  $\mathbf{s} = \mathbf{A}\delta u$
  $\mathbf{c} = \mathbf{c} - \alpha\mathbf{s}$, $\mathbf{u} = \mathbf{u} - \alpha\delta u$
  $\sigma = \tilde{\mathbf{r}}_0^*\mathbf{c}$, $\alpha = \tilde{\mathbf{r}}_0^*\mathbf{r}/\sigma$
  $\mathbf{r} = \mathbf{r} - \alpha\mathbf{c}$, $\mathbf{x} = \mathbf{x} + \alpha\mathbf{u}$
  $\delta r = \delta r - \alpha\mathbf{s}$, $\delta x = \delta x + \alpha\delta u$
  $\mathbf{s} = \mathbf{A}\mathbf{r}$
  $\beta' = \beta$, $\beta = \tilde{\mathbf{r}}_0^*\mathbf{s}/\sigma$
  $\mathbf{c} = \mathbf{s} - \beta\mathbf{c}$, $\mathbf{u} = \mathbf{r} - \beta\mathbf{u}$
  $\delta u = \delta r - \beta\delta u$
  $\delta r = \mathbf{s} - \beta'\delta r$, $\delta x = -\mathbf{r} - \beta'\delta x$
  $\mathbf{r} = \mathbf{r} - \alpha\delta r$, $\mathbf{x} = \mathbf{x} - \alpha\delta x$
  $\delta u = \mathbf{c} - \beta'\delta u$
  end

Table 3 summarizes the computational costs and the memory requirements of the original CGS method and our proposed variants per iteration. The computation of $\|\boldsymbol{r}_k\|_2$ for the termination criterion requires 1 Dot per step. $\tilde{\boldsymbol{r}}_0$ has to be stored. We did not count storage for $\mathbf{b}$. The number of vector updates in our proposed variants of CGS is much more than that in the original CGS since the recurrences of our variants have been derived by the same analogy as was used in GCGS. Note that, in our proposed variants of CGS since $\boldsymbol{r} = \boldsymbol{r}' - \alpha\mathbf{s} + \alpha\beta'\delta\boldsymbol{r}$ and $\boldsymbol{r}', \delta\boldsymbol{r} \perp \tilde{\boldsymbol{r}}_0$, we have that $\tilde{\boldsymbol{r}}_0^*\boldsymbol{r} = -\alpha\tilde{\boldsymbol{r}}_0^*\mathbf{s}$. This relation saves the computation of one inner product per step.

For variants of GPBiCG/BiCG×MR2, BiCGSTAB and CGS, preconditioning can be included by replacing the equation (1) by $(K_1^{-1}AK_2^{-1})\,(K_2\mathbf{x}) = K_1^{-1}\mathbf{b}$ and applying the algorithms to the preconditioned systems $\tilde{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$ with $\tilde{A} = K_1^{-1}AK_2^{-1}$, $\tilde{\mathbf{x}} = K_2\mathbf{x}$ and $\tilde{\mathbf{b}} = K_1^{-1}\mathbf{b}$, where $A \approx K = K_1K_2$ and K is a preconditioner. We will not further discuss preconditioning.

4. Numerical experiments

In this section, we present numerical experiments on model problems, and reveal the merit of the variants. The examples show the typical convergence behavior of the variants for situations where the convergence is slow (with a long stagnation phase) for BiCGSTAB and GPBiCG/BiCG×MR2, and the typical convergence behavior that the approximate solutions solved by the variants of CGS are more accurate.

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 $\tilde{\boldsymbol{r}}_0$ is set to random vector. The iterations are terminated when the reduction of residual norms (i.e., $\|\boldsymbol{r}_k\|_2/\|\boldsymbol{r}_0\|_2$) becomes $\leqslant 10^{-10}$.

As shown in [10,6], applying 5-point central differences to the two-dimensional convection–diffusion equation

$$-u_{xx} - u_{yy} + \gamma(xu_x + yu_y) + \beta u = f(x, y)$$

Table 4
Characteristic of coefficient matrices.

Matrices	N	NNZ	Ave. NNZ
sherman1	1000	3750	3.7
dw2048	2048	10114	4.9

Table 5
Number of MVs and explicitly computed relative residual norms.

	MVs (ratio)	True res.
GPBiCG	n.c. (−)	−
BiCG×MR2	2432 (1.15)	2.1e-09
variant 1	2100 (1.00)	7.7e-11
variant 2	2640 (1.25)	4.2e-11
variant 3	2402 (1.14)	5.6e-11
variant 4	2696 (1.28)	8.9e-11

Table 6
Number of MVs and explicitly computed relative residual norms.

	MVs (ratio)	True res.
BiCGSTAB	n.c. (−)	−
variant 1	879 (1.00)	4.7e-11
variant 2	843 (1.04)	6.5e-11

Table 7
Number of MVs and explicitly computed relative residual norms for sherman1 (on the left) and dw2048 (on the right).

	MVs (ratio)	True res.		MVs (ratio)	True res.
CGS	1132 (1.19)	1.2e-06	CGS	3734 (1.23)	1.1e-06
variant 1	950 (1.00)	1.6e-09	variant 1	3020 (1.00)	5.1e-10
variant 2	994 (1.04)	2.3e-09	variant 2	4028 (1.33)	9.1e-11

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 $M+1$ in both directions of Ω, so that the resulting system has an $M^2 \times M^2$ coefficient matrix. The parameters M, γ and β are set to $M = 64,\ \gamma = 1000$ and $\beta = 10$ ([10]) to compare the convergence behavior of the variants of GPBiCG, and $M = 63,\ \gamma = 100$ and $\beta = -200$ ([6]) to compare the convergence behavior of the variants of BiCGSTAB. The right-hand side vector $\mathbf{b}$ is given by substituting a vector $\hat{\mathbf{x}} = (1, \ldots, 1)^{\mathrm{T}}$ into the equation $\mathbf{b} = \mathbf{A}\hat{\mathbf{x}}$. We use matrices sherman1 (oil reservoir modeling, symmetric) and dw2048 (electromagnetics problem, nonsymmetric) from Tim Davis' Sparse Matrix Collection and Matrix Market to compare the convergence behavior of the variants of CGS. Table 4 shows the dimensions of the matrices (indicated by *N*), the number of nonzero entries (abbreviated as *NNZ*) and the average number of nonzero entries in each row (abbreviated as *Ave. NNZ*).

The linear systems with the coefficient matrices are solved by the original GPBiCG, BiCG×MR2, the variants of GPBiCG, the original BiCGSTAB, the variants of BiCGSTAB, the original CGS and the variants of CGS. The convergence behaviors are compared among the original GPBiCG, BiCG×MR2 and the variants of GPBiCG/BiCG×MR2, among the original BiCGSTAB and the variants of BiCGSTAB, and among the original CGS and the variants of CGS. Tables 5–7, show the number of matrix–vector products (*MVs*) and the explicitly computed relative residual 2-norms $\log_{10}(\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|_2/\|\mathbf{r}_0\|_2)$ (abbreviated as *True res.*) at termination. The convergence behaviors are displayed in Figs. 1–3, and the number of MVs (on the horizontal axis) is plotted versus the log10 of the relative residual 2-norms.

From Table 5 and Fig. 1, we learn that the residual norms of the original GPBiCG do not converge, i.e., the method has a long stagnation. The residual norms of the proposed variants and BiCG×MR2 converge.

From Table 6 and Fig. 2, we learn that the residual norms of the original BiCGSTAB do not converge, i.e., the method has a long stagnation. The residual norms of the proposed variants converge.

From Table 7 and Fig. 3, we learn that the residual norms of the original CGS and the variants converge, i.e., the methods do not have a long stagnation. The explicitly computed relative residual norms obtained by the original CGS become 10^{-6} and those obtained by our variants attain 10^{-9} for sherman1 and 10^{-10} for dw2048. The peak of the residual norms of the original CGS reaches to about 10^{10}, while that of our variants becomes about 10^{7}. It appears that this difference in the peak is a reason why the approximate solutions obtained by the variants are more accurate (see [5] for the detail).

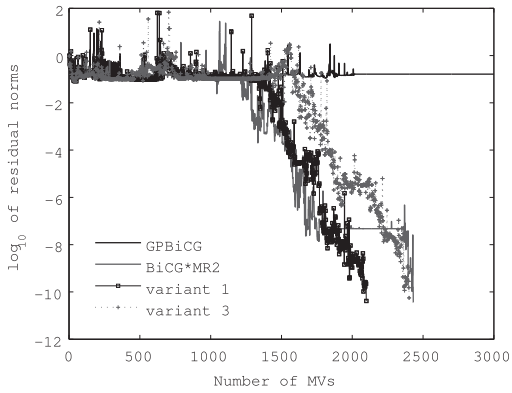

Fig. 1. Convergence history of GPBiCG, BiCG×MR2 and the variants 1 and 3.

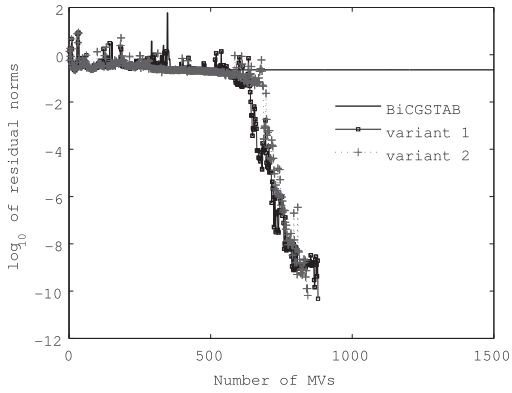

Fig. 2. Convergence history of BiCGSTAB and the variants.

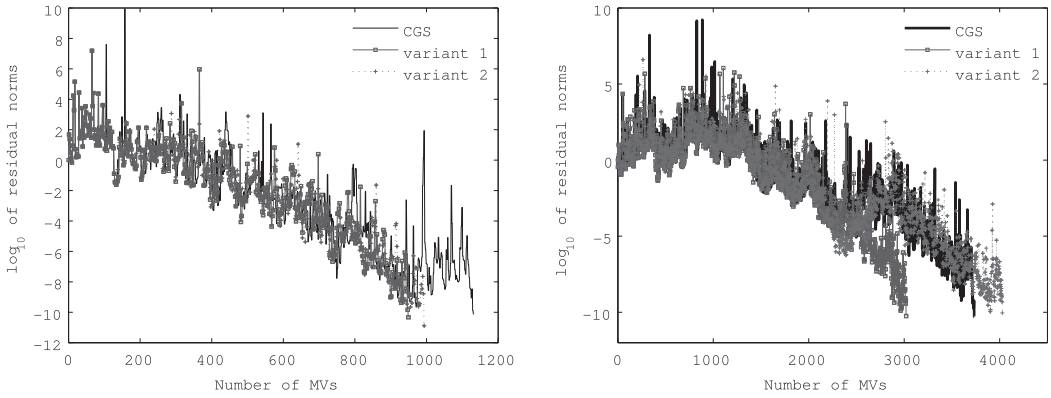

Fig. 3. Convergence history of CGS and the variants for sherman1 (on the left) and dw2048 (on the right).

Next, we focus on GPBiCG/BiCG×MR2 and BiCGSTAB of which the residual norms converge slowly (with a long stagnation phase). The quantity $\frac{|\mathbf{x}^*\mathbf{y}|}{|\mathbf{x}|^*|\mathbf{y}|}$ is known as a measure for the accuracy of the computation of the inner product between $\mathbf{x}$ and $\mathbf{y}$, where $\mathbf{x}$ and $\mathbf{y}$ are n-vectors (see [15] for the detail). The computation of the Bi-CG coefficient β of the original methods is different from that of the variants; the inner product $\tilde{\boldsymbol{r}}_0^*\boldsymbol{r}$ is computed in the original methods, while the one $\tilde{\boldsymbol{r}}_0^*\mathbf{s}$ is calculated in the variants. We therefore examine the quantity $\frac{|\tilde{\boldsymbol{r}}_0^*\boldsymbol{r}|}{|\tilde{\boldsymbol{r}}_0|^*|\boldsymbol{r}|}$ or $\frac{|\tilde{\boldsymbol{r}}_0^*\mathbf{s}|}{|\tilde{\boldsymbol{r}}_0|^*|\mathbf{s}|}$ to clarify the accuracy of the coefficient β. We display the $\log_{10}$ of the quantity $\frac{|\tilde{\boldsymbol{r}}_0^*\boldsymbol{r}|}{|\tilde{\boldsymbol{r}}_0|^*|\boldsymbol{r}|}$ for the original GPBiCG, BiCG×MR2 and BiCGSTAB methods, and the $\log_{10}$ of the quantity $\frac{|\tilde{\boldsymbol{r}}_0^*\mathbf{s}|}{|\tilde{\boldsymbol{r}}_0|^*|\mathbf{s}|}$ for the variants in the Figs. 4, 5. From Figs. 4, 5, we can observe that the quantity calculated by the original GPBiCG and

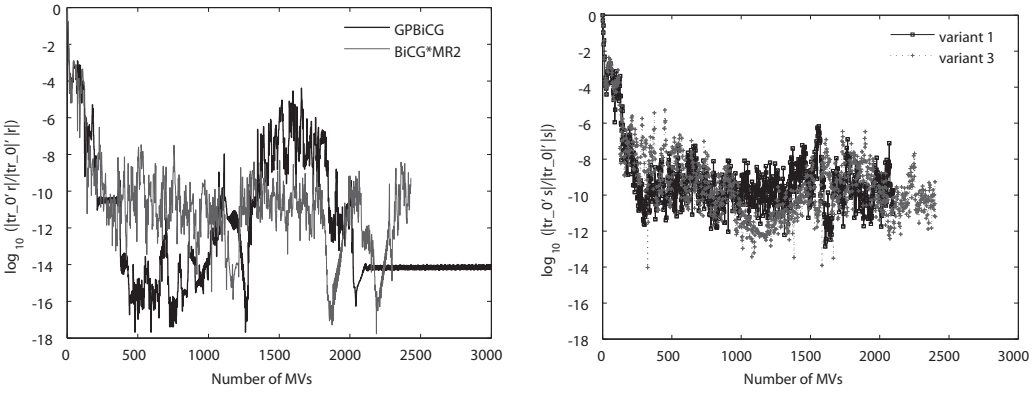

Fig. 4. Quantity $\frac{|\tilde{r}_0^* r|}{|\tilde{r}_0|^*|r|}$ for GPBiCG and BiCG×MR2 (on the left) and quantity $\frac{|\tilde{r}_0^* s|}{|\tilde{r}_0|^*|s|}$ for the variants of GPBiCG (on the right).

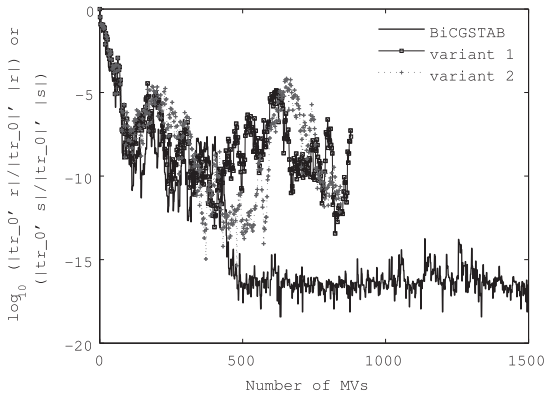

Fig. 5. Quantity $\frac{|\tilde{r}_0^* r|}{|\tilde{r}_0|^*|r|}$ for BiCGSTAB and $\frac{|\tilde{r}_0^* s|}{|\tilde{r}_0|^*|s|}$ for the variants of BiCGSTAB.

BiCG×MR2 reaches to less than 10^{-15}, but that provided by the variant 1 becomes about 10^{-12}, i.e., three digits are accurate for the variant 1 of GPBiCG/BiCG×MR2. The quantity computed by the original BiCGSTAB reaches to less than 10^{-15}, but that computed by the variant 1 becomes about 10^{-13}, i.e., two digits are accurate for the variant 1 of BiCGSTAB. The modification for the computation of the coefficient β leads to faster convergence.

5. Concluding remarks

We designed variants of CGS, BiCGSTAB and GPBiCG/BiCG×MR2, which are derived by using slightly modified recurrences of Bi-CG, i.e., the formulation of *Bi-CG*$_{(IDR)}$ used in the reformulated BiCGSTAB. Although these variants are mathematically equivalent to their counterparts, the computation of one of the Bi-CG coefficients and the recurrences of the variants are (partly) different from those of the original CGS, BiCGSTAB and GPBiCG/BiCG×MR2 methods. Numerical experiments show that the variants of BiCGSTAB and GPBiCG/BiCG×MR2 lead to faster convergence typically for linear systems for which the methods converge slowly (with long stagnation phase), and that the variants of CGS attain more accurate approximate solutions. The results imply that the hybrid Bi-CG methods profit from inside IDR.

Acknowledgments

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, 2011.

References

[1] K. Abe, G.L.G. Sleijpen, Solving linear equations with a stabilized GPBiCG method, to be published in Applied Numerical Mathematics, (http://dx.doi.org/10.1016/j.apnum.2011.06.010).
[2] 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.

[3] S.C. Eisenstat, H.C. Elman, M.H. Schultz, Variational iterative methods for nonsymmetric systems of linear equations, SIAM J. Numer. Anal. 20 (1983) 345–357.
[4] R. Fletcher, Conjugate gradient methods for indefinite systems, Lect. Notes Math. 506 (1976) 73–89.
[5] D.R. Fokkema, G.L.G. Sleijpen, H.A. vander Vorst, Generalized conjugate gradient squared, J. Comput. Appl. Math. 71 (1994) 125–146.
[6] R.W. Freund, A transposed-free quasi-minimal residual algorithm for non-hermitian linear systems, SIAM J. Sci. Comput. 14 (1993) 470–482.
[7] M.H. Gutknecht, Variants of BiCGStab for matrices with complex spectrum, SIAM J. Sci. Comput. 14 (1993) 1020–1033.
[8] M.H. Gutknecht, Lanczos-type solvers for nonsymmetric linear systems of equations, Acta Numerica 6 (1997) 217–397.
[9] H. Rutishauser, Theory of gradient method, in: booktitle Refined Iterative Methods for Comutation of the Solution and the Eigenvalues of Self-Adjoint Value Problems, Mitt. Inst. angew. Math. ETH Zürich, Birkhäuser, Basel, 1959, pp. 24–49.
[10] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Stat. Comput. 14 (1993) 461–469.
[11] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[12] G.L.G. Sleijpen, D.R. Fokkema, BiCGstab(l) for solving linear equations involving unsymmetric matrices with complex spectrum, ETNA 1 (1993) 11–31.
[13] G.L.G. Sleijpen, van Gijzen, Exploiting BiCGstab(l) strategies to induce dimension reduction, SIAM J. Sci. Comput. 32 (2010) 2687–2709.
[14] G.L.G. Sleijpen, P. Sonneveld, M.B. van Gijzen, Bi-CGSTAB as induced dimension reduction method, Appl. Numer. Math. 60 (2010) 1100–1114.
[15] G.L.G. Sleijpen, H.A. vander Vorst, Maintaining convergence properties of BiCGstab methods in finite precision arithmetic, Numer. Algorithms 10 (1995) 203–223.
[16] P. Sonneveld, CGS, A fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Comput. 10 (1989) 36–52.
[17] P. Sonneveld, M.B. van Gijzen, IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems, SIAM J. Sci. Comput. 31 (2008) 1035–1062.
[18] E.L. Stiefel, Kernel polynomial in linear algebra and their numerical applications, in: Further Contributions to the Determination of Eigenvalues, NBS Applied Mathematics Series 49 (1958) 1–22.
[19] H.A. vander 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.
[20] S.L. Zhang, GPBi-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comp. 18 (1997) 537–551.