MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

GBi-CGSTAB(s, L): IDR(s) with Higher-Order Stabilization Polynomials

Masaaki TANIO and Masaaki SUGIHARA

METR 2009–16 April 2009

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.keisu.t.u-tokyo.ac.jp/research/techrep/index.html

GBi-CGSTAB(s, L): IDR(s) with Higher-Order Stabilization Polynomials

Masaaki TANIO and Masaaki SUGIHARA

Department of Mathematical Informatics
Graduate School of Information Science and Technology
University of Tokyo
m_sugihara@mist.i.u-tokyo.ac.jp

April 2009

Abstract

IDR(s) is now recognized as one of the most effective methods, often superior to other Krylov subspace methods, for large nonsymmetric linear systems of equations. In this paper we propose an improvement upon IDR(s) by incorporating a higher-order stabilization polynomial into IDR(s). The proposed algorithm, named GBi-CGSTAB(s, L), shares desirable features with both IDR(s) and Bi-CGSTAB(L).

1 Introduction

Recently there has been proposed a new method called IDR(s) [9, 10] for solving large nonsymmetric systems of equations $A\boldsymbol{x} = \boldsymbol{b}$. Here A is a given $N \times N$ matrix and $\boldsymbol{b}$ is a given vector. IDR(s) has a feature that it requires at most $N + N/s$ matrix-vector multiplications to compute an exact solution in exact arithmetic, while the Bi-CG based methods such as CGS [8, 13], BiCGSTAB [12, 13], BiCGSTAB(L) [5], GP-BiCG [13,14] require at most $2N$ matrix-vector multiplications. Numerical experiments also report that IDR(s) is competitive with or superior to most Bi-CG based methods. It is known, however, that IDR(s) is inferior to BiCGSTAB(L) ($L > 1$) when applied to equations with nearly skew-symmetric coefficient matrices. The weakness against skew-symmetry comes from the fact that the order of the stabilization polynomials in IDR(s) is one, whereas BiCGSTAB(L) uses higher order stabilization polynomials. As mentioned in [9, 10], to overcome this weakness it is natural to incorporate higher order stabilization polynomials into IDR(s).

A milestone in this direction is the Sleijpen–Sonneveld–van Gijzen paper [6]. As is well known, the idea behind the IDR(s), so-called IDR principle, is different from that of the standard Krylov methods, which makes it difficult to incorporate a higher order stabilization polynomial into IDR(s). The paper shows that IDR(s) can also be put into the standard framework of the Krylov method. To be specific, the paper proposes a variant of Bi-CG with multiple shadow residuals and the first order stabilization polynomials, and shows that it is mathematically equivalent to IDR(s).

In this paper we first present another variant of Bi-CG with multiple shadow residuals, called GBi-CG(s), which is different from that introduced in [6]. Then we incorporate L-th order stabilization polynomials into this variant to obtain the method we propose in the paper. The derivation of this method is parallel to that of Bi-CGSTAB(L), which is obtained from Bi-CG with L-th order stabilization polynomials. The proposed algorithm is named GBi-CGSTAB(s, L). This method with $L = 1$ is mathematically equivalent to IDR(s).

This paper is organized as follows. Section 2 is a preliminary section, describing Bi-CG and Bi-CGSTAB(L). In Section 3, we first explain the variant of of Bi-CG due to Sleijpen–Sonneveld–van Gijzen, and then introduce GBi-CG(s), another variant that serves as the basis of our method. In Section 4, we derive GBi-CGSTAB(s, L), the method we propose in this paper. Section 5 shows numerical results and Section 6 concludes the paper.

Notation and Definition

Notation 1 For $\widetilde{R} \in \mathbb{C}^{N\times s}$ and $\boldsymbol{v} \in \mathbb{C}^N$ we write "$\boldsymbol{v} \perp \widetilde{R}$" to mean that $\boldsymbol{v}$ is orthogonal to all column vectors of $\widetilde{R}$. We denote by $\widetilde{R}^{\perp}$ the (maximal) linear subspace that is orthogonal to all column vectors $\widetilde{R}$.

Notation 2 For an $N \times s$ matrix C and for $j = 1, 2, \ldots, s$ we denote by $C\boldsymbol{e}_j$ the j-th column vector of C.

Notation 3 Let C and $\widetilde{R}$ be $N \times s$ matrices, and $\boldsymbol{v}$ an N dimensional vector. By the expression:

$$\boldsymbol{u} = \boldsymbol{v} - C\boldsymbol{\beta} \ \text{ such that } \boldsymbol{u} \perp \widetilde{R},$$

we mean that (i) we choose an s dimensional coefficient vector $\boldsymbol{\beta}$ such that $\boldsymbol{v} - C\boldsymbol{\beta} \perp \widetilde{R}$ and (ii) we update the vector $\boldsymbol{u}$ by computing $\boldsymbol{u} = \boldsymbol{v} - C\boldsymbol{\beta}$. More generally, if A is an $N \times N$ matrix, the expression:

$$\boldsymbol{u} = \boldsymbol{v} - C\boldsymbol{\beta} \ \text{ such that } A\boldsymbol{u} \perp \widetilde{R},$$

means that (i) we choose $\boldsymbol{\beta}$ such that $A\boldsymbol{v} - AC\boldsymbol{\beta} \perp \widetilde{R}$ and (ii) we update the vector $\boldsymbol{u}$ by computing $\boldsymbol{u} = \boldsymbol{v} - C\boldsymbol{\beta}$.

Definition 1 For $B \in \mathbb{C}^{N\times N}$, $\widetilde{R} \in \mathbb{C}^{N\times s}$ and a natural number k, the linear subspace $\mathcal{K}_k(B, \widetilde{R})$ is defined as

$$\mathcal{K}_k(B, \widetilde{R}) = \left\{ \sum_{j=0}^{k-1} B^j \widetilde{R} \boldsymbol{\gamma}_j \mid \boldsymbol{\gamma}_j \in \mathbb{C}^s \right\}. \tag{1}$$

We refer to $\mathcal{K}_k(B, \widetilde{R})$ as the block Krylov subspace [1]. Note that, if $s = 1$, $\mathcal{K}_k(B, \widetilde{R})$ is the Krylov subspace in the usual sense.

2 Preliminaries: Bi-CG and Bi-CGSTAB(L)

2.1 Bi-CG

The Bi-CG algorithm [4] generates approximate solutions $\boldsymbol{x}_k$ according to the following recurrences:

$$\begin{cases} \boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{u}_k \ \text{ such that } \ \boldsymbol{r}_{k+1} \perp \tilde{\boldsymbol{r}}_k, \\ \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{u}_k, \\ \tilde{\boldsymbol{r}}_{k+1} = \tilde{\boldsymbol{r}}_k - \alpha_k A^* \tilde{\boldsymbol{u}}_k, \\ \boldsymbol{u}_{k+1} = \boldsymbol{r}_{k+1} - \beta_{k+1}\boldsymbol{u}_k \ \text{ such that } \ A\boldsymbol{u}_{k+1} \perp \tilde{\boldsymbol{r}}_k, \\ \tilde{\boldsymbol{u}}_{k+1} = \tilde{\boldsymbol{r}}_{k+1} - \beta_{k+1}\tilde{\boldsymbol{u}}_k. \end{cases} \tag{2}$$

A crucial property is that the residuals $\boldsymbol{r}_k$ and the auxiliary vectors $A\boldsymbol{u}_k$ satisfy global bi-orthogonality:

$$\boldsymbol{r}_k, A\boldsymbol{u}_k \perp \mathcal{K}_k(A^*, \tilde{\boldsymbol{r}}_0). \tag{3}$$

By introducing a vector $\tilde{\boldsymbol{s}}_k = \tilde{q}_k(A^*)\tilde{\boldsymbol{r}}_0$ with a polynomial $\tilde{q}_k(t)$ of degree k, and by replacing $\tilde{\boldsymbol{r}}_k$ with $\tilde{\boldsymbol{s}}_k$, the Bi-CG algorithm can be rewritten as follows:

$$\begin{cases} \boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{u}_k \ \text{ such that } \ \boldsymbol{r}_{k+1} \perp \tilde{\boldsymbol{s}}_k, \\ \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{u}_k, \\ \boldsymbol{u}_{k+1} = \boldsymbol{r}_{k+1} - \beta_{k+1}\boldsymbol{u}_k \ \text{ such that } \ A\boldsymbol{u}_{k+1} \perp \tilde{\boldsymbol{s}}_k. \end{cases} \tag{4}$$

In exact arithmetic, the residuals $\boldsymbol{r}_k$ and the approximate solutions $\boldsymbol{x}_k$ generated by (2) coincide with those by (4). In (4) the coefficients α_k and β_{k+1} are computed as follows:

$$\begin{cases} \alpha_k &= \dfrac{(\boldsymbol{r}_k, \tilde{\boldsymbol{s}}_k)}{(A\boldsymbol{u}_k, \tilde{\boldsymbol{s}}_k)}, \\ \beta_{k+1} &= \dfrac{(A\boldsymbol{r}_{k+1}, \tilde{\boldsymbol{s}}_k)}{(A\boldsymbol{u}_k, \tilde{\boldsymbol{s}}_k)} = \dfrac{\tau_k}{\tau_{k+1}} \dfrac{(\boldsymbol{r}_{k+1}, \tilde{\boldsymbol{s}}_{k+1})}{(A\boldsymbol{u}_k, \tilde{\boldsymbol{s}}_k)}, \end{cases} \tag{5}$$

where τ_k is a scalar representing the leading coefficient of the polynomial $\tilde{q}_k(t)$.

Recently, a number of improvements of the Bi-CG algorithm have been proposed on the basis of the following two observations: (i) the degree of freedom in the choice of the shadow residuals $\tilde{\boldsymbol{s}}_k$ can profitably be exploited, and (ii) the variable $\tilde{\boldsymbol{s}}_k$ appears only in the formulas for the coefficients α_k and β_{k+1}, which can be avoided by using the relations:

$$(\boldsymbol{r}_k, \tilde{\boldsymbol{s}}_k) = (\boldsymbol{r}_k, \tilde{q}_k(A^*)\tilde{\boldsymbol{r}}_0) = (\tilde{q}_k(A)\boldsymbol{r}_k, \tilde{\boldsymbol{r}}_0), \quad (A\boldsymbol{u}_k, \tilde{\boldsymbol{s}}_k) = (A\tilde{q}_k(A)\boldsymbol{u}_k, \tilde{\boldsymbol{r}}_0). \tag{6}$$

It is indeed possible [5, 8, 12, 14] to generalize the Bi-CG method according to the following strategies.

(i) Instead of $\boldsymbol{r}_k$, $\boldsymbol{u}_k$, $\boldsymbol{x}_k$, and $\tilde{\boldsymbol{s}}_k$, we update the transformed residual $\tilde{q}_k(A)\boldsymbol{r}_k$, the auxiliary vector $\tilde{q}_k(A)\boldsymbol{u}_k$ and a vector $\boldsymbol{x}'_k$ representing the approximate solution for which $\tilde{q}_k(A)\boldsymbol{r}_k$ is the residual. The initial shadow residual $\tilde{\boldsymbol{r}}_0$ is kept throughout, without being updated.

(ii) The polynomial $\tilde{q}_k(t)$, often called the stabilization polynomial, is chosen so that the norm of the transformed residual $\tilde{q}_k(A)\boldsymbol{r}_k$ may be smaller.

(iii) The coefficients α_k and β_{k+1} are computed by using the relations (5) and (6).

2.2 Bi-CGSTAB(L)

We review the Bi-CGSTAB(L) algorithm. It is derived from Bi-CG with a stabilization polynomial that is a product of polynomials of degree L.

Suppose that we are given a sequence of polynomials $p_i(t)$ of degree L for $i = 1, 2, \ldots$ satisfying $p_i(0) = 1$. For $k = mL$, the k-th stabilization polynomial $Q_k(t)$ is defined as $Q_k(t) = p_m(t) \cdots p_2(t)p_1(t)$. Note that $Q_k(t)$ is of degree k. We also say that $Q_k(t)$ is an L-th order stabilization polynomial to mean that each factor $p_i(t)$ is a polynomial of degree L.

We then set the residual $\boldsymbol{r}_k$ and the auxiliary vector $\boldsymbol{u}_{k-1}$ as

$$\boldsymbol{r}_k = Q_k(A)\boldsymbol{r}_k^{\mathrm{B}}, \quad \boldsymbol{u}_{k-1} = Q_k(A)\boldsymbol{u}_{k-1}^{\mathrm{B}},$$

where $\boldsymbol{r}_k^{\mathrm{B}}$ and $\boldsymbol{u}_{k-1}^{\mathrm{B}}$ are the residual and the auxiliary vector in Bi-CG, denoted as $\boldsymbol{r}_k$ and $\boldsymbol{u}_{k-1}$ in Section 2.1. Furthermore, we define approximate solutions $\boldsymbol{x}_k$ and $\hat{\boldsymbol{x}}_k^{(i)}$ as those vectors which respectively satisfy

$$\boldsymbol{b} - A\boldsymbol{x}_k = \boldsymbol{r}_k = Q_k(A)\boldsymbol{r}_k^{\mathrm{B}}, \quad \boldsymbol{b} - A\hat{\boldsymbol{x}}_k^{(i)} = \boldsymbol{r}_{k+i} = Q_k(A)\boldsymbol{r}_{k+i}^{\mathrm{B}}.$$

In an iteration of Bi-CGSTAB(L), the vectors $\boldsymbol{r}_k, \boldsymbol{u}_k$ and $\boldsymbol{x}_k$ are updated to $\boldsymbol{r}_{k+L}$, $\boldsymbol{u}_{k+L-1}$ and $\boldsymbol{x}_{k+L}$, respectively. The iteration consists of the Bi-CG part and the MR part (Figure 1), where "MR" stands for "Minimal Residual."

Bi-CG part: In this part, $Q_k\boldsymbol{r}_k^{\mathrm{B}}$, $Q_k\boldsymbol{u}_{k-1}^{\mathrm{B}}$ and $\boldsymbol{x}_k$ are given. Then the i-th step, to be specified later, is performed successively for $i = 0, 1, \ldots, L-1$. Finally, vectors $Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}}, \ldots, A^L Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}}$; $Q_k\boldsymbol{u}_{k+L-1}^{\mathrm{B}}, \ldots, A^L Q_k\boldsymbol{u}_{k+L-1}^{\mathrm{B}}$, and $\hat{\boldsymbol{x}}_k^{(L)}$ are computed. An execution of the Bi-CG part can be done with $2L$ matrix-vector multiplications.

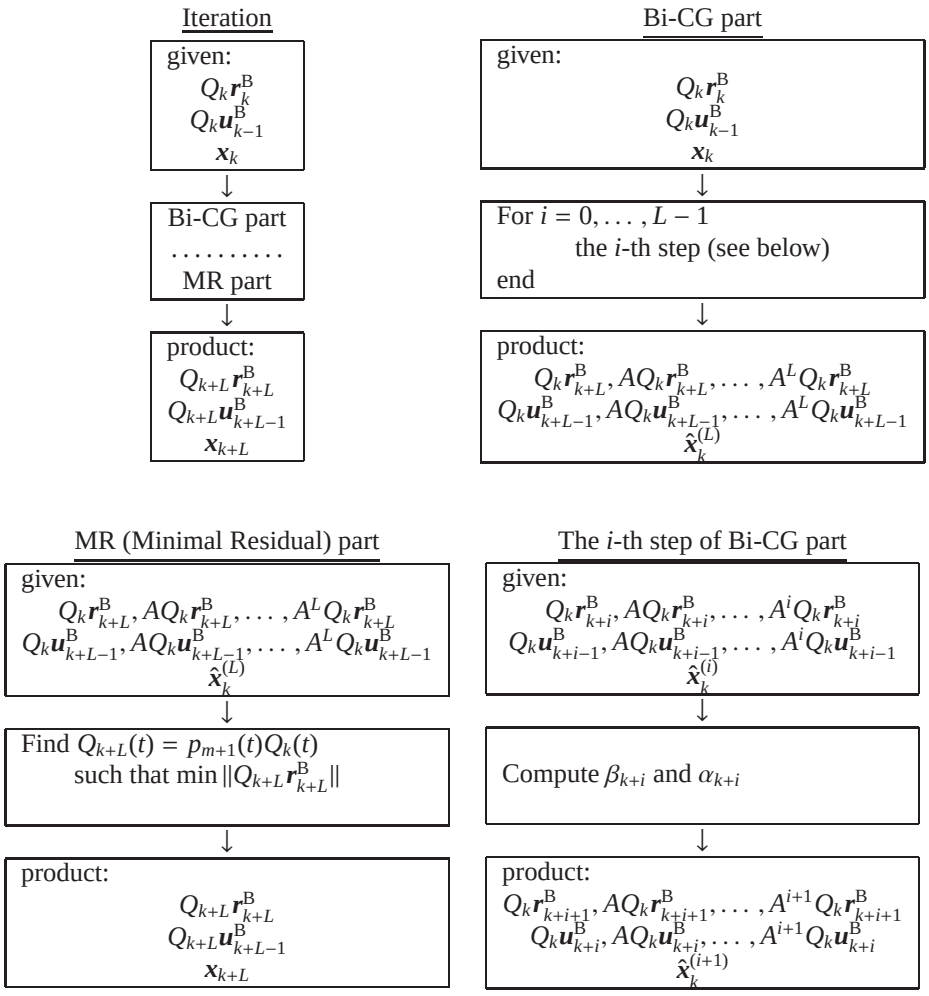

Figure 1: An iteration of Bi-CGSTAB(L)

MR part: In this part, by using the output of Bi-CG part, we choose the parameters $\gamma_1^{(m+1)}, \gamma_2^{(m+1)}, \ldots, \gamma_L^{(m+1)}$ in the polynomial $p_{m+1}(t) = 1 - \sum_{i=1}^{L} \gamma_i^{(m+1)}$ such that the norm of the new residual $\boldsymbol{r}_{k+L}$ is minimum. When the polynomial $p_{m+1}(t)$ is determined, we make the following updates on the basis of the relation $Q_{k+L}(t) = p_{m+1}(t)Q_k(t)$:

$$Q_{k+L}\boldsymbol{r}_{k+L}^{\mathrm{B}} = Q_k\boldsymbol{r}_{k+L}^{\mathrm{B}} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k \boldsymbol{r}_{k+L}^{\mathrm{B}}, \tag{7}$$

$$Q_{k+L}\boldsymbol{u}_{k+L-1}^{\mathrm{B}} = Q_k\boldsymbol{u}_{k+L-1}^{\mathrm{B}} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k \boldsymbol{u}_{k+L-1}^{\mathrm{B}}, \tag{8}$$

$$\boldsymbol{x}_{k+L} = \hat{\boldsymbol{x}}_k^{(L)} + \sum_{i=0}^{L-1} \gamma_i^{(m+1)} A^i Q_k \boldsymbol{r}_{k+L}^{\mathrm{B}}. \tag{9}$$

It is mentioned that in practical implementations the residual $Q_{k+L}\boldsymbol{r}_{k+L}^{\mathrm{B}}$ is computed with the aid the modified Gram-Schmidt orthogonalization process; see [5] for detail and Algorithm 1 below.

Detail of the Bi-CG part: We now describe the Bi-CG part in Bi-CGSTAB(L) at some length, as it is useful for our exposition of our algorithm in Section 4.2.

The Bi-CG part consists of L steps, from the 0-th to the $(L-1)$-st step. The i-th step in the Bi-CG part is shown in Figure 1 (bottom-right), where $i = 0, 1, \ldots, L-1$. Given $A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{B}}$, $A^j Q_k \boldsymbol{u}_{k+i-1}^{\mathrm{B}} (j = 0, \ldots, i)$ and $\hat{\boldsymbol{x}}_k^{(i)}$, the i-th step computes $A^j Q_k \boldsymbol{r}_{k+i+1}^{\mathrm{B}}$, $A^j Q_k \boldsymbol{u}_{k+i}^{\mathrm{B}}$ $(j = 0, \ldots, i+1)$ and $\hat{\boldsymbol{x}}_k^{(i+1)}$.

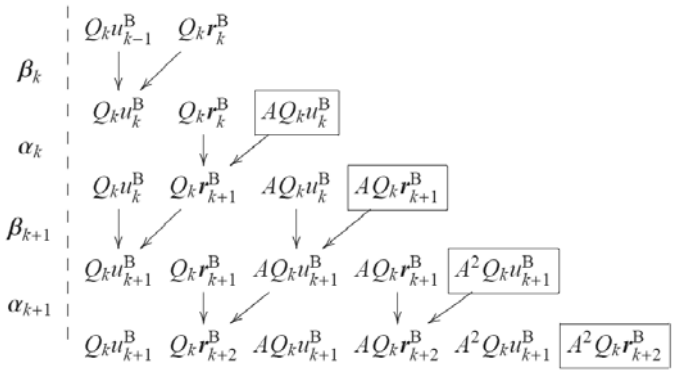

Figure 2: Flowchart of the Bi-CG part of Bi-CGSTAB(2)

We refer to Figure 2, which illustrates the Bi-CG part in the case of $L = 2$. The computation proceeds from row to row, replacing vectors from the previous row by vectors on the next row. Vector updates derived from the Bi-CG relations (2) are indicated by arrows in Figure 2.

The i-th step for $i = 0$ consists of the first row to the third in Figure 2. In the transition from the first row to the second, after the computation of the coefficient

β_k in Bi-CG, we update the vector such that $Q_k\boldsymbol{u}_k^{\mathrm{B}} = Q_k\boldsymbol{r}_k^{\mathrm{B}} - \beta_k Q_k\boldsymbol{u}_{k-1}^{\mathrm{B}}$. Then the vector $AQ_k\boldsymbol{u}_k^{\mathrm{B}}$ is obtained by multiplication by the matrix A. From the second to the third, after the computation of the coefficient α_k in Bi-CG, we compute the vector such that $Q_k\boldsymbol{r}_{k+1}^{\mathrm{B}} = Q_k\boldsymbol{r}_k^{\mathrm{B}} - \alpha_k AQ_k\boldsymbol{u}_k^{\mathrm{B}}$. Moreover, $\boldsymbol{x}_k$ is replaced to $\hat{\boldsymbol{x}}_k^{(1)}$ as $\hat{\boldsymbol{x}}_k^{(1)} = \boldsymbol{x}_k + \alpha_k Q_k\boldsymbol{u}_k^{\mathrm{B}}$ (this update is not indicated in Figure 2). Then the vector $AQ_k\boldsymbol{r}_{k+1}^{\mathrm{B}}$ is obtained by multiplication by the matrix A.

Next, we show the i-th step for $i = 1$ in a similar way. From the third row to the fourth, after the computation of the coefficient β_{k+1}, we update the vectors such that $Q_k\boldsymbol{u}_{k+1}^{\mathrm{B}} = Q_k\boldsymbol{r}_{k+1}^{\mathrm{B}} - \beta_{k+1}Q_k\boldsymbol{u}_k^{\mathrm{B}}$ and $AQ_k\boldsymbol{u}_{k+1}^{\mathrm{B}} = AQ_k\boldsymbol{r}_{k+1}^{\mathrm{B}} - \beta_{k+1}AQ_k\boldsymbol{u}_k^{\mathrm{B}}$. Then the vector $A^2Q_k\boldsymbol{u}_{k+1}^{\mathrm{B}}$ is obtained by multiplication by the matrix A. From the fourth to the fifth, after the computation of the coefficient α_{k+1}, we update the vectors such that $Q_k\boldsymbol{r}_{k+2}^{\mathrm{B}} = Q_k\boldsymbol{r}_{k+1}^{\mathrm{B}} - \alpha_{k+1}AQ_k\boldsymbol{u}_{k+1}^{\mathrm{B}}$ and $AQ_k\boldsymbol{r}_{k+2}^{\mathrm{B}} = AQ_k\boldsymbol{r}_{k+1}^{\mathrm{B}} - \alpha_{k+1}A^2Q_k\boldsymbol{u}_{k+1}^{\mathrm{B}}$. Moreover, $\hat{\boldsymbol{x}}_k^{(1)}$ is replaced to $\hat{\boldsymbol{x}}_k^{(2)}$ as $\hat{\boldsymbol{x}}_k^{(2)} = \hat{\boldsymbol{x}}_k^{(1)} + \alpha_{k+1}Q_k\boldsymbol{u}_{k+1}^{\mathrm{B}}$ (this update is not indicated in Figure 2). Then the vector $A^2Q_k\boldsymbol{r}_{k+2}^{\mathrm{B}}$ is obtained by multiplication by the matrix A. Finally, the vectors $Q_k\boldsymbol{r}_{k+2}^{\mathrm{B}}$, $AQ_k\boldsymbol{r}_{k+2}^{\mathrm{B}}$, $A^2Q_k\boldsymbol{r}_{k+2}^{\mathrm{B}}$, $Q_k\boldsymbol{u}_{k+1}^{\mathrm{B}}$, $AQ_k\boldsymbol{u}_{k+1}^{\mathrm{B}}$, $A^2Q_k\boldsymbol{u}_{k+1}^{\mathrm{B}}$ and $\hat{\boldsymbol{x}}_k^{(2)}$ are given. The Bi-CG part ends at this point if $L = 2$.

Now, we show the i-th step for a general i. It is assumed for the moment that the coefficients for Bi-CG in (2) can be computed, which will be discussed later. First, after the computation of the coefficient β_{k+i} in Bi-CG, by the relation (2), we update $A^jQ_k\boldsymbol{u}_{k+i-1}^{\mathrm{B}} \to A^jQ_k\boldsymbol{u}_{k+i}^{\mathrm{B}}$ $(j = 0, 1, \ldots, i)$ such that $A^jQ_k\boldsymbol{u}_{k+i}^{\mathrm{B}} = A^jQ_k\boldsymbol{r}_{k+i}^{\mathrm{B}} - \beta_{k+i}A^jQ_k\boldsymbol{u}_{k+i-1}^{\mathrm{B}}$. Then the vector $A^{i+1}Q_k\boldsymbol{u}_{k+i}^{\mathrm{B}}$ is obtained by multiplication by the matrix A. Second, after the computation of the coefficient α_{k+i} in Bi-CG, we use the relation (2) to update $A^jQ_k\boldsymbol{r}_{k+i}^{\mathrm{B}} \to A^jQ_k\boldsymbol{r}_{k+i+1}^{\mathrm{B}}$ such that $A^jQ_k\boldsymbol{r}_{k+i+1}^{\mathrm{B}} = A^jQ_k\boldsymbol{r}_{k+i}^{\mathrm{B}} - \alpha_{k+i}A^{j+1}Q_k\boldsymbol{u}_{k+i}^{\mathrm{B}}$ $(j = 0, 1, \ldots, i)$. Then the vector $A^{i+1}Q_k\boldsymbol{r}_{k+i+1}^{\mathrm{B}}$ is obtained by multiplication by the matrix A. Moreover we update $\hat{\boldsymbol{x}}_k^{(i)} \to \hat{\boldsymbol{x}}_k^{(i+1)}$ such that $\hat{\boldsymbol{x}}_k^{(i+1)} = \hat{\boldsymbol{x}}_k^{(i)} + \alpha_{k+i}Q_k\boldsymbol{u}_{k+i}^{\mathrm{B}}$. Finally the vectors $A^jQ_k\boldsymbol{r}_{k+i+1}^{\mathrm{B}}, A^jQ_k\boldsymbol{u}_{k+i}^{\mathrm{B}}$ $(j = 0, 1, \ldots, i+1)$ and $\hat{\boldsymbol{x}}_k^{(i+1)}$ are output and the i-th step is completed.

It remains to explain how to compute the coefficients α_{k+i} and β_{k+i}.

Computation of α_{k+i}

Before the computation of the coefficient α_{k+i}, we are given $A^jQ_k\boldsymbol{r}_{k+i}^{\mathrm{B}}$ $(j = 0, \ldots, i)$ and $A^jQ_k\boldsymbol{u}_{k+i}^{\mathrm{B}}$ $(j = 0, \ldots, i+1)$. Consider the vector $\tilde{\mathbf{s}}_{k+i}$ in (4) for Bi-CG, though it is not available at hand. For the choice of $\tilde{\mathbf{s}}_{k+i} = (A^*)^iQ_k(A^*)\tilde{\boldsymbol{r}}_0$, the coefficient α_{k+i} is calculated by (5) and (6) as

$$\alpha_{k+i} = \frac{(\boldsymbol{r}_{k+i}^{\mathrm{B}}, (A^*)^iQ_k(A^*)\tilde{\boldsymbol{r}}_0)}{(A\boldsymbol{u}_{k+i}^{\mathrm{B}}, (A^*)^iQ_k(A^*)\tilde{\boldsymbol{r}}_0)} = \frac{(A^iQ_k(A)\boldsymbol{r}_{k+i}^{\mathrm{B}}, \tilde{\boldsymbol{r}}_0)}{(A^{i+1}Q_k(A)\boldsymbol{u}_{k+i}^{\mathrm{B}}, \tilde{\boldsymbol{r}}_0)}.$$

This expression affords a computable formula for α_{k+i}, since the vectors $A^iQ_k\boldsymbol{r}_{k+i}^{\mathrm{B}}$, $A^{i+1}Q_k\boldsymbol{u}_{k+i}^{\mathrm{B}}$ and $\tilde{\boldsymbol{r}}_0$ are available.

For the efficient computation of α_{k+i} as well as β_{k+i} below, it is convenient to employ auxiliary variables $\rho_i^{(k)} = (A^iQ_k\boldsymbol{r}_{k+i}^{\mathrm{B}}, \tilde{\boldsymbol{r}}_0)$ and $\gamma_{i+1}^{(k)} = (A^{i+1}Q_k\boldsymbol{u}_{k+i}^{\mathrm{B}}, \tilde{\boldsymbol{r}}_0)$. We then have $\alpha_{k+i} = \rho_i^{(k)}/\gamma_{i+1}^{(k)}$, which will be used in practical implementations.

Computation of β_{k+i}

For $i > 0$: Before the computation of the coefficient β_{k+i}, we are given $A^j Q_k \boldsymbol{r}^{\mathrm{B}}_{k+i}$ ($j = 0, \ldots, i$) and $A^j Q_k \boldsymbol{u}^{\mathrm{B}}_{k+i-1}$ ($j = 0, \ldots, i$). Consider the vector $\tilde{\boldsymbol{s}}_{k+i-1}$ in (4) for Bi-CG. For the choice of $\tilde{\boldsymbol{s}}_{k+i-1} = (A^*)^{i-1} Q_k(A^*)\tilde{\boldsymbol{r}}_0$, the coefficient β_{k+i} is calculated by (5) and (6), as

$$\beta_{k+i} = \frac{(A\boldsymbol{r}^{\mathrm{B}}_{k+i}, (A^*)^{i-1} Q_k(A^*)\tilde{\boldsymbol{r}}_0)}{(A\boldsymbol{u}^{\mathrm{B}}_{k+i-1}, (A^*)^{i-1} Q_k(A^*)\tilde{\boldsymbol{r}}_0)} = \frac{(A^i Q_k(A)\boldsymbol{r}^{\mathrm{B}}_{k+i}, \tilde{\boldsymbol{r}}_0)}{(A^i Q_k(A)\boldsymbol{u}^{\mathrm{B}}_{k+i-1}, \tilde{\boldsymbol{r}}_0)}.$$

This expression affords a computable formula for β_{k+i}. Furthermore, by rewriting the right-hand side above we obtain

$$\beta_{k+i} = \frac{(A^{i-1} Q_k(A)\boldsymbol{r}^{\mathrm{B}}_{k+i-1}, \tilde{\boldsymbol{r}}_0)}{(A^i Q_k(A)\boldsymbol{u}^{\mathrm{B}}_{k+i-1}, \tilde{\boldsymbol{r}}_0)} \cdot \frac{(A^i Q_k(A)\boldsymbol{r}^{\mathrm{B}}_{k+i}, \tilde{\boldsymbol{r}}_0)}{(A^{i-1} Q_k(A)\boldsymbol{r}^{\mathrm{B}}_{k+i-1}, \tilde{\boldsymbol{r}}_0)} = \alpha_{k+i-1} \frac{\rho_i^{(k)}}{\rho_{i-1}^{(k)}}.$$

This expression is more suitable for practical implementations.

For $i = 0$: Before the computation of the coefficient β_k, we have the vectors $A^L Q_{k-L} \boldsymbol{r}^{\mathrm{B}}_k$, $Q_k \boldsymbol{r}^{\mathrm{B}}_k$ and $A^L Q_{k-L} \boldsymbol{u}^{\mathrm{B}}_{k-1}$, which are given in the previous iteration of Bi-CGSTAB(L). Consider the vector $\tilde{\boldsymbol{s}}_{k-1}$ in (4) for Bi-CG. For the choice of $\tilde{\boldsymbol{s}}_{k-1} = (A^*)^{L-1} Q_{k-L}(A^*)\tilde{\boldsymbol{r}}_0$, the coefficient β_k is calculated by (5) and (6) as

$$\beta_k = \frac{(A\boldsymbol{r}^{\mathrm{B}}_k, (A^*)^{L-1} Q_{k-L}(A^*)\tilde{\boldsymbol{r}}_0)}{(A\boldsymbol{u}^{\mathrm{B}}_{k-1}, (A^*)^{L-1} Q_{k-L}(A^*)\tilde{\boldsymbol{r}}_0)} = \frac{(A^L Q_{k-L}(A)\boldsymbol{r}^{\mathrm{B}}_k, \tilde{\boldsymbol{r}}_0)}{(A^L Q_{k-L}(A)\boldsymbol{u}^{\mathrm{B}}_{k-1}, \tilde{\boldsymbol{r}}_0)}.$$

This expression affords a computable formula for β_k. For practical implementations, we rewrite the right-hand side above to obtain

$$\begin{aligned}
\beta_k &= \frac{(A^{L-1} Q_{k-L}(A)\boldsymbol{r}^{\mathrm{B}}_{k-1}, \tilde{\boldsymbol{r}}_0)}{(A^L Q_{k-L}(A)\boldsymbol{u}^{\mathrm{B}}_{k-1}, \tilde{\boldsymbol{r}}_0)} \cdot \frac{(A^L Q_{k-L}(A)\boldsymbol{r}^{\mathrm{B}}_k, \tilde{\boldsymbol{r}}_0)}{(A^{L-1} Q_{k-L}(A)\boldsymbol{r}^{\mathrm{B}}_{k-1}, \tilde{\boldsymbol{r}}_0)} \\
&= \alpha_{k-1} \cdot \frac{(A^L Q_{k-L}(A)\boldsymbol{r}^{\mathrm{B}}_k, \tilde{\boldsymbol{r}}_0)}{(A^{L-1} Q_{k-L}(A)\boldsymbol{r}^{\mathrm{B}}_{k-1}, \tilde{\boldsymbol{r}}_0)} \\
&= \alpha_{k-1} \cdot \frac{(Q_k \boldsymbol{r}^{\mathrm{B}}_k, \tilde{\boldsymbol{r}}_0)/(-\gamma_L^{(m)})}{(A^{L-1} Q_{k-L}(A)\boldsymbol{r}^{\mathrm{B}}_{k-1}, \tilde{\boldsymbol{r}}_0)} = \alpha_{k-1} \cdot \frac{\rho_0^{(k)}/(-\gamma_L^{(m)})}{\rho_{L-1}^{(k-L)}}.
\end{aligned}$$

Here we have used the relation $-\gamma_L^{(m)}(A^L Q_{k-L}\boldsymbol{r}^{\mathrm{B}}_k, \tilde{\boldsymbol{r}}_0) = (Q_k \boldsymbol{r}^{\mathrm{B}}_k, \tilde{\boldsymbol{r}}_0)$, which follows from (3) and (7).

The following is the Bi-CGSTAB(L) algorithm in full detail:

Algorithm 1: Bi-CGSTAB(L) algorithm [5]

1. $k = -L$
2. choose $\boldsymbol{x}_0$ and $\boldsymbol{r}_0^*$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$

3. $\boldsymbol{u}_{-1} = 0,\ \rho_0 = 1,\ \alpha = 0,\ \omega = 1$
4. repeat until $\|\boldsymbol{r}_{k+L}\| < \varepsilon$ (tolerance)
5. $k = k + L$
6. $\hat{\boldsymbol{u}}_0 = \boldsymbol{u}_{k-1},\ \hat{\boldsymbol{r}}_0 = \boldsymbol{r}_k,\ \hat{\boldsymbol{x}}_0 = \boldsymbol{x}_k,\ \rho_0 = -\omega\rho_0$
7. for $j = 0, 1, \ldots, L-1$
8. $\rho_1 = (\hat{\boldsymbol{r}}_j, \boldsymbol{r}_0^*),\ \beta = \alpha\frac{\rho_1}{\rho_0},\ \rho_0 = \rho_1$
9. $\hat{\boldsymbol{u}}_i = \hat{\boldsymbol{r}}_i - \beta\hat{\boldsymbol{u}}_i\ (i = 0, 1, \ldots, j)$
10. $\hat{\boldsymbol{u}}_{j+1} = A\hat{\boldsymbol{u}}_j,\ \gamma = (\hat{\boldsymbol{u}}_{j+1}, \boldsymbol{r}_0^*),\ \alpha = \frac{\rho_0}{\gamma}$
11. $\hat{\boldsymbol{r}}_i = \hat{\boldsymbol{r}}_i - \alpha\hat{\boldsymbol{u}}_{i+1}\ (i = 0, 1, \ldots, j)$
12. $\hat{\boldsymbol{r}}_{j+1} = A\hat{\boldsymbol{r}}_j,\ \hat{\boldsymbol{x}}_0 = \hat{\boldsymbol{x}}_0 + \alpha\hat{\boldsymbol{u}}_0$
13. end for
14. for $j = 1, 2, \ldots, L$
15. $\tau_{ij} = \frac{1}{\sigma_i}(\hat{\boldsymbol{r}}_i, \hat{\boldsymbol{r}}_j),\ \hat{\boldsymbol{r}}_j = \hat{\boldsymbol{r}}_j - \tau_{ij}\hat{\boldsymbol{r}}_i\ \ (i = 1, 2, \ldots, j-1)$
16. $\sigma_j = (\hat{\boldsymbol{r}}_j, \hat{\boldsymbol{r}}_j),\ \gamma'_j = \frac{1}{\sigma_j} = (\hat{\boldsymbol{r}}_j, \hat{\boldsymbol{r}}_0)$
17. end for
18. $\gamma_L = \gamma'_L,\ \omega = \gamma_L$
19. $\gamma_j = \gamma'_j - \sum_{i=j+1}^{L} \tau_{ji}\gamma_i\ \ (j = L-1, \ldots, 1)$
20. $\gamma''_j = \gamma_{j+1} + \sum_{i=j+1}^{L-1} \tau_{ji}\gamma_{i+1}\ \ (j = 1, \ldots, L-1)$
21. $\hat{\boldsymbol{x}}_0 = \hat{\boldsymbol{x}}_0 + \gamma_1\hat{\boldsymbol{r}}_0,\ \hat{\boldsymbol{r}}_0 = \hat{\boldsymbol{r}}_0 - \gamma'_L\hat{\boldsymbol{r}}_L,\ \hat{\boldsymbol{u}}_0 = \hat{\boldsymbol{u}}_0 - \gamma_L\hat{\boldsymbol{u}}_L$
22. $\hat{\boldsymbol{u}}_0 = \hat{\boldsymbol{u}}_0 - \gamma_j\hat{\boldsymbol{u}}_j\ (j = 1, \ldots, L-1)$
23. $\hat{\boldsymbol{x}}_0 = \hat{\boldsymbol{x}}_0 + \gamma''_j\hat{\boldsymbol{r}}_j,\ \hat{\boldsymbol{r}}_0 = \hat{\boldsymbol{r}}_0 - \gamma'_j\hat{\boldsymbol{r}}_j\ (j = 1, \ldots, L-1)$
24. $\boldsymbol{u}_{k+L-1} = \hat{\boldsymbol{u}}_0,\ \boldsymbol{r}_{k+L} = \hat{\boldsymbol{r}}_0,\ \boldsymbol{x}_{k+L} = \hat{\boldsymbol{x}}_0$
25. end repeat

3 Bi-CG with Multiple Shadow Residuals

In the Bi-CG algorithm the residual $\boldsymbol{r}_k$ is updated so that the condition

$$\boldsymbol{r}_k \perp \mathcal{K}_k(A^*, \widetilde{r}_0)$$

is satisfied. Similarly, in an algorithm with multiple shadow residuals the residual $\boldsymbol{r}_k$ is updated on the basis of the condition:

$$\boldsymbol{r}_k \perp \mathcal{K}_k(A^*, \widetilde{R}_0), \tag{10}$$

where $\widetilde{R}_0$ is an $N \times s$ matrix consisting of s column vectors chosen at the beginning of the algorithm.

We show two such algorithms below. The first is due to Sleijpen–Sonneveld–van Gijzen [6], and the second is ours, which will be improved in Section 4 to the proposed algorithm of the present paper.

3.1 Bi-CG(s)

The algorithm of [6] is described here as Algorithm 2 below. For convenience of reference we name it Bi-CG(s), although no name was given in [6]. For the condition (10) it employs an $N \times s$ matrix U_k, which consists of multiple auxiliary vectors corresponding to the auxiliary vector $\boldsymbol{u}_k$ in Bi-CG. It also uses an $N \times s$ matrix $\widetilde{R}_k$ such that, for any $K \geq 1$, the column vectors of $\widetilde{R}_0, \widetilde{R}_1, \ldots, \widetilde{R}_{K-1}$ span the subspace $\mathcal{K}_K(A^*, \widetilde{R}_0)$. The simplest choice is $\widetilde{R}_k = (A^*)^k \widetilde{R}_0$. Another possibility is $\widetilde{R}_k = \Phi_k(A^*)\widetilde{R}_0$ with a polynomial $\Phi_k(t)$ of degree k.

Algorithm 2: Bi-CG(s) algorithm [6]

choose $\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
set $N \times s$ matrix $U_0 = [\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{s-1}\boldsymbol{r}_0]$
$k = 0$
repeat until $\|\boldsymbol{r}_k\| < \varepsilon$ (tolerance)
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - AU_k\boldsymbol{\alpha}_k$ such that $\boldsymbol{r}_{k+1} \perp \widetilde{R}_k$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + U_k\boldsymbol{\alpha}_k$
  $\boldsymbol{v} = \boldsymbol{r}_{k+1}$
  for $j = 1, \ldots, s$
    $U_{k+1}\boldsymbol{e}_j = \boldsymbol{v} - U_k\boldsymbol{\beta}_{k+1}^{(j)}$ such that $AU_{k+1}\boldsymbol{e}_j \perp \widetilde{R}_k$
    $\boldsymbol{v} = AU_{k+1}\boldsymbol{e}_j$
  end for
  $k = k + 1$, update $\widetilde{R}_k$
end repeat

In Bi-CG(s), the residual and the auxiliary matrix have the following relation:

Proposition 1 ([6]) *Assume that no breakdown occurs till the k-th iteration of Bi-CG(s) and that $\boldsymbol{\alpha}_i(s) \neq 0$ for every i with $0 \leq i < k$, where $\boldsymbol{\alpha}_i(s)$ means the s-th element of the s-dimensional vector $\boldsymbol{\alpha}_i$. Then the following are true.*
a) $\boldsymbol{r}_k \in \mathcal{K}_{ks+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{ks}(A, \boldsymbol{r}_0)$.
b) $\boldsymbol{r}_k, AU_k\boldsymbol{e}_j \perp \mathcal{K}_k(A^*, \widetilde{R}_0)$, *where* $j = 1, \ldots, s$.

In the generic case, the assumptions in Proposition 1 are satisfied. Moreover, in the generic case, the overall computational cost of Bi-CG(s) for solving a system of equations can be estimated roughly as follows. The subspace $\mathcal{K}_{[N/s]}(A^*, \widetilde{\boldsymbol{r}}_0)$ is an N-dimensional space, which implies that the GBi-CG(s) terminates after $[N/s]$ iterations. In an iteration, $2s$ MATVECs (matrix-vector multiplications) are needed, and therefore, $2s \times N/s = 2N$ MATVECs are needed for computing the exact solution.

3.2 GBi-CG(s)

Our variant, to be called GBi-CG(s), is based on the observation that in Bi-CG(s) the auxiliary column vectors of U_k used in computing $U_{k+1}\boldsymbol{e}_j$ may be replaced by

the corresponding column vectors of U_{k+1} if they are already computed. By using the newest vectors available, this algorithm is expected to have improved numerical stability. The algorithm reads as follows.

Algorithm 3: GBi-CG(s) algorithm

choose $\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
set $N \times s$ matrix $U_0 = [\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{s-1}\boldsymbol{r}_0]$
$k = 0$
repeat until $\|\boldsymbol{r}_k\| < \varepsilon$ (tolerance)
  $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - AU_k\boldsymbol{\alpha}_k$ such that $\boldsymbol{r}_{k+1} \perp \widetilde{R}_k$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + U_k\boldsymbol{\alpha}_k$
  $U_{k+1}\boldsymbol{e}_1 = \boldsymbol{r}_{k+1} - U_k\boldsymbol{\beta}_{k+1}^{(1)}$ such that $AU_{k+1}\boldsymbol{e}_1 \perp \widetilde{R}_k$
  $\boldsymbol{v} = AU_{k+1}\boldsymbol{e}_1$
  for $j = 2, \ldots, s$
    set $U_k^{(j)} = [\boldsymbol{r}_{k+1}, AU_{k+1}\boldsymbol{e}_1, \ldots, AU_{k+1}\boldsymbol{e}_{j-2}, U_k\boldsymbol{e}_j, \ldots, U_k\boldsymbol{e}_s]$
    $U_{k+1}\boldsymbol{e}_j = \boldsymbol{v} - U_k^{(j)}\boldsymbol{\beta}_{k+1}^{(j)}$ such that $AU_{k+1}\boldsymbol{e}_j \perp \widetilde{R}_k$
    $\boldsymbol{v} = AU_{k+1}\boldsymbol{e}_j$
  end for
  $k = k + 1$, update $\widetilde{R}_k$
end repeat

To state the properties of GBi-CG(s) we define

$$\sigma_k = \widetilde{R}_k^* A U_k, \quad \sigma_k^{(j)} = \widetilde{R}_k^* A U_k^{(j)} \ (j = 2, \ldots, s), \quad \sigma_k' = \widetilde{R}_k^* U_k',$$

where U_k' is an $N \times s$ matrix defined as $U_k' = [r_k, AU_k\boldsymbol{e}_1, \ldots, AU_k\boldsymbol{e}_{s-1}]$.

Proposition 2 *Suppose that the matrices $\sigma_i, \sigma_i^{(j)} (j = 2, \ldots, s), \sigma_i'$ are nonsingular for all $i < k$. Then the following are true.*
a) *The algorithm does not break down in the k-th step.*
b) $\boldsymbol{r}_k \in \mathcal{K}_{ks+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{ks}(A, \boldsymbol{r}_0)$.
c) $AU_k\boldsymbol{e}_j \in \mathcal{K}_{ks+j+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{ks+j}(A, \boldsymbol{r}_0)$, *where* $j = 1, \ldots, s$.
d) $\boldsymbol{r}_k, AU_k\boldsymbol{e}_j \perp \mathcal{K}_k(A^*, \widetilde{R}_0)$, *where* $j = 1, \ldots, s$.

Proof **a)** The coefficients $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_{i+1}^{(j)}$ are computed as follows:

$$\begin{aligned}
\boldsymbol{\alpha}_i &= (\widetilde{R}_i^* A U_i)^{-1} \widetilde{R}_i^* \boldsymbol{r}_i = (\sigma_i)^{-1} \widetilde{R}_i^* \boldsymbol{r}_i, \\
\boldsymbol{\beta}_{i+1}^{(1)} &= (\widetilde{R}_i^* A U_i)^{-1} \widetilde{R}_i^* A \boldsymbol{r}_{i+1} = (\sigma_i)^{-1} \widetilde{R}_i^* A \boldsymbol{r}_{i+1}, \\
\boldsymbol{\beta}_{i+1}^{(j)} &= (\widetilde{R}_i^* A U_i^{(j)})^{-1} \widetilde{R}_i^* \boldsymbol{v} = (\sigma_i^{(j)})^{-1} \widetilde{R}_i^* \boldsymbol{v} \quad (j = 2, \ldots, s).
\end{aligned}$$

By the assumption of the nonsingularity of σ_i, $\sigma_i^{(j)}$, GBi-CG(s) does not break down in the k-th step.

b), c) We prove $\boldsymbol{r}_i \in \mathcal{K}_{is+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{is}(A, \boldsymbol{r}_0)$ and $AU_i\boldsymbol{e}_j \in \mathcal{K}_{is+j+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{is+j}(A, \boldsymbol{r}_0)$ $(j = 1, \ldots, s)$ by induction on $i = 1, 2, \ldots, k$.

(i) When $i = 1$, the residual is updated as

$$\boldsymbol{r}_1 = \boldsymbol{r}_0 - AU_0\boldsymbol{\alpha}_0 = \boldsymbol{r}_0 - A[\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{s-1}\boldsymbol{r}_0]\boldsymbol{\alpha}_0. \tag{11}$$

Therefore, to prove the relation $\boldsymbol{r}_1 \in \mathcal{K}_{s+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_s(A, \boldsymbol{r}_0)$, it is sufficient to prove $\boldsymbol{\alpha}_0(s) \neq 0$. By left-multiplying (11) with $\widetilde{R}_0^*$ we obtain $\widetilde{R}_0^*\boldsymbol{r}_0 = \widetilde{R}_0^*AU_0\boldsymbol{\alpha}_0$. Then we have $\dim\{\widetilde{R}_0^*\boldsymbol{r}_0, \widetilde{R}_0^*A\boldsymbol{r}_0, \ldots, \widetilde{R}_0^*A^{s-1}\boldsymbol{r}_0\} = s$ by the assumed nonsingularity of $\sigma_0' = \widetilde{R}_0^*U_0 = \widetilde{R}_0^*[\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{s-1}\boldsymbol{r}_0]$. Therefore $\widetilde{R}_0^*\boldsymbol{r}_0 \notin \mathrm{span}\{\widetilde{R}_0^*A\boldsymbol{r}_0, \ldots, \widetilde{R}_0^*A^{s-1}\boldsymbol{r}_0\} = \mathrm{span}\{\widetilde{R}_0^*AU_0\boldsymbol{e}_1, \ldots, \widetilde{R}_0^*AU_0\boldsymbol{e}_{s-1}\}$, which implies $\boldsymbol{\alpha}_0(s) \neq 0$. Hence follows $\boldsymbol{r}_1 \in \mathcal{K}_{s+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_s(A, \boldsymbol{r}_0)$.

The update $AU_1\boldsymbol{e}_1$ is made as

$$AU_1\boldsymbol{e}_1 = A\boldsymbol{r}_1 - AU_0\boldsymbol{\beta}_1^{(1)},$$

where $A\boldsymbol{r}_1 \in \mathcal{K}_{s+2}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{s+1}(A, \boldsymbol{r}_0)$ and $AU_0\boldsymbol{e}_j \in \mathcal{K}_{s+1}(A, \boldsymbol{r}_0)$ $(j = 1, 2, \ldots, s)$. Thus $AU_1\boldsymbol{e}_1 \in \mathcal{K}_{s+2}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{s+1}(A, \boldsymbol{r}_0)$ follows. By the same argument as $AU_1\boldsymbol{e}_1$, we can show $AU_1\boldsymbol{e}_j \in \mathcal{K}_{s+1+j}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{s+j}(A, \boldsymbol{r}_0)$ $(j = 2, \ldots, s)$.

(ii) When $i = m$ $(< k)$, we assume $\boldsymbol{r}_m \in \mathcal{K}_{ms+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{ms}(A, \boldsymbol{r}_0)$ and $AU_m\boldsymbol{e}_j \in \mathcal{K}_{ms+1+j}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{ms+j}(A, \boldsymbol{r}_0)$. Since

$$\boldsymbol{r}_{m+1} = \boldsymbol{r}_m - AU_m\boldsymbol{\alpha}_m, \tag{12}$$

to prove $\boldsymbol{r}_{m+1} \in \mathcal{K}_{(m+1)s+1}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{(m+1)s}(A, \boldsymbol{r}_0)$, it is sufficient to show that $\boldsymbol{\alpha}_m(s) \neq 0$, which can be proved by the same argument as above. In the same manner we can show that $AU_{m+1}\boldsymbol{e}_j \in \mathcal{K}_{(m+1)s+1+j}(A, \boldsymbol{r}_0) \setminus \mathcal{K}_{(m+1)s+j}(A, \boldsymbol{r}_0)$ $(j = 1, \ldots, s)$.

d) We prove $\boldsymbol{r}_i, AU_i\boldsymbol{e}_j \perp \mathcal{K}_i(A^*, \widetilde{R}_0)$ $(j = 1, \ldots, s)$ by induction on $i = 1, 2, \ldots, k$.

(i) When $i = 1$, we can see $\boldsymbol{r}_1, AU_1\boldsymbol{e}_j \perp \widetilde{R}_0$ (i.e., $\perp \mathcal{K}_1(A^*, \widetilde{R}_0)$) $(j = 1, \ldots, s)$ by the assumptions that $\sigma_0, \sigma_0^{(j)} (j = 2, \ldots, s)$ are nonsingular.

(ii) When $i = m (< k)$, we assume that $\boldsymbol{r}_m, AU_m\boldsymbol{e}_j \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$ $(j = 1, \ldots, s)$. Then the update of the residual, $\boldsymbol{r}_{m+1} = \boldsymbol{r}_m - AU_m\boldsymbol{\alpha}_m$, shows the orthogonality $\boldsymbol{r}_{m+1} \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$. We also have $\boldsymbol{r}_{m+1} \perp \widetilde{R}_m$ by the assumption of nonsingularity of σ_m. From these two we obtain $\boldsymbol{r}_{m+1} \perp \mathcal{K}_{m+1}(A^*, \widetilde{R}_0)$, where we make use of the relation $\mathcal{K}_m(A^*, \widetilde{R}_0)^\perp \cap \widetilde{R}_m^\perp = \mathcal{K}_{m+1}(A^*, \widetilde{R}_0)^\perp$, which follows from the fact that $\widetilde{R}_m = \Phi_m(A^*)\widetilde{R}_0$ with a polynomial $\Phi_m(t)$ of degree m.

Next, we consider $AU_{m+1}\boldsymbol{e}_j$ for $j = 1, \ldots, s$ in turn. For $j = 1$, $AU_{m+1}\boldsymbol{e}_1$ is updated as

$$AU_{m+1}\boldsymbol{e}_1 = A\boldsymbol{v} - AU_m\boldsymbol{\beta}_{m+1}^{(1)}, \tag{13}$$

where $\boldsymbol{v} = \boldsymbol{r}_{m+1}$. Since $\boldsymbol{v} (= \boldsymbol{r}_{m+1}) \perp \mathcal{K}_{m+1}(A^*, \widetilde{R}_0)$, we have $A\boldsymbol{v} \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$. We also have $AU_m\boldsymbol{\beta}_{m+1}^{(1)} \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$ from the assumption of $AU_m\boldsymbol{e}_j \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$ $(j = 1, \ldots, s)$. Then (13) shows $AU_{m+1}\boldsymbol{e}_1 \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$. On the other hand,

we have $AU_{m+1}\boldsymbol{e}_1 \perp \widetilde{R}_m$ by the nonsingularity of σ_m. Combination of these two implies $AU_{m+1}\boldsymbol{e}_1 \perp \mathcal{K}_{m+1}(A^*, \widetilde{R}_0)$.

By the same argument, $AU_{m+1}\boldsymbol{e}_j \perp \mathcal{K}_{m+1}(A^*, \widetilde{R}_0)$ for $j = 2, \ldots, s$ can be proved successively. Here we demonstrate the case of $j = 2$. The vector $AU_{m+1}\boldsymbol{e}_2$ is updated as

$$AU_{m+1}\boldsymbol{e}_2 = A\boldsymbol{v} - AU_m^{(2)}\boldsymbol{\beta}_{m+1}^{(2)},$$

where $\boldsymbol{v} = AU_{m+1}\boldsymbol{e}_1$. The vector $AU_m^{(2)}\boldsymbol{e}_1 (= A\boldsymbol{r}_{m+1})$ satisfies $AU_m^{(2)}\boldsymbol{e}_1 \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$, the vectors $AU_m^{(2)}\boldsymbol{e}_j (= AU_m\boldsymbol{e}_j)$ $(j = 2, \ldots, s)$ also satisfy $AU_m^{(2)}\boldsymbol{e}_j \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$ $(j = 2, \ldots, s)$ and the vector $A\boldsymbol{v} (= A^2U_{m+1}\boldsymbol{e}_1)$ satisfies $A\boldsymbol{v} \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$ since $\boldsymbol{v} (= AU_{m+1}\boldsymbol{e}_1) \perp \mathcal{K}_{m+1}(A^*, \widetilde{R}_0)$. Therefore the orthogonality $AU_{m+1}\boldsymbol{e}_2 \perp \mathcal{K}_m(A^*, \widetilde{R}_0)$ follows. Moreover by the assumption of nonsingularity of $\sigma_m^{(2)}$, we see $AU_{m+1}\boldsymbol{e}_2 \perp \widetilde{R}_m$. Thus $AU_{m+1}\boldsymbol{e}_2 \perp \mathcal{K}_{m+1}(A^*, \widetilde{R}_0)$ is proved.

This completes the proof of Proposition 2.

In the generic case, the assumptions in Proposition 2 are satisfied. GBi-CG(s) requires $2N$ matrix-vector multiplications for computing the exact solution, the same as Bi-CG(s).

4 GBi-CGSTAB(s, L)

GBi-CGSTAB(s, L), the proposed algorithm, is derived from GBi-CG(s) through the introduction of the stabilization polynomial. The overall structure of GBi-CGSTAB(s, L) is similar to that of Bi-CGSTAB(L).

We start by giving the overview of the GBi-CGSTAB(s, L) algorithm, whereas the details will be explained subsequently.

4.1 Overview

Suppose that we are given a sequence of polynomials $p_i(t)$ of degree L for $i = 1, 2, \ldots$ satisfying $p_i(0) = 1$. For $k = mL$, the k-th stabilization polynomial $Q_k(t)$ is defined as $Q_k(t) = p_m(t) \cdots p_2(t)p_1(t)$.

In this section we denote the vector $\boldsymbol{r}_k$ and the matrices U_k and $U_k^{(j)}$ $(j = 2, \ldots, s)$ in GBi-CG(s) as $\boldsymbol{r}_k^{\mathrm{GB}}$, U_k^{GB}, $U_k^{(j),\mathrm{GB}}$, respectively. We then set the residual $\boldsymbol{r}_k$ and the auxiliary matrix U_{k-1} as

$$\boldsymbol{r}_k = Q_k(A)\boldsymbol{r}_k^{\mathrm{GB}}, \quad U_{k-1} = Q_k(A)U_{k-1}^{\mathrm{GB}}.$$

Furthermore, we define approximate solutions $\boldsymbol{x}_k$ and $\hat{\boldsymbol{x}}_k^{(i)}$ as those vectors which respectively satisfy

$$\boldsymbol{b} - A\boldsymbol{x}_k = \boldsymbol{r}_k = Q_k(A)\boldsymbol{r}_k^{\mathrm{GB}}, \quad \boldsymbol{b} - A\hat{\boldsymbol{x}}_k^{(i)} = \boldsymbol{r}_{k+i} = Q_k(A)\boldsymbol{r}_{k+i}^{\mathrm{GB}}.$$

In an iteration of GBi-CGSTAB(s, L), the vectors $\boldsymbol{r}_k$ and $\boldsymbol{x}_k$ and the matrix U_{k-1} are updated to $\boldsymbol{r}_{k+L}$, $\boldsymbol{x}_{k+L}$ and U_{k+L-1}, respectively. The iteration consists of

the GBi-CG(s) part and the MR part (Figure 3), where "MR" stands for "Minimal Residual," as before.

GBi-CG(s) part: In this part, $Q_k \boldsymbol{r}_k^{\mathrm{GB}}$, $Q_k U_{k-1}^{\mathrm{GB}}$ and $\boldsymbol{x}_k$ are given. Then the i-th step, to be specified later, is performed successively for $i = 0, 1, \ldots, L-1$. Finally, the vectors $Q_k \boldsymbol{r}_{k+L}^{\mathrm{GB}}, \ldots, A^L Q_k \boldsymbol{r}_{k+L}^{\mathrm{GB}}$ and $\hat{\boldsymbol{x}}_k^{(L)}$ and the matrices $Q_k U_{k+L-1}^{\mathrm{GB}}, \ldots, A^L Q_k U_{k+L-1}^{\mathrm{GB}}$ are given. An execution of the GBi-CG(s) part can be done with $(s+1)L$ matrix-vector multiplications.

MR part: In this part, by using the output of the GBi-CG(s) part, we choose the parameters $\gamma_1^{(m+1)}, \gamma_2^{(m+1)}, \ldots, \gamma_L^{(m+1)}$ in the polynomial $p_{m+1}(t) = 1 - \sum_{i=1}^{L} \gamma_i^{(m+1)}$ such that the norm of the new residual $\boldsymbol{r}_{k+L}$ is minimum.

When the polynomial $p_{m+1}(t)$ is determined, we make the following updates on the basis of the relation $Q_{k+L}(t) = p_{m+1}(t) Q_k(t)$:

$$Q_{k+L} \boldsymbol{r}_{k+L}^{\mathrm{GB}} = Q_k \boldsymbol{r}_{k+L}^{\mathrm{GB}} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k \boldsymbol{r}_{k+L}^{\mathrm{GB}}, \tag{14}$$

$$Q_{k+L} U_{k+L-1}^{\mathrm{GB}} = Q_k U_{k+L-1}^{\mathrm{GB}} - \sum_{i=1}^{L} \gamma_i^{(m+1)} A^i Q_k U_{k+L-1}^{\mathrm{GB}}, \tag{15}$$

$$\boldsymbol{x}_{k+L} = \hat{\boldsymbol{x}}_k^{(L)} + \sum_{i=0}^{L-1} \gamma_i^{(m+1)} A^i Q_k \boldsymbol{r}_{k+L}^{\mathrm{GB}}. \tag{16}$$

4.2 Detail of the GBi-CG(s) part

We describe the GBi-CG(s) part in GBi-CGSTAB(s, L). Note the parallelism with the Bi-CG part of Bi-CGSTAB(L) described in Section 2.2.

The GBi-CG(s) part consists of L steps, from the 0-th to the $(L-1)$-st step. The i-th step in the GBi-CG(s) part is shown in Figure 3 (bottom-right), where $i = 0, 1, \ldots, L-1$. Given $A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$, $A^j Q_k U_{k+i-1}^{\mathrm{GB}}$ ($j = 0, \ldots, i$) and $\hat{\boldsymbol{x}}_k^{(i)}$, the i-th step computes $A^j Q_k \boldsymbol{r}_{k+i+1}^{\mathrm{GB}}$, $A^j Q_k U_{k+i}^{\mathrm{B}}$ ($j = 0, \ldots, i+1$) and $\hat{\boldsymbol{x}}_k^{(i+1)}$.

We refer to Figure 4, which illustrates the GBi-CG(s) part in the case of $(s, L) = (2, 2)$. The computation proceeds from row to row, replacing vectors from the previous row by vectors on the next row. Vector updates derived from GBi-CG(s) relations (3) are indicated by arrows in Figure 4.

The i-th step for $i = 0$ consists of the first row to the fourth in Figure 4. In the transition from the first row to the second, after the computation of the coefficient $\boldsymbol{\beta}_k^{(1)}$ in GBi-CG(s), we update the vector such that $Q_k U_k^{\mathrm{GB}} \boldsymbol{e}_1 = Q_k \boldsymbol{r}_k^{\mathrm{GB}} - Q_k U_{k-1}^{\mathrm{GB}} \boldsymbol{\beta}_k$. Then the vector $A Q_k U_k^{\mathrm{GB}} \boldsymbol{e}_1$ is obtained by multiplication by the matrix A. We note that the matrix $Q_k U_{k-1}^{(2),\mathrm{GB}}$ ($= [A U_k^{\mathrm{GB}} \boldsymbol{e}_1, U_{k-1}^{\mathrm{GB}} \boldsymbol{e}_2]$) is then available. From the second row to the third, after the computation of $\boldsymbol{\beta}_k^{(2)}$ in GBi-CG(s), we update the vector such that $Q_k U_k^{\mathrm{GB}} \boldsymbol{e}_2 = Q_k \boldsymbol{r}_k^{\mathrm{GB}} - Q_k U_{k-1}^{(2),\mathrm{GB}} \boldsymbol{\beta}_k^{(2)}$. Then the vector $A Q_k U_k^{\mathrm{GB}} \boldsymbol{e}_2$ is obtained by multiplication by the matrix A. From the third to the fourth, after the computation of $\boldsymbol{\alpha}_k$ in GBi-CG(s), we update the vector such that $Q_k \boldsymbol{r}_{k+1}^{\mathrm{GB}} =$

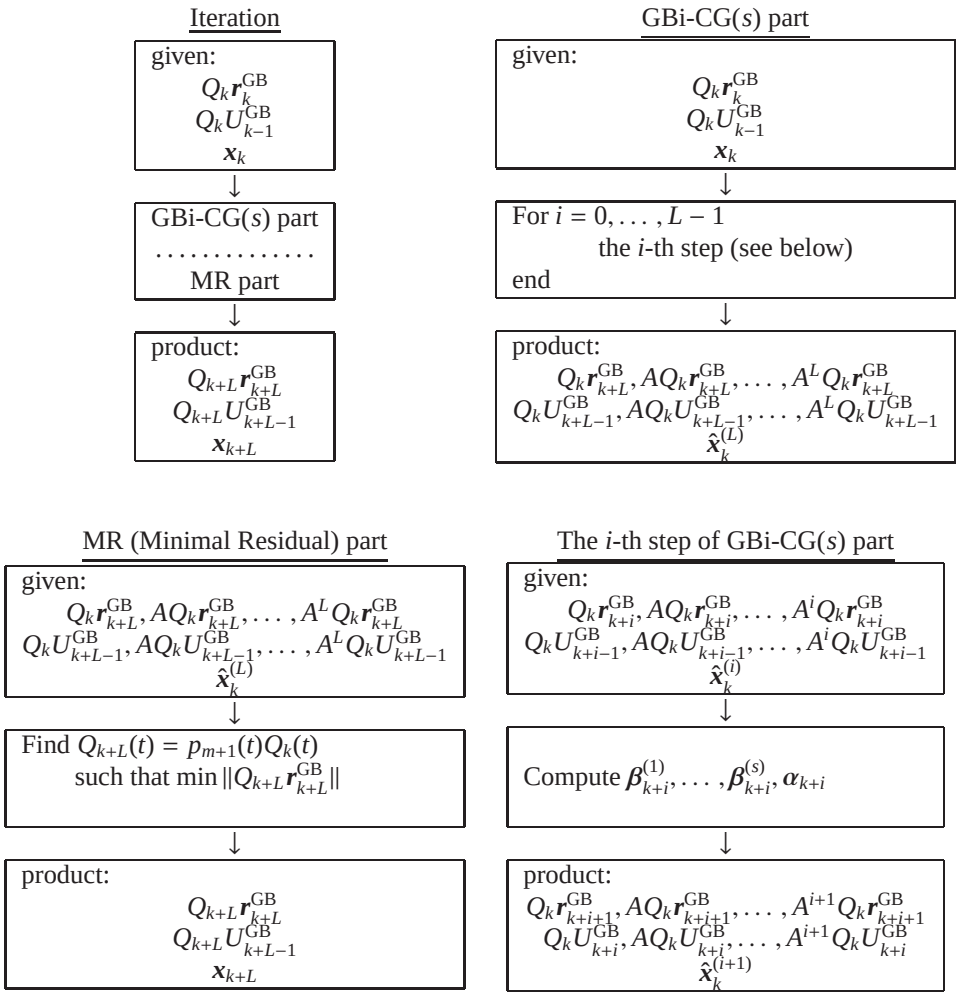

Figure 3: An iteration of GBi-CGSTAB(s, L)

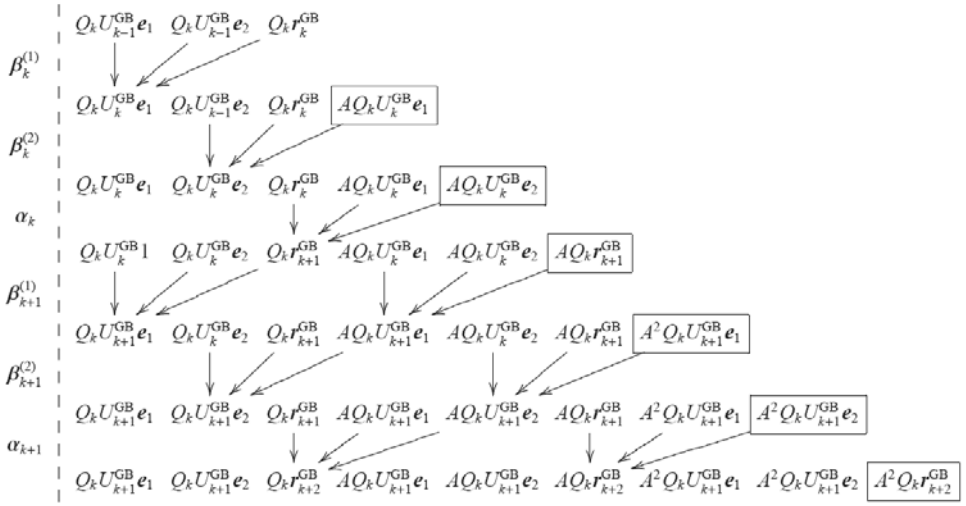

Figure 4: Flowchart of the GBi-CG(s) part of GBi-CGSTAB(2, 2)

$Q_k\boldsymbol{r}_k^{\mathrm{GB}} - AQ_kU_k^{\mathrm{GB}}\boldsymbol{\alpha}_k$. Then the vector $AQ_k\boldsymbol{r}_{k+1}^{\mathrm{GB}}$ is obtained by multiplication by the matrix A.

Next, we show the i-th step for $i = 1$. From the fourth to the fifth, after the computation of the coefficient $\boldsymbol{\beta}_{k+1}^{(1)}$, we update the vectors such that $Q_kU_{k+1}^{\mathrm{GB}}\boldsymbol{e}_1 = Q_k\boldsymbol{r}_{k+1}^{\mathrm{GB}} - Q_kU_{k+1}^{\mathrm{GB}}\boldsymbol{\beta}_{k+1}^{(1)}$ and $AQ_kU_{k+1}^{\mathrm{GB}} = AQ_k\boldsymbol{r}_{k+1}^{\mathrm{GB}} - AQ_kU_k^{\mathrm{GB}}\boldsymbol{\beta}_{k+1}^{(2)}$. Then the vector $A^2Q_kU_{k+1}^{\mathrm{GB}}\boldsymbol{e}_1$ is obtained by multiplication by the matrix A. We note that the matrix $Q_kU_k^{(2),\mathrm{GB}}(= [AU_{k+1}^{\mathrm{GB}}\boldsymbol{e}_1, U_k^{\mathrm{GB}}\boldsymbol{e}_2])$ and $AQ_kU_k^{(2),\mathrm{GB}}$ are now available. From the fifth to the sixth, after the computation of the coefficient $\boldsymbol{\beta}_{k+1}^{(2)}$, we update the vectors such that $Q_kU_{k+1}^{\mathrm{GB}}\boldsymbol{e}_2 = Q_k\boldsymbol{r}_{k+1}^{\mathrm{GB}} - AQ_kU_{k+1}^{(2),\mathrm{GB}}\boldsymbol{\beta}_{k+1}^{(2)}$ and $AQ_kU_{k+1}^{\mathrm{GB}}\boldsymbol{e}_2 = AQ_k\boldsymbol{r}_{k+1}^{\mathrm{GB}} - A^2Q_kU_k^{(2),\mathrm{GB}}\boldsymbol{\beta}_{k+1}^{(2)}$. Then he vector $A^2Q_kU_{k+1}^{\mathrm{GB}}\boldsymbol{e}_2$ is obtained by multiplication by the matrix A. From the sixth to the seventh, after the computation of the coefficient $\boldsymbol{\alpha}_{k+1}$, we update the vectors such that $Q_k\boldsymbol{r}_{k+2}^{\mathrm{GB}} = Q_k\boldsymbol{r}_{k+1}^{\mathrm{GB}} - AQ_kU_{k+1}^{\mathrm{GB}}\boldsymbol{\alpha}_{k+1}$ and $AQ_k\boldsymbol{r}_{k+2}^{\mathrm{GB}} = AQ_k\boldsymbol{r}_{k+1}^{\mathrm{GB}} - A^2Q_kU_{k+1}^{\mathrm{GB}}\boldsymbol{\alpha}_{k+1}$. Moreover, $\hat{\boldsymbol{x}}_k^{(1)}$ is replaced to $\hat{\boldsymbol{x}}_k^{(2)}$ as $\hat{\boldsymbol{x}}_k^{(2)} = \hat{\boldsymbol{x}}_k^{(1)} + Q_kU_{k+1}^{\mathrm{GB}}\boldsymbol{\alpha}_{k+1}$ (this update is not indicated in Figure 4). Then the vector $A^2Q_k\boldsymbol{r}_{k+2}^{\mathrm{GB}}$ is obtained by multiplication by the matrix A. Finally, the vectors $Q_k\boldsymbol{r}_{k+2}^{\mathrm{GB}}, AQ_k\boldsymbol{r}_{k+2}^{\mathrm{GB}}, A^2Q_k\boldsymbol{r}_{k+2}^{\mathrm{GB}}, \hat{\boldsymbol{x}}_k^{(2)}$ and the matrices $Q_kU_{k+1}^{\mathrm{GB}}, AQ_kU_{k+1}^{\mathrm{GB}}$ and $A^2Q_kU_{k+1}^{\mathrm{GB}}$ are given. The GBi-CG(s) part ends at this point if $L = 2$.

Now, we show the i-th step for a general i. It is assumed for the moment that the coefficient vectors for GBi-CG(s) in Algorithm 3 can be computed, which will be discussed later.

The update of the auxiliary matrix U can be done in s substeps as follows. In the first substep, after the computation of the coefficient $\boldsymbol{\beta}_{k+i}^{(1)}$ in GBi-CG(s), by the relation in Algorithm 3, we update $A^jQ_kU_{k+i-1}^{\mathrm{GB}}\boldsymbol{e}_1 \rightarrow A^jQ_kU_{k+i}^{\mathrm{GB}}\boldsymbol{e}_1$ ($j = 0, 1, \ldots, i$) such that $A^jQ_kU_{k+i}^{\mathrm{GB}}\boldsymbol{e}_1 = A^jQ_k\boldsymbol{r}_{k+i}^{\mathrm{GB}} - A^jQ_kU_{k+i-1}^{\mathrm{GB}}\boldsymbol{\beta}_{k+i}^{(1)}$. Then the vector $A^{i+1}Q_kU_{k+i}^{\mathrm{GB}}\boldsymbol{e}_1$ is obtained by multiplication by the matrix A. We note that the ma-

trices $A^j U_{k-1}^{(2),\mathrm{GB}}$ $(j = 0, 1, \ldots, i)$, to be used in the 2nd substep, are now available. In the second substep, after the computation of the coefficient $\boldsymbol{\beta}_{k+i}^{(2)}$ in GBi-CG(s), by the relation in Algorithm 3, we update $A^j Q_k U_{k+i-1}^{\mathrm{GB}} \boldsymbol{e}_2 \to A^j Q_k U_{k+i}^{\mathrm{GB}} \boldsymbol{e}_2$ such that $A^j Q_k U_{k+i}^{\mathrm{GB}} \boldsymbol{e}_2 = A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}} - A^j Q_k U_{k+i-1}^{(2),\mathrm{GB}} \boldsymbol{\beta}_{k+i}^{(2)}$ $(j = 0, 1, \ldots, i)$. Then the vector $A^{i+1} Q_k U_{k+i}^{\mathrm{GB}} \boldsymbol{e}_2$ is obtained by multiplication by the matrix A. We note that the matrices $A^j U_{k-1}^{(3),\mathrm{GB}}$ $(j = 0, 1, \ldots, i)$, to be used in the 3rd substep, are now available. We continue in the same way. In the t-th substep, where $t = 3, \ldots, s$, we update the vectors $A^j Q_k U_{k-1}^{\mathrm{GB}} \boldsymbol{e}_t \to A^j Q_k U_k^{\mathrm{GB}} \boldsymbol{e}_t$ $(j = 0, \ldots, i)$ and compute $A^{i+1} Q_k U_k^{\mathrm{GB}}$. Then the update of the matrices: $A^j Q_k U_{k-1}^{\mathrm{GB}} \to A^j Q_k U_k^{\mathrm{GB}}$ $(j = 0, 1, \ldots, i)$ and the computation of the matrix $A^{i+1} Q_k U_k^{\mathrm{GB}}$ are done.

The update of the residual $\boldsymbol{r}$ can be done as follows. Firstly, after the computation of the coefficient $\boldsymbol{\alpha}_{k+i}$ in GBi-CG(s), by the relation (Algorithm 3), we update $A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}} \to A^j Q_k \boldsymbol{r}_{k+i+1}^{\mathrm{GB}}$ such that $A^j Q_k \boldsymbol{r}_{k+i+1}^{\mathrm{GB}} = A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}} - A^{j+1} Q_k U_{k+i}^{\mathrm{GB}} \boldsymbol{\alpha}_{k+i}$ $(j = 0, 1, \ldots, i)$. Then the vector $A^{i+1} Q_k \boldsymbol{r}_{k+i+1}^{\mathrm{GB}}$ is obtained by multiplication by the matrix A.

The i-th step ends by returning the vectors $A^j Q_k \boldsymbol{r}_{k+i+1}^{\mathrm{GB}}$ $(j = 0, 1, \ldots, i+1)$ and $\hat{\boldsymbol{x}}_k^{(i+1)}$, and the matrices $A^j Q_k U_{k+i}^{\mathrm{GB}}$ $(j = 0, 1, \ldots, i+1)$

It remains to explain how to compute the coefficients $\boldsymbol{\alpha}_{k+i}$ and $\boldsymbol{\beta}_{k+i}^{(1)}, \ldots, \boldsymbol{\beta}_{k+i}^{(s)}$.

Computation of $\boldsymbol{\alpha}_{k+i}$

Before the computation of the coefficient $\boldsymbol{\alpha}_{k+i}$, we are given $A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$ $(j = 0, \ldots, i)$ and $A^j Q_k U_{k+i}^{\mathrm{GB}}$ $(j = 0, \ldots, i+1)$. Consider the matrix $\widetilde{R}_{k+i}$ in GBi-CG(s) (Algorithm 3), though it is not available at hand. For the choice of $\widetilde{R}_{k+i} = (A^*)^i Q_k (A^*) \widetilde{R}_0$, the vector $\boldsymbol{\alpha}_{k+i}$ is calculated as

$$
\begin{aligned}
\boldsymbol{\alpha}_{k+i} &= (\widetilde{R}_{k+i}^* A U_{k+i}^{\mathrm{GB}})^{-1} (\widetilde{R}_{k+i}^* \boldsymbol{r}_{k+i}^{\mathrm{GB}}) \\
&= \{((A^*)^i Q_k (A^*) \widetilde{R}_0)^* A U_{k+i}^{\mathrm{GB}}\}^{-1} \{((A^*)^i Q_k (A^*) \widetilde{R}_0)^* \boldsymbol{r}_{k+i}^{\mathrm{GB}}\} \\
&= \{\widetilde{R}_0^* A^{i+1} Q_k(A) U_{k+i}^{\mathrm{GB}}\}^{-1} \{\widetilde{R}_0^* A^i Q_k(A) \boldsymbol{r}_{k+i}^{\mathrm{GB}}\}.
\end{aligned}
$$

This expression affords a computable formula for $\boldsymbol{\alpha}_{k+i}$, since the vector $A^i Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$, and the matrices $A^{i+1} Q_k U_{k+i}^{\mathrm{GB}}$ and $\widetilde{R}_0$ are available.

For the efficient computation of $\boldsymbol{\alpha}_{k+i}$ as well as $\boldsymbol{\beta}_{k+i}^{(1)}, \ldots, \boldsymbol{\beta}_{k+i}^{(s)}$ below, it is convenient to employ auxiliary $s \times s$ matrix $M_{i+1}^{(k)} = \widetilde{R}_0^* A^{i+1} Q_k U_{k+i}^{\mathrm{GB}}$ and an s-dimensional vector $\boldsymbol{m}_i^{(k)} = \widetilde{R}_0^* A^i Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$. We then have $\boldsymbol{\alpha}_{k+i} = (M_{i+1}^{(k)})^{-1} \boldsymbol{m}_i^{(k)}$, which are used in our implementation.

Computation of $\boldsymbol{\beta}_{k+i}^{(1)}$

For $i > 0$: Before the computation of the coefficient $\boldsymbol{\beta}_{k+i}^{(1)}$, we are given $A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$ $(j = 0, \ldots, i)$ and $A^j Q_k U_{k+i-1}^{\mathrm{GB}}$ $(j = 0, \ldots, i)$. Consider the matrix $\widetilde{R}_{k+i-1}$ in GBi-CG(s)

(Algorithm 3). For the choice of $\widetilde{R}_{k+i-1} = (A^*)^{i-1}Q_k(A^*)\widetilde{R}_0$, the vector $\boldsymbol{\beta}_{k+i}^{(1)}$ is calculated as

$$\begin{aligned}\boldsymbol{\beta}_{k+i}^{(1)} &= (\widetilde{R}_{k+i-1}^* A U_{k+i-1}^{\mathrm{GB}})^{-1}(\widetilde{R}_{k+i-1}^* A \boldsymbol{r}_{k+i}^{\mathrm{GB}}) \\ &= \{((A^*)^{i-1}Q_k(A^*)\widetilde{R}_0)^* A U_{k+i}^{\mathrm{GB}}\}^{-1}\{((A^*)^{i-1}Q_k(A^*)\widetilde{R}_0)^* A \boldsymbol{r}_{k+i}^{\mathrm{GB}}\} \\ &= \{\widetilde{R}_0^* A^i Q_k(A) U_{k+i-1}^{\mathrm{GB}}\}^{-1}\{\widetilde{R}_0^* A^i Q_k(A)\boldsymbol{r}_{k+i}^{\mathrm{GB}}\} \\ &= (M_i^{(k)})^{-1}\boldsymbol{m}_i^{(k)}.\end{aligned}$$

This formula is used in our implementation.

For $i = 0$: Before the computation of the coefficient $\boldsymbol{\beta}^{(1)}$, we have the vectors $A^L Q_{k-L}\boldsymbol{r}_k^{\mathrm{GB}}$ and $Q_k\boldsymbol{r}_k^{\mathrm{GB}}$ and the matrix $A^L Q_{k-L}U_{k-1}^{\mathrm{GB}}$, which are given in the previous iteration of Bi-CGSTAB(L). Consider the matrix $\widetilde{R}_{k-1}$ in GBi-CG(s) (Algorithm 3). For the choice of $\widetilde{R}_{k-1} = (A^*)^{L-1}Q_{k-L}(A^*)\widetilde{R}_0$, the coefficient vector $\boldsymbol{\beta}_k^{(1)}$ is calculated as

$$\begin{aligned}\boldsymbol{\beta}_k^{(1)} &= (\widetilde{R}_{k-1}^* A U_{k-1}^{\mathrm{GB}})^{-1}(\widetilde{R}_{k-1}^* A \boldsymbol{r}_k^{\mathrm{GB}}) \\ &= \{((A^*)^{L-1}Q_{k-L}(A^*)\widetilde{R}_0)^* A U_k^{\mathrm{GB}}\}^{-1}\{((A^*)^{L-1}Q_{k-L}(A^*)\widetilde{R}_0)^* A \boldsymbol{r}_k^{\mathrm{GB}}\} \\ &= \{\widetilde{R}_0^* A^L Q_{k-L}(A) U_{k-1}^{\mathrm{GB}}\}^{-1}\{\widetilde{R}_0^* A^L Q_{k-L}(A)\boldsymbol{r}_k^{\mathrm{GB}}\}.\end{aligned}$$

This expression affords a computable formula for $\boldsymbol{\beta}_k^{(1)}$. For practical implementations, we rewrite the right-hand side above to obtain

$$\begin{aligned}\boldsymbol{\beta}_k^{(1)} &= \{-\gamma_L^{(m)}\widetilde{R}_0^* A^L Q_{k-L}(A) U_{k-1}^{\mathrm{GB}}\}^{-1}\{-\gamma_L^{(m)}\widetilde{R}_0^* A^L Q_{k-L}(A)\boldsymbol{r}_k^{\mathrm{GB}}\} \\ &= \{-\gamma_L^{(m)} M_L^{(k-L)}\}^{-1}\{\widetilde{R}_0^* Q_k \boldsymbol{r}_k^{\mathrm{GB}}\} \\ &= \{-\gamma_L^{(m)} M_L^{(k-L)}\}^{-1}\boldsymbol{m}_0^{(k)}.\end{aligned}$$

Here we have used the relation

$$-\gamma_L^{(m)}\widetilde{R}_0^* A^L Q_{k-L}\boldsymbol{r}_k^{\mathrm{GB}} = \widetilde{R}_0^* Q_k \boldsymbol{r}_k^{\mathrm{GB}}, \tag{17}$$

which follows from Proposition 2, d) and (14).

Computation of $\boldsymbol{\beta}_{k+i}^{(t)}$ ($t = 2, \ldots, s$)

For $i > 0$: Before the computation of the coefficient $\boldsymbol{\beta}_{k+i}^{(t)}$, we have $A^j Q_k \boldsymbol{r}_{k+i}^{\mathrm{GB}}$ ($j = 0, \ldots, i$), $A^j Q_k U_{k+i}^{\mathrm{GB}}\boldsymbol{e}_\nu$ ($j = 0, \ldots, i+1; \nu = 1, \ldots, t-1$) and $A^j Q_k U_{k+i-1}^{\mathrm{GB}}\boldsymbol{e}_\nu$ ($j = 0, \ldots, i; \nu = t, \ldots, s$). Thus $A^j Q_k U_{k+i-1}^{(t),\mathrm{GB}}$ ($j = 0, \ldots, i$) are also available. Consider the matrix $\widetilde{R}_{k+i-1}$ in GBi-CG(s) (Algorithm 3). For the choice of $\widetilde{R}_{k+i-1} = (A^*)^{i-1}Q_k(A^*)\widetilde{R}_0$, the coefficient vector $\boldsymbol{\beta}_{k+i}^{(t)}$ is calculated as

$$\begin{aligned}\boldsymbol{\beta}_{k+i}^{(t)} &= (\widetilde{R}_{k+i-1}^* A U_{k+i-1}^{(t),\mathrm{GB}})^{-1}(\widetilde{R}_{k+i-1}^* A^2 U_{k+i}^{\mathrm{GB}}\boldsymbol{e}_{t-1}) \\ &= \{((A^*)^{i-1}Q_k(A^*)\widetilde{R}_0)^* A U_{k+i}^{(t),\mathrm{GB}}\}^{-1}\{((A^*)^{i-1}Q_k(A^*)\widetilde{R}_0)^* A^2 U_{k+i}^{\mathrm{GB}}\boldsymbol{e}_{t-1}\} \\ &= \{\widetilde{R}_0^* A^i Q_k(A) U_{k+i-1}^{(t),\mathrm{GB}}\}^{-1}\{\widetilde{R}_0^* A^{i+1} Q_k(A) U_{k+i}^{\mathrm{GB}}\boldsymbol{e}_{t-1}\} \\ &= [\boldsymbol{m}_i^{(k)}, M_{i+1}^{(k)}\boldsymbol{e}_1, \ldots, M_{i+1}^{(k)}\boldsymbol{e}_{t-2}, M_i^{(k)}\boldsymbol{e}_t, \ldots, M_i^{(k)}\boldsymbol{e}_s]^{-1} M_{i+1}^{(k)}\boldsymbol{e}_{t-1}.\end{aligned}$$

For the last equality note that

$$U_{k+i-1}^{(t),\mathrm{GB}} = [\boldsymbol{r}_{k+i}^{\mathrm{GB}}, AU_{k+i}^{\mathrm{GB}}\boldsymbol{e}_1, \ldots, AU_{k+i}^{\mathrm{GB}}\boldsymbol{e}_{t-2}, U_{k+i-1}^{\mathrm{GB}}\boldsymbol{e}_t, \ldots, U_{k+i-1}^{\mathrm{GB}}\boldsymbol{e}_s].$$

The above expression of $\boldsymbol{\beta}_{k+i}^{(t)}$ is used in our implementation.

For $i = 0$: Before the computation of the coefficient $\boldsymbol{\beta}_k^{(t)}$, we have the vectors $Q_k\boldsymbol{r}_k^{\mathrm{GB}}$, $AQ_kU_k^{\mathrm{GB}}\boldsymbol{e}_\nu$ ($\nu = 1, \ldots, t-1$) and the previous vectors $A^LQ_{k-L}U_{k-1}^{\mathrm{GB}}\boldsymbol{e}_\nu$ ($\nu = t, \ldots, s$). Consider the matrix $\widetilde{R}_{k-1}$ in GBi-CG(s) (Algorithm 3). For the choice of $\widetilde{R}_{k-1} = (A^*)^{L-1}Q_{k-L}(A^*)\widetilde{R}_0$, the coefficient vector $\boldsymbol{\beta}_k^{(t)}$ is calculated as

$$\begin{aligned}
\boldsymbol{\beta}_k^{(t)} &= (\widetilde{R}_{k-1}^*AU_{k-1}^{(t),\mathrm{GB}})^{-1}(\widetilde{R}_{k-1}^*A^2U_k^{\mathrm{GB}}\boldsymbol{e}_{t-1}) \\
&= \{((A^*)^{L-1}Q_{k-L}(A^*)\widetilde{R}_0)^*AU_k^{(t),\mathrm{GB}}\}^{-1}\{((A^*)^{L-1}Q_{k-L}(A^*)\widetilde{R}_0)^*A^2U_k^{\mathrm{GB}}\boldsymbol{e}_{t-1}\} \\
&= \{\widetilde{R}_0^*A^LQ_{k-L}(A)U_{k-1}^{(t),\mathrm{GB}}\}^{-1}\{\widetilde{R}_0^*A^{L+1}Q_{k-L}(A)U_k^{\mathrm{GB}}\boldsymbol{e}_{t-1}\} \\
&= \{-\gamma_L^{(m)}\widetilde{R}_0^*A^LQ_{k-L}(A)U_{k-1}^{(t),\mathrm{GB}}\}^{-1}\{-\gamma_L^{(m)}\widetilde{R}_0^*A^{L+1}Q_{k-L}(A)U_k^{\mathrm{GB}}\boldsymbol{e}_{t-1}\} \qquad (18) \\
&= [\boldsymbol{m}_0^{(k)}, M_1^{(k)}\boldsymbol{e}_1, \ldots, M_1^{(k)}\boldsymbol{e}_{t-2}, -\gamma_L^{(m)}M_L^{(k-L)}\boldsymbol{e}_t, \ldots, -\gamma_L^{(m)}M_L^{(k-L)}\boldsymbol{e}_s]^{-1}M_1^{(k)}\boldsymbol{e}_{t-1},
\end{aligned}$$

where the equality between (18) and the last line can be shown as follows. First note

$$U_{k-1}^{(t),\mathrm{GB}} = [\boldsymbol{r}_k^{\mathrm{GB}}, AU_k^{\mathrm{GB}}\boldsymbol{e}_1, \ldots, AU_k^{\mathrm{GB}}\boldsymbol{e}_{t-2}, U_{k-1}^{\mathrm{GB}}\boldsymbol{e}_t, \ldots, U_{k-1}^{\mathrm{GB}}\boldsymbol{e}_s].$$

Also note

$$-\gamma_L^{(m)}\widetilde{R}_0^*A^{L+1}Q_{k-L}(A)U_k^{\mathrm{GB}}\boldsymbol{e}_\nu = \widetilde{R}_0^*AQ_k(A)U_k^{\mathrm{GB}}\boldsymbol{e}_\nu \quad (\nu = 1, \ldots, t-1), \qquad (19)$$

which follows from Proposition 2, d) and (15). For the first factor of (18) we have

$$\begin{aligned}
&-\gamma_L^{(m)}\widetilde{R}_0^*A^LQ_{k-L}(A)U_{k-1}^{(t),\mathrm{GB}} \\
&= -\gamma_L^{(m)}\widetilde{R}_0^*[A^LQ_{k-L}\boldsymbol{r}_k^{\mathrm{GB}}, A^{L+1}Q_{k-L}U_k^{\mathrm{GB}}\boldsymbol{e}_1, \ldots, A^{L+1}Q_{k-L}U_k^{\mathrm{GB}}\boldsymbol{e}_{t-2}, \\
&\qquad A^LQ_{k-L}U_{k-1}^{\mathrm{GB}}\boldsymbol{e}_t, \ldots, A^LQ_{k-L}U_{k-1}^{\mathrm{GB}}\boldsymbol{e}_s] \\
&= \widetilde{R}_0^*[Q_k\boldsymbol{r}_k^{\mathrm{GB}}, AQ_kU_k^{\mathrm{GB}}\boldsymbol{e}_1, \ldots, AQ_kU_k^{\mathrm{GB}}\boldsymbol{e}_{t-2}, \\
&\qquad -\gamma_L^{(m)}A^LQ_{k-L}U_{k-1}^{\mathrm{GB}}\boldsymbol{e}_t, \ldots, -\gamma_L^{(m)}A^LQ_{k-L}U_{k-1}^{\mathrm{GB}}\boldsymbol{e}_s] \\
&= [\boldsymbol{m}_0^{(k)}, M_1^{(k)}\boldsymbol{e}_1, \ldots, M_1^{(k)}\boldsymbol{e}_{t-2}, -\gamma_L^{(m)}M_L^{(k-L)}\boldsymbol{e}_t, \ldots, -\gamma_L^{(m)}M_L^{(k-L)}\boldsymbol{e}_s],
\end{aligned}$$

where (17) and (19) are used. The second factor of (18) can be rewritten by (19) as

$$-\gamma_L^{(m)}\widetilde{R}_0^*A^{L+1}Q_{k-L}(A)U_k^{\mathrm{GB}}\boldsymbol{e}_{t-1} = \widetilde{R}_0^*AQ_k(A)U_k^{\mathrm{GB}}\boldsymbol{e}_{t-1} = M_1^{(k)}\boldsymbol{e}_{t-1}.$$

The following is the GBi-CGSTAB(s, L) algorithm in full detail:

Algorithm 4: GBi-CGSTAB(s, L) algorithm

1. choose $\boldsymbol{x}_0$ and $N \times s$ matrix $\widetilde{R}_0$
2. set $U_0 = [\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{s-1}\boldsymbol{r}_0]$, $U_1 = AU_0$
3. $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
4. $M = \widetilde{R}_0^* U_1$, $\boldsymbol{m} = \widetilde{R}_0^* \boldsymbol{r}_0$
5. solve $M\boldsymbol{\beta} = \boldsymbol{m}$ for $\boldsymbol{\beta}$
6. $\boldsymbol{r}_0 = \boldsymbol{r}_0 - U_1\boldsymbol{\beta}$, $\boldsymbol{x}_0 = \boldsymbol{x}_0 + U_0\boldsymbol{\beta}$
7. $\boldsymbol{r}_1 = A\boldsymbol{r}_0$, iter = 0, $\omega = -1$
8. repeat until $\|\boldsymbol{r}_0\| < \varepsilon$ (tolerance)
9. $M = -\omega M$
10. for $i = 0, 1, \ldots, L-1$
11. if (iter = 0)$\wedge(i = 0)$ $i = 1$
12. $\boldsymbol{m} = \widetilde{R}_0^* \boldsymbol{r}_i$
13. for $j = 1, \ldots, s$
14. if ($j = 1$)
15. solve $M\boldsymbol{\beta} = \boldsymbol{m}$ for $\boldsymbol{\beta}$
16. $U_k\boldsymbol{e}_j = \boldsymbol{r}_k - \sum_{q=1}^{s} U_k\boldsymbol{e}_q\beta(q)$ $(k = 0, \ldots, i)$
17. else
18. solve $[\boldsymbol{m}, M\boldsymbol{e}_1, \ldots, M\boldsymbol{e}_{j-2}, M\boldsymbol{e}_j, \ldots, M\boldsymbol{e}_s]\boldsymbol{\beta} = M\boldsymbol{e}_{j-1}$ for $\boldsymbol{\beta}$
19. $U_k\boldsymbol{e}_j = U_{k+1}\boldsymbol{e}_{j-1} - \boldsymbol{r}_k\beta(1) - \sum_{q=1}^{j-2} U_{k+1}\boldsymbol{e}_q\beta(q+1) - \sum_{q=j}^{s} U_k\boldsymbol{e}_q\beta(q)$ $(k = 0, \ldots, i)$
20. end if
21. $U_{i+1}\boldsymbol{e}_j = AU_i\boldsymbol{e}_j$
22. $M\boldsymbol{e}_j = \widetilde{R}_0^* U_{i+1}\boldsymbol{e}_j$
23. end for
24. solve $M\boldsymbol{\beta} = \boldsymbol{m}$ for $\boldsymbol{\beta}$
25. $\boldsymbol{r}_k = \boldsymbol{r}_k - U_{k+1}\boldsymbol{\beta}$ $(k = 0, \ldots, i)$
26. $\boldsymbol{x}_0 = \boldsymbol{x}_0 + U_0\boldsymbol{\beta}$
27. $\boldsymbol{r}_{i+1} = A\boldsymbol{r}_i$
28. end for
29. for $j = 1, 2, \ldots, L$
30. $\tau_{ij} = \frac{1}{\sigma_i}(\boldsymbol{r}_i, \boldsymbol{r}_j)$, $\boldsymbol{r}_j = \boldsymbol{r}_j - \tau_{ij}\boldsymbol{r}_i$ $(i = 1, 2, \ldots, j-1)$
31. $\sigma_j = (\boldsymbol{r}_j, \boldsymbol{r}_j)$, $\gamma'_j = \frac{1}{\sigma_j} = (\boldsymbol{r}_j, \boldsymbol{r}_0)$
32. end for
33. $\gamma_L = \gamma'_L$, $\omega = \gamma_L$
34. $\gamma_j = \gamma'_j - \sum_{i=j+1}^{L} \tau_{ji}\gamma_i$ $(j = L-1, \ldots, 1)$
35. $\gamma''_j = \gamma_{j+1} + \sum_{i=j+1}^{L-1} \tau_{ji}\gamma_{i+1}$ $(j = 1, \ldots, L-1)$

36. $\boldsymbol{x}_0 = \boldsymbol{x}_0 + \gamma_1 \boldsymbol{r}_0,\ \boldsymbol{r}_0 = \boldsymbol{r}_0 - \gamma'_L \boldsymbol{r}_L,\ U_0 = U_0 - \gamma_L U_L$
37. $U_0 = U_0 - \gamma_j U_j\ (j = 1, \ldots, L-1)$
38. $\boldsymbol{x}_0 = \boldsymbol{x}_0 + \gamma''_j \boldsymbol{r}_j,\ \boldsymbol{r}_0 = \boldsymbol{r}_0 - \gamma'_j \boldsymbol{r}_j\ (j = 1, \ldots, L-1)$
39. iter =iter $+(s+1)L$
40. end repeat

A remark is in order about the initial vectors. In the above algorithm we set $U_0 = [\boldsymbol{r}_0, A\boldsymbol{r}_0 \ldots, A^{s-1}\boldsymbol{r}_0]$ for ease of description. In practical implementations, however, it is recommended that some orthogonalization procedure be applied to $\boldsymbol{r}_0, A\boldsymbol{r}_0 \ldots, A^{s-1}\boldsymbol{r}_0$ to avoid or mitigate numerical instability due to rounding errors. We have in fact adopted this idea in our numerical experiments reported in Section 5.

4.3 GBi-CGSTAB(s, L) with preconditioning

We describe the right-preconditioned GBi-CGSTAB(s, L). For a matrix $\hat{A}$ that is close to the given matrix A, we consider the right-preconditioned system

$$A\hat{A}^{-1}\boldsymbol{y} = \boldsymbol{b}, \quad \boldsymbol{y} = \hat{A}\boldsymbol{x}.$$

GBi-CGSTAB(s, L) applied to this system of equations in $\boldsymbol{y}$ can be translated to an iteration in $\boldsymbol{x}$ as follows.

Algorithm 5: GBi-CGSTAB(s, L) algorithm with preconditioning

1. choose $\boldsymbol{x}_0$ and $N \times s$ matrices $\widetilde{R}_0$ and U_0
2. set $U_0 = [\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{s-1}\boldsymbol{r}_0]$, $\hat{U}_0 = \hat{A}^{-1}U_0$
3. $U_1 = A\hat{U}_0$
4. $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
5. $M = \widetilde{R}_0^* U_1, \boldsymbol{m} = \widetilde{R}_0^* \boldsymbol{r}_0$
6. solve $M\boldsymbol{\beta} = \boldsymbol{m}$ for $\boldsymbol{\beta}$
7. $\boldsymbol{r}_0 = \boldsymbol{r}_0 - U_1\boldsymbol{\beta}$, $\boldsymbol{x}_0 = \boldsymbol{x}_0 + \hat{U}_0\boldsymbol{\beta}$
8. $\hat{\boldsymbol{r}}_0 = \hat{A}^{-1}\boldsymbol{r}_0$
9. $\boldsymbol{r}_1 = A\widetilde{\boldsymbol{r}_0}$, iter = 0, $\omega = -1$
10. repeat until $\|\boldsymbol{r}_0\| < \varepsilon$ (tolerance)
11. $M = -\omega M$
12. for $i = 0, 1, \ldots, L-1$
13. if (iter = 0)$\wedge(i = 0)$ $i = 1$
14. $\boldsymbol{m} = \widetilde{R}_0^* \boldsymbol{r}_i$
15. for $j = 1, \ldots, s$
16. if ($j = 1$)
17. Solve $M\boldsymbol{\beta} = \boldsymbol{m}$ for $\boldsymbol{\beta}$
18. $U_k \boldsymbol{e}_j = \boldsymbol{r}_k - \sum_{q=1}^{s} U_k \boldsymbol{e}_q \beta(q)\ (k = 0, \ldots, i)$

19. $\hat{U}_k \boldsymbol{e}_j = \hat{\boldsymbol{r}}_k - \sum_{q=1}^{s} \hat{U}_k \boldsymbol{e}_q \beta(q)$ $(k = 0, \ldots, i-1)$
20. else
21. solve $[\boldsymbol{m}, M\boldsymbol{e}_1, \ldots, M\boldsymbol{e}_{j-2}, M\boldsymbol{e}_j, \ldots, M\boldsymbol{e}_s]\boldsymbol{\beta} = M\boldsymbol{e}_{j-1}$ for $\boldsymbol{\beta}$
22. $U_k \boldsymbol{e}_j = U_{k+1}\boldsymbol{e}_{j-1} - \boldsymbol{r}_k \beta(1) - \sum_{q=1}^{j-2} U_{k+1}\boldsymbol{e}_q \beta(q+1) - \sum_{q=j}^{s} U_k \boldsymbol{e}_q \beta(q)$ $(k = 0, \ldots, i)$
23. $\hat{U}_k \boldsymbol{e}_j = \hat{U}_{k+1}\boldsymbol{e}_{j-1} - \hat{\boldsymbol{r}}_k \beta(1) - \sum_{q=1}^{j-2} \hat{U}_{k+1}\boldsymbol{e}_q \beta(q+1) - \sum_{q=j}^{s} \hat{U}_k \boldsymbol{e}_q \beta(q)$ $(k = 0, \ldots, i-1)$
24. end if
25. $\hat{U}_i \boldsymbol{e}_j = \hat{A}^{-1} U_i \boldsymbol{e}_j$
26. $U_{i+1}\boldsymbol{e}_j = A\hat{U}_i \boldsymbol{e}_j$
27. $M\boldsymbol{e}_j = \widetilde{R}_0^* U_{i+1}\boldsymbol{e}_j$
28. end for
29. solve $M\boldsymbol{\beta} = \boldsymbol{m}$ for $\boldsymbol{\beta}$
30. $\boldsymbol{r}_k = \boldsymbol{r}_k - U_{k+1}\boldsymbol{\beta}$ $(k = 0, \ldots, i)$
31. $\hat{\boldsymbol{r}}_k = \hat{\boldsymbol{r}}_k - \hat{U}_{k+1}\boldsymbol{\beta}$ $(k = 0, \ldots, i-1)$
32. $\boldsymbol{x}_0 = \boldsymbol{x}_0 + \hat{U}_0 \boldsymbol{\beta}$
33. $\hat{\boldsymbol{r}}_i = \hat{A}^{-1}\boldsymbol{r}_i$
34. $\boldsymbol{r}_{i+1} = A\hat{\boldsymbol{r}}_i$
35. end for
36. for $j = 1, 2, \ldots, L$
37. $\tau_{ij} = \frac{1}{\sigma_i}(\boldsymbol{r}_i, \boldsymbol{r}_j)$, $\boldsymbol{r}_j = \boldsymbol{r}_j - \tau_{ij}\boldsymbol{r}_i$ $(i = 1, 2, \ldots, j-1)$
38. $\sigma_j = (\boldsymbol{r}_j, \boldsymbol{r}_j)$, $\gamma'_j = \frac{1}{\sigma_j} = (\boldsymbol{r}_j, \boldsymbol{r}_0)$
39. end for
40. $\gamma_L = \gamma'_L$, $\omega = \gamma_L$
41. $\gamma_j = \gamma'_j - \sum_{i=j+1}^{L} \tau_{ji}\gamma_i$ $(j = L-1, \ldots, 1)$
42. $U_0 = U_0 - \gamma_j U_j$ $(j = 1, \ldots, L)$
43. $\boldsymbol{x}_0 = \boldsymbol{x}_0 + \gamma_{j+1}\hat{\boldsymbol{r}}_j$, $\boldsymbol{r}_0 = \boldsymbol{r}_0 - \gamma'_{j+1}\boldsymbol{r}_{j+1}$ $(j = 0, \ldots, L-1)$
44. iter =iter $+(s+1)L$
45. end repeat

4.4 The relation between IDR(s) and GBi-CGSTAB(s, 1)

In this section, we reveal a relation between the residuals of GBi-CGSTAB(s, 1) and IDR(s) in exact arithmetic. For linearly independent vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_s$, put $\widetilde{R}_0 = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_s]$ and define $\boldsymbol{p}_j$ for $j > s$ by $\boldsymbol{p}_{ms+i} = (A^*)^m \boldsymbol{p}_i$ $(1 \le i \le s, m \ge 1)$.

Proposition 3 *Suppose that the assumption of Proposition 2 for* GBi-CG(s) *holds for some k. Also suppose that* IDR(s) *with* $\mathcal{S} = \widetilde{R}_0^{\perp}$ *does not break down before the*

$(s+1)$*k-th step. If, in addition, the matrices*

$$S_{ms} = \begin{bmatrix} \boldsymbol{p}_1^* A\boldsymbol{r}_0 & \boldsymbol{p}_1^* A^2\boldsymbol{r}_0 & \dots & \boldsymbol{p}_1^* A^{ms}\boldsymbol{r}_0 \\ \boldsymbol{p}_2^* A\boldsymbol{r}_0 & \boldsymbol{p}_2^* A^2\boldsymbol{r}_0 & \dots & \boldsymbol{p}_2^* A^{ms}\boldsymbol{r}_0 \\ \vdots & \vdots & & \vdots \\ \boldsymbol{p}_{ms}^* A\boldsymbol{r}_0 & \boldsymbol{p}_{ms}^* A^2\boldsymbol{r}_0 & \dots & \boldsymbol{p}_{ms}^* A^{ms}\boldsymbol{r}_0 \end{bmatrix} \tag{20}$$

are nonsingular for all m with $0 < m \leq k$*, then the residual* $\boldsymbol{r}_m$ *in* GBi-CGSTAB$(s, 1)$ *is identical with the residual* $\boldsymbol{r}_{(s+1)m}$ *in* IDR(s) *for all m with* $0 < m \leq k$*.*

Proof First, we show how the two residuals are represented by the matrix A and the initial residual $\boldsymbol{r}_0$. After the m-th step, the residual of GBi-CGSTAB(s, L) is represented as

$$\boldsymbol{r}_m = Q_m(A)\boldsymbol{r}_m^{\mathrm{GB}},$$

where $Q_m(t) = p_1(t)p_2(t)\cdots p_m(t)$ is the m-th stabilization polynomial. The coefficient of $p_m(t) = 1 - \omega_m t$ is determined from

$$\min_{\omega_m} \|(I - \omega_m A)Q_{m-1}\boldsymbol{r}_m^{\mathrm{GB}}\|.$$

Secondly, by [10], the residual $\boldsymbol{r}_{(s+1)m}$ in IDR(s) is represented as

$$\boldsymbol{r}_{m(s+1)} = \Omega_m(A)\Psi_{ms}(A)\boldsymbol{r}_0,$$

where $\Omega_m(t) = (1-\omega_m' t)(1-\omega_{m-1}' t)\cdots(1-\omega_1' t)$ is the m-th stabilization polynomial in IDR(s). The coefficient ω_m' is determined from

$$\min_{\omega_m'} \|(I - \omega_m' A)\boldsymbol{v}\|,$$

where $\boldsymbol{v} = \Omega_{m-1}(A)\Psi_{ms}(A)\boldsymbol{r}_0$.

Now it is sufficient to show that $\boldsymbol{r}_m^{\mathrm{GB}} = \Psi_{ms}\boldsymbol{r}_0$. By Proposition 2 and [10], both $\boldsymbol{r}_m^{\mathrm{GB}}$ and $\Psi_{ms}\boldsymbol{r}_0$ belong to $\mathcal{K}_{ms+1}(A, \boldsymbol{r}_0)$ and hence they are both represented as $(I - c_1 A - \cdots - c_{ms}A^{ms})\boldsymbol{r}_0$. The coefficients c_i here are determined from the orthogonality of $\boldsymbol{r}_m^{\mathrm{GB}}$ and $\Psi_{ms}\boldsymbol{r}_0$ to $\mathcal{K}_m(A^*, \widetilde{R}_0) = \mathrm{span}\{\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_{ms}\}$, i.e., from

$$S_{ms}[c_1, \dots, c_{ms}]^\top = [\boldsymbol{p}_1^*\boldsymbol{r}_0, \dots, \boldsymbol{p}_{ms}^*\boldsymbol{r}_0]^\top,$$

where S_{ms} is a nonsingular matrix (by the assumption). Hence the coefficients $c_1, \dots, c_{ms}$ for $\boldsymbol{r}_m^{\mathrm{GB}}$ and $\Psi_{ms}\boldsymbol{r}_0$ are identical, which means $\boldsymbol{r}_m^{\mathrm{GB}} = \Psi_{ms}\boldsymbol{r}_0$.

4.5 Computational cost and memory requirement

Computational cost of GBi-CGSTAB(s, L) can be evaluated roughly as follows under the assumption that no breakdown occurs. In an iteration of GBi-CGSTAB(s, L), $(s+1)L$ MATVECs (matrix-vector multiplications) are needed. The algorithm terminates in $[N/sL]$ iterations. Hence, the number of MATVECs is $(s+1)L \times N/sL =$

$N + N/s$ in solving linear equations by GBi-CGSTAB(s, L). Recall that "$N + N/s$ MATVECs for convergence" is a feature shared by IDR(s).

Other computational cost and memory requirements are summarized in Table1. We follow the convention of [10] to scale numbers of operations by the number of matrix-vector multiplications.

- AXPY means the number of operations of the form "(scalar) $\times$ (vector) + (vector)," where an addition of two vectors and a scalar multiplication of a vector is weighted 0.5.
- DOT means the number of inner products.
- MEMORY shows the memory requirements in terms of the number of N dimensional vectors, including storage for the right-hand side and the solution and excluding storage for the system matrix and the preconditioner.

It should be noted that GBi-CGSTAB$(s, 1)$ has an advantage over IDR(s) in the number of AXPY. They are mathematically equivalent, but have difference in algorithms.

Table 1: Numbers of vector operations and memory requirements

Method	MVs	AXPY	DOT	MEMORY
Bi-CGSTAB(L)	1	$\frac{3}{4}(L+3)$	$\frac{1}{4}(L+7)$	$2L+5$
IDR(s)	1	$2s + \frac{3}{2} + \frac{1}{s+1}$	$s + \frac{2}{s+1}$	$3s+5$
GBi-CGSTAB(s, L)	1	$\frac{s(L+1)}{2} + 2 + \frac{L-1}{2s+2}$	$s + \frac{L+3}{2s+2}$	$sL + L + 2s + 3$
GBi-CGSTAB$(1, L)$	1	$\frac{3}{4}(L+3)$	$\frac{1}{4}(L+7)$	$2L+5$
GBi-CGSTAB$(s, 1)$	1	$s+2$	$s + \frac{2}{s+1}$	$3s+4$

5 Numerical Experiments

5.1 A 3-dimensional convection-dominated problem

The first problem, taken from [5, 10], arises from a discretization of a partial differential equation. We consider

$$u_{xx} + u_{yy} + u_{zz} + 1000u_x = F$$

on $[0, 1] \times [0, 1] \times [0, 1]$ with the Dirichlet boundary condition, where the function F is specified in such a way that $u(x, y, z) = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$ is a solution to this problem. Discretization by central differences is adopted. The number of grid points is 52 in each direction of the xyz-space, and a system of 125,000 linear equations results. The coefficient matrix is nearly skew-symmetric,

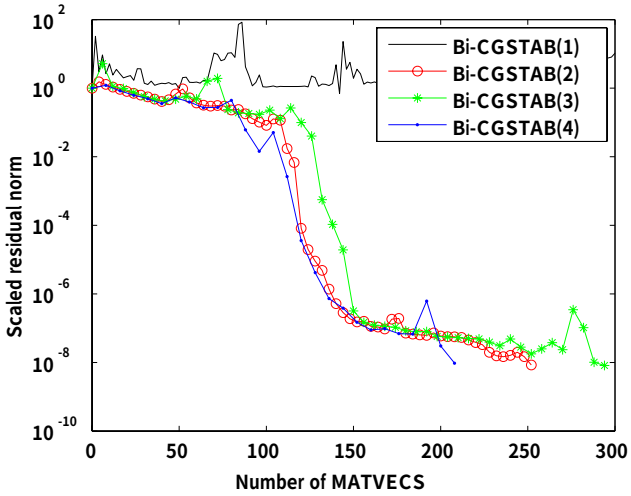

Figure 5: Convergence of Bi-CGSTAB(L) (Example 1)

and it is observed in [5, 10] that this causes slow convergence for methods with linear ($L = 1$) stabilization polynomials.

We have applied Bi-CGSTAB(L), IDR(s) and GBi-CGSTAB(s, L) with parameters $s \in [1, 4]$ and $L \in [1, 4]$. No preconditioning is used. We start with $\boldsymbol{x}_0 = [0, 0, \ldots, 0]^\top$ and stop the iterations when the residual norm, scaled by the norm of the right-hand side vector, drops below 10^{-8}. Figures 5, 6, 7 and 8 show the convergence. While it is confirmed that the algorithms with $L = 1$ has poor convergence, we see that algorithms with $L > 1$ are significantly more efficient. In particular the proposed algorithm GBi-CGSTAB(s, L), with $L > 2$, converges faster than IDR(s) and is comparable to Bi-CGSTAB(L). The total numbers of matrix-vector multiplications needed to solve the problem are tabulated in Tables 2 and 3.

All the experiments, including those in Sections 5.2 and 5.3, are performed using a home-made program with MATLAB 7.5.

5.2 Helmholz-like equation

Second, we consider the following Helmholz-like equation:

$$u_{xx} + u_{yy} + \sigma^2 u + 0.1 u_x = F$$

with the Dirichlet boundary condition, where the function F is specified in such a way that $u(x, y) = \sin(\sqrt{\sigma^2 - 1/2}\, x) \cos(y/2)$ is a solution to this problem. We set $\sigma = 4.16$. A discretization results in a system of 40,000 linear equations.

We have applied Bi-CGSTAB(L), IDR(s) and GBi-CGSTAB(s, L) with parameters $s \in [1, 4]$ and $L \in [1, 4]$. We have adopted a standard preconditioner of

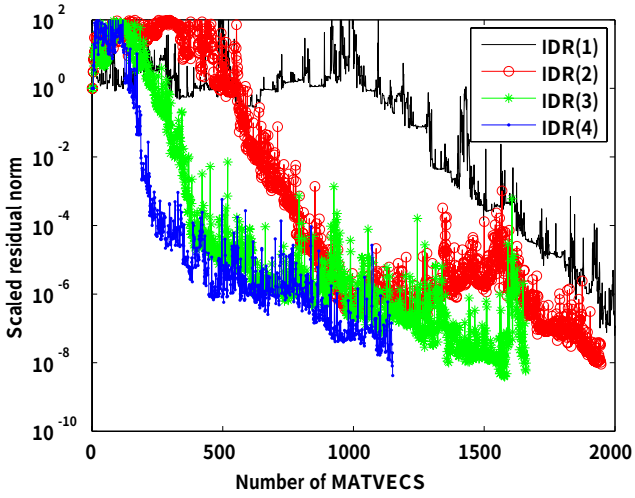

Figure 6: Convergence of IDR(s) (Example 1)

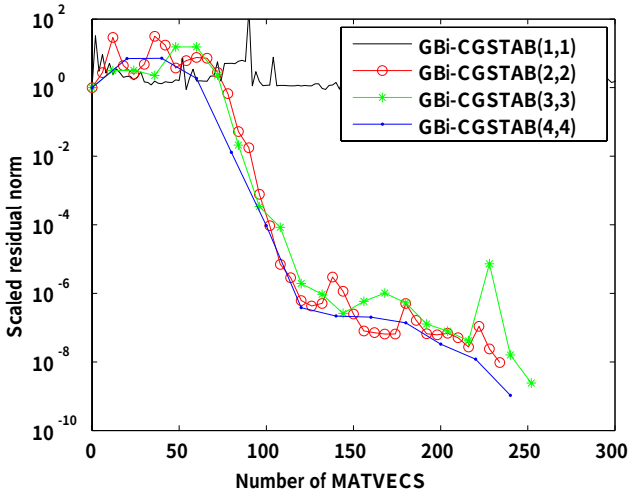

Figure 7: Convergence of GBi-CGSTAB(s, L) (Example 1)

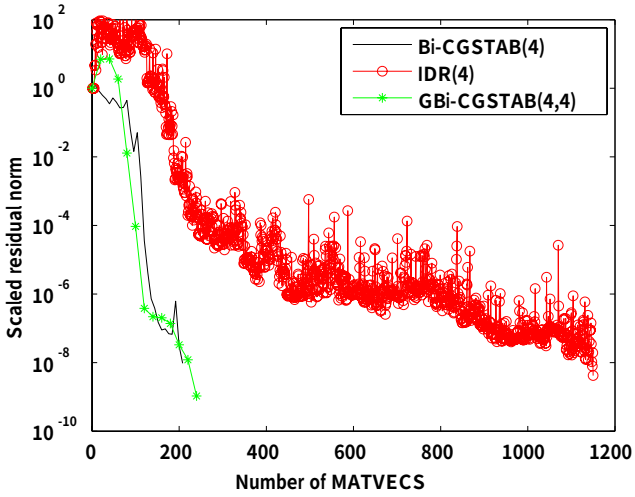

Figure 8: Comparison of the three methods with $s = 4$, $L = 4$ (Example 1)

Table 2: Number of matrix-vector multiplications in Bi-CGSTAB(L) and IDR(s) (Example 1)

METHOD	MATVECS
Bi-CGSTAB(1)	2112
Bi-CGSTAB(2)	252
Bi-CGSTAB(3)	294
Bi-CGSTAB(4)	208
IDR(1)	2044
IDR(2)	1947
IDR(3)	1660
IDR(4)	1150

Table 3: Number of matrix-vector multiplications in GBi-CGSTAB(s, L) (Example 1)

s \ L	1	2	3	4
1	2070	240	252	224
2	1983	234	270	252
3	1396	232	252	240
4	1155	240	255	240

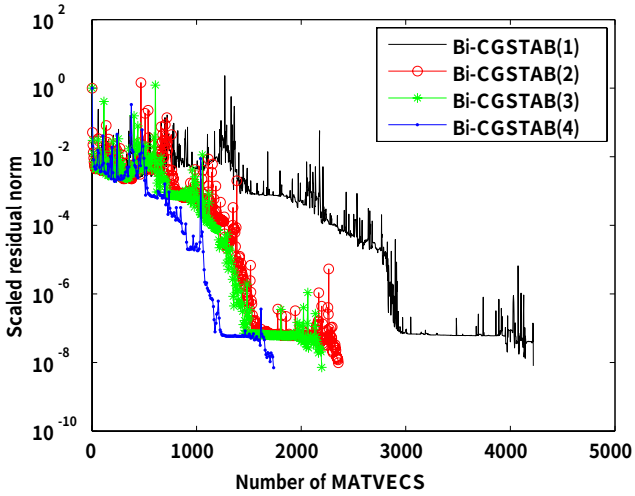

Figure 9: Convergence of Bi-CGSTAB(L) (Example 2)

ILU(0). We start with $\boldsymbol{x}_0 = [0, 0, \ldots, 0]^\top$ and stop the iterations when the residual norm, scaled by the norm of the right-hand side vector, drops below 10^{-8}. Figures 9, 10, 11 and 12 show the convergence. We see that higher-order stabilization polynomials and higher-dimensional shadow residuals are effective. GBi-CGSTAB(s, L) with $L > 1$ tends to converge faster than IDR(s).

5.3 University of Florida sparse matrix collection

Finally we consider the matrices of the University of Florida sparse matrix collection [2] (Table 6). We choose $\boldsymbol{b} = A[1, 1, \ldots, 1]^\top$ as the right vectors. We fix the parameters $(s, L) = (4, 4)$. We start with $\boldsymbol{x}_0 = [0, 0, \ldots, 0]^\top$ and stop the iterations when the residual norm, scaled by the norm of the right-hand side vector, drops below 10^{-8}.

The numbers of matrix-vector multiplications needed to solve the problems by Bi-CGSTAB(4), IDR(4) and GBi-CGSTAB(4, 4) are compared in Table 7, (a) with no preconditioner and (b) with a preconditioner of ILU(0). For most of the problems in Table 6, the convergence of GBi-CGSTAB(s, L) turned out to be roughly the same as that of IDR(s).

6 Conclusion

We have proposed GBi-CGSTAB(s, L), a new variant of Bi-CGSTAB with multiple ($s > 1$) shadow residuals and with higher order ($L > 1$) stabilization polynomials.

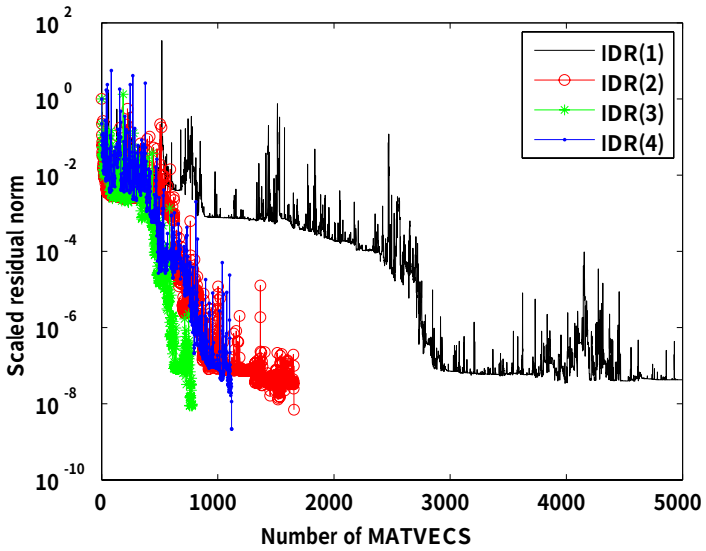

Figure 10: Convergence of IDR(s) (Example 2)

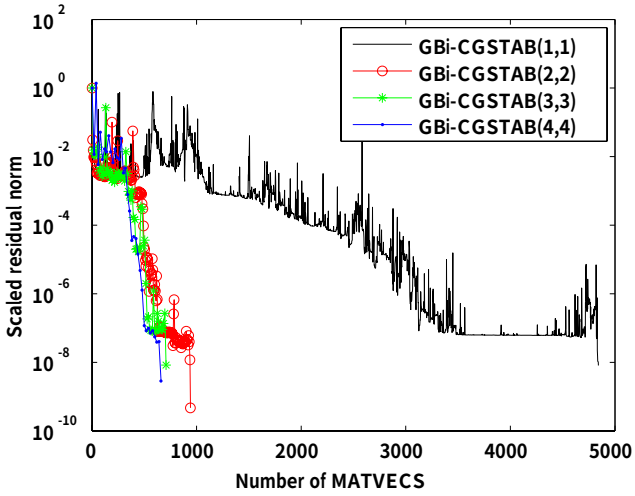

Figure 11: Convergence of GBi-CGSTAB(s, L) (Example 2)

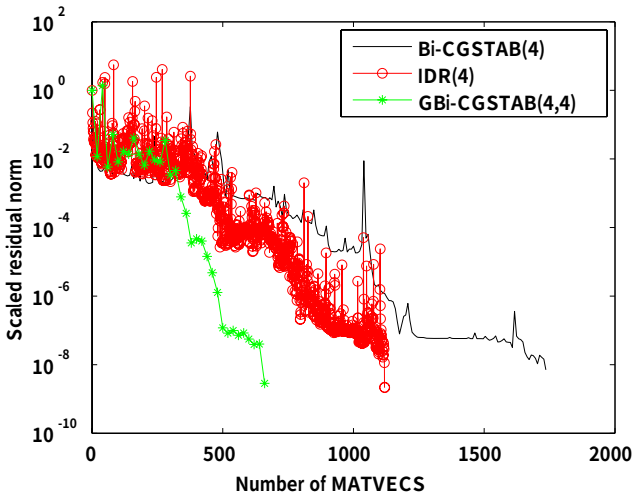

Figure 12: Comparison of the three methods with $s = 4$, $L = 4$ (Example 2)

Table 4: Number of matrix-vector multiplications in Bi-CGSTAB(L) and IDR(s) (Example 2) ("n.c." means no convergence within 10000 matrix-vector multiplications .)

METHOD	MATVECS
Bi-CGSTAB(1)	4220
Bi-CGSTAB(2)	2356
Bi-CGSTAB(3)	2196
Bi-CGSTAB(4)	1736
IDR(1)	n.c.
IDR(2)	1656
IDR(3)	780
IDR(4)	1120

Table 5: Number of matrix-vector multiplications in GBi-CGSTAB(s, L) (Example 2)

s \ L	1	2	3	4
1	4842	2440	2382	1648
2	1419	942	909	696
3	944	784	708	736
4	690	590	585	660

Table 6: Size and the number of nonzero entries of the matrices (Example 3)

	size	entries
add20	2395	17319
add32	4960	23884
epb3	84617	463625
memplus	17758	126150
poisson3Da	13514	352762
poisson3Db	85623	2374949
raefsky2	3242	293551
sme3Da	12504	874887
sme3Db	29067	2081063
wang4	26068	177196

This method, GBi-CGSTAB(s, L), has the property that, in exact arithmetic, it can compute the exact solution with at most $N + N/s$ matrix-vector multiplications. Through the numerical experiments we have shown that GBi-CGSTAB(s, L) shares good features with both IDR(s) and Bi-CGSTAB(L).

Acknowledgements

The authors wish to express their sincere gratitude to Professor Kazuo Murota of the University of Tokyo for his comments and suggestions. This work is supported by a Grant-in-Aid for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan

After the completion of this manuscript the authors learned about a recent report [7] on a closely related topic by G. L. G. Sleijpen and M. B. van Gijzen. The proposed algorithms are similar but different in updating the auxiliary vectors, denoted as U_k in the present paper. Our algorithm is to theirs what the Gauss-Seidel iteration is to the Jacobi iteration.

References

[1] J. I. Aliaga, D. L. Boley, R. W. Freund and V. Hernandez, A Lanczos-type method for multiple starting vectors, Mathematics of Computation, Vol. 69 (2000), pp. 1577–1602.

[2] A. Bayliss, C. I. Goldstein and E. Turkel, An iterative method for the Helmholtz equation, Journal of Computational Physics, Vol. 49 (1983), pp. 443–457.

Table 7: Number of matrix-vector multiplications to solve the problems (Example 3) ("n.c." means no convergence within 10000 matrix-vector multiplications.)

(a) Without preconditioner

	Bi-CGSTAB(4)	IDR(4)	GBi-CGSTAB(4, 4)
add20	576	515	440
add32	104	105	100
epb3	6376	4265	3940
memplus	1480	1185	1100
poisson3Da	200	190	220
poisson3Db	392	335	340
raefsky2	640	450	460
sme3Da	n.c.	4995	3600
sme3Db	n.c.	6615	4660
wang4	680	460	500

(b) With preconditioner ILU(0)

	Bi-CGSTAB(4)	IDR(4)	GBi-CGSTAB(4, 4)
add20	184	205	160
add32	48	55	60
epb3	120	130	140
memplus	312	395	280
poisson3Da	64	70	60
poisson3Db	160	130	140
raefsky2	64	55	60
sme3Da	1856	570	600
sme3Db	1856	800	720
wang4	80	70	80

[3] T. Davis, University of Florida sparse matrix collection, http://www.cise.ufl.edu/research/sparse/matrices/index.html.

[4] R. Fletcher, Conjugate gradient methods for indefinite systems, in Proceedings of the Dundee Biennal Conference on Numerical Analysis 1974, G. A. Watson, ed., Springer-Verlag, New York, 1975, pp. 73–89.

[5] G. L. G. Sleijpen and D. R. Fokkema, BICGSTAB(L) for equations involving unsymmetric matrices with complex spectrum, Electronic Transactions on Numerical Analysis, Vol. 1 (1993), pp. 11–32.

[6] G. L. G. Sleijpen, P. Sonneveld and M. B. van Gijzen, Bi-CGSTAB as an induced dimension reduction method, Reports of the Department of Applied Mathematical Analysis, REPORT 08-07 (2008), Delft University of Technology.

[7] G. L. G. Sleijpen and M. B. van Gijzen, Exploiting BiCGstab(ℓ) strategies to induce dimension reduction, Reports of the Department of Applied Mathematical Analysis, REPORT 09-02 (2009), Delft University of Technology.

[8] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear system, SIAM Journal on Scientific and Statistical Computing, Vol. 10 (1989), pp. 36–52.

[9] P. Sonneveld and M. B. van Gijzen, IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations, Reports of the Department of Applied Mathematical Analysis, REPORT 07-07 (2007), Delft University of Technology.

[10] P. Sonneveld and M. B. van Gijzen, IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations, SIAM Journal on Scientific Computing, Vol. 31 (2008), pp. 1035–1062.

[11] M. Tanio and M. Sugihara, GIDR(s, L): generalized IDR(s) (in Japanese), The 2008 annual conference of the Japan Society for Industrial and Applied Mathematics, pp. 411–412, Chiba, Japan, September, 2008.

[12] H. A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, Vol. 13 (1992), pp. 631–644.

[13] H. A. van der Vorst, Iterative Krylov Methods for Large Linear Systems, Cambridge University Press, 2003.

[14] S.-L. Zhang, GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM Journal on Scientific Computing, Vol. 18 (1997), pp. 537–551.