Received: 8 March 2019 | Revised: 6 December 2019 | Accepted: 20 February 2020

DOI: 10.1002/nla.2298

RESEARCH ARTICLE

WILEY

GPBi-CGstab(L): A Lanczos-type product method unifying Bi-CGstab(L) and GPBi-CG

Kensuke Aihara

Department of Computer Science, Tokyo City University, Tokyo, Japan

Correspondence
Kensuke Aihara, Department of Computer Science, Tokyo City University, 1-28-1 Tamazutsumi, Setagaya-ku, Tokyo 158-8557, Japan.
Email: aiharak@tcu.ac.jp

Funding information
Japan Society for the Promotion of Science, Grant/Award Number: JP18K18064

Summary

Lanczos-type product methods (LTPMs), in which the residuals are defined by the product of stabilizing polynomials and the Bi-CG residuals, are effective iterative solvers for large sparse nonsymmetric linear systems. Bi-CGstab(L) and GPBi-CG are popular LTPMs and can be viewed as two different generalizations of other typical methods, such as CGS, Bi-CGSTAB, and Bi-CGStab2. Bi-CGstab(L) uses stabilizing polynomials of degree L, while GPBi-CG uses polynomials given by a three-term recurrence (or equivalently, a coupled two-term recurrence) modeled after the Lanczos residual polynomials. Therefore, Bi-CGstab(L) and GPBi-CG have different aspects of generalization as a framework of LTPMs. In the present paper, we propose novel stabilizing polynomials, which combine the above two types of polynomials. The resulting method is referred to as GPBi-CGstab(L). Numerical experiments demonstrate that our presented method is more effective than conventional LTPMs.

KEYWORDS

linear system, Krylov subspace method, Lanczos-type product method, Bi-CGstab(L), GPBi-CG

1 | INTRODUCTION

The bi-conjugate gradient method (Bi-CG)[1] is one of the most basic iterative solvers for large sparse nonsymmetric linear systems with the form $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n\times n}$ is nonsingular and $\boldsymbol{b} \in \mathbb{R}^n$. In the present study, we treat Lanczos-type product methods (LTPMs),[2] which are characterized by generating the residuals defined by the form

$$\boldsymbol{r}_k := H_k(A)\boldsymbol{r}_k^{bicg},$$

where $\boldsymbol{r}_k^{bicg}$ is the kth Bi-CG residual and $H_k(\lambda)$ is a so-called stabilizing polynomial of degree k. These types of methods are also called hybrid Bi-CG methods. By selecting appropriate polynomials $H_k(\lambda)$, LTPMs can have better convergence than the underlying Bi-CG for various problems. For this reason, several LTPMs have been developed over the years.

Studies on the stabilizing polynomials stem from the CG squared method (CGS),[3] which has been developed by Sonneveld as an efficient variant of Bi-CG. The CGS residuals are defined by $\boldsymbol{r}_k^{cgs} := R_k^2(A)\boldsymbol{r}_0$, where $\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$ is an initial residual and $R_k(\lambda)$ is a Lanczos residual polynomial, that is, the residual polynomial of Bi-CG, and $\boldsymbol{r}_k^{bicg} = R_k(A)\boldsymbol{r}_0$ holds. Therefore, CGS can be viewed as the first LTPM by selecting $H_k(\lambda) := R_k(\lambda)$. Although CGS often converges at twice the speed of Bi-CG, it also exhibits larger oscillations in the residual norms, leading to a loss of accuracy of the approximate solutions. Thus, other stabilized variants have been proposed in the class of LTPMs.

The Bi-CG stabilized method (Bi-CGSTAB)[4] proposed by van der Vorst uses the stabilizing polynomials defined by

$$H_0(\lambda) := 1, \quad H_{k+1}(\lambda) := (1 - \zeta_k \lambda) H_k(\lambda), \tag{1}$$

where the coefficient $\zeta_k \in \mathbb{R}$ is determined, for example, by locally minimizing the residual at each iteration. Presently, Bi-CGSTAB is one of the most popular methods in many fields. However, real stabilizing polynomials of degree 1 do not work well if $A \in \mathbb{R}^{n \times n}$ has complex eigenvalues such that the imaginary part is relatively large compared to the real part. To overcome this difficulty, Gutknecht[5] incorporated second-degree polynomials into the iteration process of Bi-CGSTAB. The resulting method is known as Bi-CGStab2. Subsequently, Sleijpen and Fokkema[6] developed the more general method Bi-CGstab(L), which uses stabilizing polynomials of degree L; that is,

$$H_0(\lambda) := 1, \quad H_{k+L}(\lambda) := (1 - \zeta_{k,1}\lambda - \zeta_{k,2}\lambda^2 - \cdots - \zeta_{k,L}\lambda^L) H_k(\lambda), \tag{2}$$

where the integer k is a multiple of the parameter L. The coefficients $\zeta_{k,j} \in \mathbb{R}$ for $j = 1, 2, \ldots, L$ are usually selected such that the updated residual is locally minimized. The formulation (2) is a natural extension of (1); that is, Bi-CGstab(L) simplifies to Bi-CGSTAB when $L = 1$. Bi-CGStab2 is also covered by Bi-CGstab(L) in theory; more precisely, when $L = 2$, Bi-CGstab(2) generates the same residuals as obtained for even iterations of Bi-CGStab2 (if there is no breakdown). However, they exhibit significantly different convergence properties in actual computations. Bi-CGstab(L) with $L > 1$ (e.g., $L = 2$ or 4) is actually often more effective than Bi-CGSTAB and Bi-CGStab2, especially for highly nonsymmetric matrices with complex eigenvalues close to the imaginary axis.

As another extension of CGS, Fokkema et al.[7] developed generalized CGS methods (GCGS), which use a product of two nearby Bi-CG polynomials (instead of the squares of one Bi-CG polynomial). In these methods the stabilizing polynomials are given by coupled two-term recurrences, as in the Lanczos residual polynomials. Thus, GCGS has two recurrence coefficients as parameters, and the conventional CGS and Bi-CGSTAB can be viewed as special cases of GCGS. On the other hand, based on a recurrence property of the Lanczos residual polynomials, Zhang[8] introduced more general and essential polynomials:

$$\begin{aligned} H_0(\lambda) &:= 1, \quad H_1(\lambda) := (1 - \zeta_0 \lambda) H_0(\lambda), \\ H_{k+1}(\lambda) &:= (1 + \eta_k - \zeta_k \lambda) H_k(\lambda) - \eta_k H_{k-1}(\lambda), \end{aligned} \tag{3}$$

where $k = 1, 2, \ldots$, and $\zeta_k, \eta_k \in \mathbb{R}$ are independent parameters. This formulation gives a collection of the product-type methods based on Bi-CG, and some of the conventional LTPMs can be derived. By choosing the two parameters to be identical to those in the Lanczos residual polynomials with the Bi-CG coefficients, $H_k(\lambda)$ corresponds to $R_k(\lambda)$, and it leads to CGS. When $\eta_k = 0$, (3) simplifies to (1), and we can recover Bi-CGSTAB by selecting ζ_k to minimize the residuals. Bi-CGStab2 can be obtained by minimizing the residual with respect to ζ_k and η_k at even iterations, and to ζ_k at odd iterations with $\eta_k = 0$. Zhang suggested determining ζ_k and η_k to minimize the residual at each iteration. The resulting method, named GPBi-CG, is quite competitive in LTPMs for systems with a complex spectrum. Note that, in an actual derivation of the GPBi-CG algorithm, coupled two-term recurrences are utilized (instead of (3)) to construct $H_k(\lambda)$. For details, we refer to section 4 in Zhang.[8]

It is important to develop a comprehensive algorithm in LTPMs, because this may enable us to classify the methods in a simple manner, and also the resulting new method may have higher performance by combining the strengths of the existing methods. Bi-CGstab(L) and GPBi-CG are two valuable algorithms both as specific effective solvers and as frameworks of LTPMs. However, because there is no inclusion relationship between the two methods, it is important to consider a more general framework that unifies Bi-CGstab(L) and GPBi-CG. Therefore, in the present paper, we propose such a novel method, named GPBi-CGstab(L). The stabilizing polynomials of the new method consist of Lth degree polynomials and relaxation terms defined by the differences of the two previous polynomials; a specific definition is given in Section 3. There are $L + 1$ parameters that determine the stabilizing polynomials. The resulting method simplifies to Bi-CGstab(L) if one parameter in the relaxation term is set to 0, and it is mathematically equivalent to GPBi-CG in the case of $L = 1$. Therefore, the conventional LTPMs, such as CGS, Bi-CGSTAB, and Bi-CGStab2, are also included in this framework. As a specific algorithm, it would be natural and effective to select appropriate $L + 1$ parameters to locally minimize the residuals. Numerical experiments demonstrate that the proposed GPBi-CGstab(L) can obtain better convergence than the conventional LTPMs when solving nonsymmetric linear systems, especially with a complex spectrum.

The remainder of this paper is organized as follows. In the next section, we give derivations of the Bi-CGstab(L) and GPBi-CG algorithms, which are useful for designing the new method. In Section 3, we introduce comprehensive stabilizing polynomials and derive an algorithm for GPBi-CGstab(L) unifying Bi-CGstab(L) and GPBi-CG. In Section 4, we demonstrate the effectiveness of GPBi-CGstab(L) through numerical experiments on model problems. The concluding remarks and future prospects are presented in Section 5.

2 | BI-CGSTAB(L) AND GPBI-CG

In this section, we describe the Bi-CG recurrences used in LTPMs and derive Bi-CGstab(L) and GPBi-CG algorithms individually. These derivations differ slightly from the original ones but are useful to design the novel method.

2.1 | Bi-CG recurrences in LTPMs

We first recall the recurrence relations in the standard Bi-CG. Let $\boldsymbol{r}_k^{bicg}$ and $\boldsymbol{p}_k^{bicg}$ be the residual and direction vectors of Bi-CG, respectively. It is well-known that these vectors satisfy the following coupled two-term recurrences:

$$\begin{aligned}\boldsymbol{r}_{k+1}^{bicg} &= \boldsymbol{r}_k^{bicg} - \alpha_k A\boldsymbol{p}_k^{bicg},\\ \boldsymbol{p}_{k+1}^{bicg} &= \boldsymbol{r}_{k+1}^{bicg} - \beta_k \boldsymbol{p}_k^{bicg}.\end{aligned} \tag{4}$$

Here, the initial direction is defined as $\boldsymbol{p}_0^{bicg} := \boldsymbol{r}_0^{bicg}$ and the Bi-CG coefficients α_k and β_k are determined by the bi-orthogonality conditions $\boldsymbol{r}_{k+1}^{bicg}, A\boldsymbol{p}_{k+1}^{bicg} \perp \mathcal{K}_{k+1}(A^\top, \tilde{\boldsymbol{r}}_0)$, where $\mathcal{K}_k(A^\top, \tilde{\boldsymbol{r}}_0)$ is the kth Krylov subspace spanned by $A^\top$ (the transpose of A) and an initial shadow residual $\tilde{\boldsymbol{r}}_0$.

In the following, $H_k(A)$ is denoted as H_k for simplicity. By multiplying (4) from the left by H_k, we have the following:

$$\begin{aligned}\mathrm{H}_k\boldsymbol{r}_{k+1}^{bicg} &= \mathrm{H}_k\boldsymbol{r}_k^{bicg} - \alpha_k A\mathrm{H}_k\boldsymbol{p}_k^{bicg},\\ \mathrm{H}_k\boldsymbol{p}_{k+1}^{bicg} &= \mathrm{H}_k\boldsymbol{r}_{k+1}^{bicg} - \beta_k \mathrm{H}_k\boldsymbol{p}_k^{bicg}.\end{aligned} \tag{5}$$

The residual and direction vectors of LTPMs are expressed as $\boldsymbol{r}_k = \mathrm{H}_k\boldsymbol{r}_k^{bicg}$ and $\boldsymbol{p}_k = \mathrm{H}_k\boldsymbol{p}_k^{bicg}$, respectively. Therefore, by introducing the intermediate auxiliary vectors $\boldsymbol{r}_k^{(1)} := \mathrm{H}_k\boldsymbol{r}_{k+1}^{bicg}$ and $\boldsymbol{p}_k^{(1)} := \mathrm{H}_k\boldsymbol{p}_{k+1}^{bicg}$ into (5), we obtain the update formulas (called a Bi-CG step) as are often used in LTPMs:

$$\begin{aligned}\text{Compute } A\boldsymbol{p}_k &= A \star \boldsymbol{p}_k,\\ \boldsymbol{r}_k^{(1)} &= \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k,\\ \boldsymbol{p}_k^{(1)} &= \boldsymbol{r}_k^{(1)} - \beta_k \boldsymbol{p}_k,\end{aligned} \tag{6}$$

where $\star$ denotes an explicit matrix–vector multiplication. The associated approximations can be expressed by $\boldsymbol{x}_k^{(1)} = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k$.

From the biorthogonality conditions, α_k and β_k can be expressed by

$$\alpha_k = \frac{(\tilde{\boldsymbol{r}}_0, \mathrm{H}_k\boldsymbol{r}_k^{bicg})}{\sigma_k} = \frac{(\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k)}{\sigma_k}, \quad \beta_k = \frac{(\tilde{\boldsymbol{r}}_0, A\mathrm{H}_k\boldsymbol{r}_{k+1}^{bicg})}{\sigma_k} = \frac{(\tilde{\boldsymbol{r}}_0, A\boldsymbol{r}_k^{(1)})}{\sigma_k}, \tag{7}$$

where $\sigma_k := (\tilde{\boldsymbol{r}}_0, A\mathrm{H}_k\boldsymbol{p}_k^{bicg}) = (\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)$. Note that there exist different formulations of the Bi-CG coefficients. The formulation (7) is used in, for example, Abe and Sleijpen.[9,10] The vector $A\boldsymbol{r}_k^{(1)}$ is computed in the iteration process. If $A\boldsymbol{r}_k^{(1)}$ is obtained by a matrix–vector multiplication (i.e., $A\boldsymbol{r}_k^{(1)} = A \star \boldsymbol{r}_k^{(1)}$), the vector $A\boldsymbol{p}_k^{(1)}$, usually required for updating iteration vectors, can be obtained by a vector update (i.e., $A\boldsymbol{p}_k^{(1)} = A \star \boldsymbol{r}_k^{(1)} - \beta_k A\boldsymbol{p}_k$). These computations are often performed together with (6) as parts of the Bi-CG step. For the numerical stabilities of these formulations, we refer to Abe and Sleijpen.[9,10]

The above Bi-CG recurrences play an important role in the derivation of LTPM computational schemes in the present study.

2.2 | Variant of Bi-CGstab(L)

Here, we derive a variant of the Bi-CGstab(L) algorithm, which uses the stabilizing polynomials $H_k(\lambda)$ given by (2).

The residual and direction vectors of Bi-CGstab(L) are updated from $\boldsymbol{r}_k = \mathrm{H}_k \boldsymbol{r}_k^{bicg}$ and $\boldsymbol{p}_k = \mathrm{H}_k \boldsymbol{p}_k^{bicg}$ to $\boldsymbol{r}_{k+L} = \mathrm{H}_{k+L} \boldsymbol{r}_{k+L}^{bicg}$ and $\boldsymbol{p}_{k+L} = \mathrm{H}_{k+L} \boldsymbol{p}_{k+L}^{bicg}$, respectively, where k is a multiple of L. This updating process is counted as one cycle of Bi-CGstab(L), and it consists of repeating the Bi-CG step L times and performing Lth degree minimization.[6,11]

We assume that an approximation $\boldsymbol{x}_k$, the corresponding residual $\boldsymbol{r}_k = \mathrm{H}_k \boldsymbol{r}_k^{bicg}$, and the direction vector $\boldsymbol{p}_k = \mathrm{H}_k \boldsymbol{p}_k^{bicg}$ are given at the beginning of the cycle. At the jth repetition of the Bi-CG steps ($j = 1, 2, \dots, L$), we can use the following recurrence relations derived from (5).

$$\begin{aligned} A^i \mathrm{H}_k \boldsymbol{r}_{k+j}^{bicg} &= A^i \mathrm{H}_k \boldsymbol{r}_{k+j-1}^{bicg} - \alpha_{k+j-1} A^{i+1} \mathrm{H}_k \boldsymbol{p}_{k+j-1}^{bicg}, \quad i = 0, 1, \dots, j-1, \\ A^i \mathrm{H}_k \boldsymbol{p}_{k+j}^{bicg} &= A^i \mathrm{H}_k \boldsymbol{r}_{k+j}^{bicg} - \beta_{k+j-1} A^i \mathrm{H}_k \boldsymbol{p}_{k+j-1}^{bicg}, \quad i = 0, 1, \dots, j. \end{aligned} \tag{8}$$

Like in (6) we introduce the intermediate auxiliary vectors $\boldsymbol{r}_k^{(j)} := \mathrm{H}_k \boldsymbol{r}_{k+j}^{bicg}$ and $\boldsymbol{p}_k^{(j)} := \mathrm{H}_k \boldsymbol{p}_{k+j}^{bicg}$ into (8), and we also rewrite the coefficients as $\alpha_k^{(j)} := \alpha_{k+j}$ and $\beta_k^{(j)} := \beta_{k+j}$. Then, starting with $\boldsymbol{r}_k^{(0)} := \boldsymbol{r}_k = \mathrm{H}_k \boldsymbol{r}_k^{bicg}$ and $\boldsymbol{p}_k^{(0)} := \boldsymbol{p}_k = \mathrm{H}_k \boldsymbol{p}_k^{bicg}$, the updating process of the jth repetition can be expressed as follows.

$$\text{Compute } A^j \boldsymbol{p}_k^{(j-1)} = A \star A^{j-1} \boldsymbol{p}_k^{(j-1)}, \tag{9}$$

$$A^i \boldsymbol{r}_k^{(j)} = A^i \boldsymbol{r}_k^{(j-1)} - \alpha_k^{(j-1)} A^{i+1} \boldsymbol{p}_k^{(j-1)}, \quad i = 0, 1, \dots, j-1, \tag{10}$$

$$\text{Compute } A^j \boldsymbol{r}_k^{(j)} = A \star A^{j-1} \boldsymbol{r}_k^{(j)}, \tag{11}$$

$$A^i \boldsymbol{p}_k^{(j)} = A^i \boldsymbol{r}_k^{(j)} - \beta_k^{(j-1)} A^i \boldsymbol{p}_k^{(j-1)}, \quad i = 0, 1, \dots, j. \tag{12}$$

Corresponding to (10) with $i = 0$, the associated approximations can be expressed by

$$\boldsymbol{x}_k^{(j)} = \boldsymbol{x}_k^{(j-1)} + \alpha_k^{(j-1)} \boldsymbol{p}_k^{(j-1)} \tag{13}$$

with $\boldsymbol{x}_k^{(0)} := \boldsymbol{x}_k$. Here, using the iteration vectors, the Bi-CG coefficients $\alpha_k^{(j)}$ and $\beta_k^{(j)}$ can be computed as follows.

$$\alpha_k^{(j)} = \frac{(\tilde{\boldsymbol{r}}_0, A^j \mathrm{H}_k \boldsymbol{r}_{k+j}^{bicg})}{\sigma_k^{(j)}} = \frac{(\tilde{\boldsymbol{r}}_0, A^j \boldsymbol{r}_k^{(j)})}{\sigma_k^{(j)}}, \tag{14}$$

$$\beta_k^{(j)} = \frac{(\tilde{\boldsymbol{r}}_0, A^{j+1} \mathrm{H}_k \boldsymbol{r}_{k+j+1}^{bicg})}{\sigma_k^{(j)}} = \frac{(\tilde{\boldsymbol{r}}_0, A^{j+1} \boldsymbol{r}_k^{(j+1)})}{\sigma_k^{(j)}}, \tag{15}$$

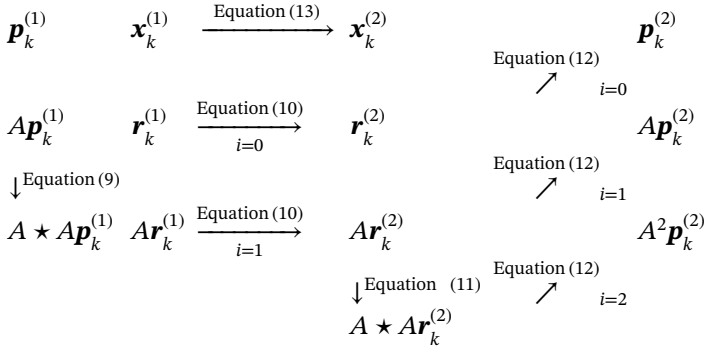
where $\sigma_k^{(j)} := (\tilde{\boldsymbol{r}}_0, A^{j+1} \mathrm{H}_k \boldsymbol{p}_{k+j}^{bicg}) = (\tilde{\boldsymbol{r}}_0, A^{j+1} \boldsymbol{p}_k^{(j)})$. Therefore, when we have $\boldsymbol{x}_k^{(j-1)}$, $A^i \boldsymbol{r}_k^{(j-1)}$, and $A^i \boldsymbol{p}_k^{(j-1)}$ for $i = 0, 1, \dots, j-1$ at the beginning of the jth repetition, the new vectors $\boldsymbol{x}_k^{(j)}$, $A^i \boldsymbol{r}_k^{(j)}$, and $A^i \boldsymbol{p}_k^{(j)}$ for $i = 0, 1, \dots, j$ are generated by (9)–(15).

Following the depictions in Sleijpen and van Gijzen,[11] the computational schemes of the first and second repetitions of the Bi-CG steps are displayed as follows.

- Computational scheme of the first repetition ($j = 1$) of the Bi-CG steps:

$$\begin{array}{ccccc} \boldsymbol{p}_k & \boldsymbol{x}_k & \xrightarrow{\text{Equation (13)}} & \boldsymbol{x}_k^{(1)} & \\ & & & & \boldsymbol{p}_k^{(1)} \\ \downarrow{\scriptstyle \text{Equation (9)}} & & & & \nearrow{\scriptstyle \text{Equation (12)}}_{i=0} \\ A \star \boldsymbol{p}_k & \boldsymbol{r}_k & \xrightarrow[i=0]{\text{Equation (10)}} & \boldsymbol{r}_k^{(1)} & A\boldsymbol{p}_k^{(1)} \\ & & & \downarrow{\scriptstyle \text{Equation (11)}} & \nearrow{\scriptstyle \text{Equation (12)}}_{i=1} \\ & & & A \star \boldsymbol{r}_k^{(1)} & \end{array} \tag{16}$$

- Computational scheme of the second repetition ($j = 2$) of the Bi-CG steps:



After the Bi-CG step is repeated L times, we have $\boldsymbol{x}_k^{(L)}$, $A^j\boldsymbol{r}_k^{(L)} = A^j\mathrm{H}_k\boldsymbol{r}_{k+L}^{bicg}$, and $A^j\boldsymbol{p}_k^{(L)} = A^j\mathrm{H}_k\boldsymbol{p}_{k+L}^{bicg}$ for $j = 0, 1, \dots, L$. From (2), the next residual $\boldsymbol{r}_{k+L}$ can be computed by

$$\begin{aligned}\boldsymbol{r}_{k+L} &= \mathrm{H}_{k+L}\boldsymbol{r}_{k+L}^{bicg}\\ &= (I - \zeta_{k,1}A - \cdots - \zeta_{k,L}A^L)\mathrm{H}_k\boldsymbol{r}_{k+L}^{bicg}\\ &= \boldsymbol{r}_k^{(L)} - \zeta_{k,1}A\boldsymbol{r}_k^{(L)} - \cdots - \zeta_{k,L}A^L\boldsymbol{r}_k^{(L)},\end{aligned}$$

where $\zeta_{k,j}$ for $j = 1, 2, \dots, L$ are usually determined by minimizing $\|\boldsymbol{r}_{k+L}\|_2$ (cf. section 4.2.1 in Sleijpen et al.[12]). The associated approximation $\boldsymbol{x}_{k+L}$ and the next direction vector $\boldsymbol{p}_{k+L} = \mathrm{H}_{k+L}\boldsymbol{p}_{k+L}^{bicg}$ can also be updated as follows.

$$\begin{aligned}\boldsymbol{x}_{k+L} &= \boldsymbol{x}_k^{(L)} + \zeta_{k,1}\boldsymbol{r}_k^{(L)} + \cdots + \zeta_{k,L}A^{L-1}\boldsymbol{r}_k^{(L)},\\ \boldsymbol{p}_{k+L} &= \boldsymbol{p}_k^{(L)} - \zeta_{k,1}A\boldsymbol{p}_k^{(L)} - \cdots - \zeta_{k,L}A^L\boldsymbol{p}_k^{(L)}.\end{aligned}$$

Algorithm 1 presents a variant of Bi-CGstab(L). As used in Sleijpen and van Gijzen,[11] the notation in the algorithm follows MATLAB conventions. The vectors $\mathbf{r}_i$ and $\mathbf{p}_i$ for $i = 0, 1, \dots, j$ are related to $\mathbf{r}$ and $\mathbf{p}$ as $\mathbf{r} = [\mathbf{r}_0; \mathbf{r}_1; \dots; \mathbf{r}_j]$ and $\mathbf{p} = [\mathbf{p}_0; \mathbf{p}_1; \dots; \mathbf{p}_j]$, respectively, where $[\mathbf{w}_0; \mathbf{w}_1; \dots; \mathbf{w}_j] := [\mathbf{w}_0^\top, \mathbf{w}_1^\top, \dots, \mathbf{w}_j^\top]^\top$. Note that the vectors $A^i\boldsymbol{r}_k^{(j)}$ and $A^i\boldsymbol{p}_k^{(j)}$ correspond to $\mathbf{r}_i$ and $\mathbf{p}_i$, respectively.

Algorithm 1. Variant of Bi-CGstab(L)

Select an initial guess $\mathbf{x}$
Compute $\mathbf{r}_0 = \mathbf{b} - A \star \mathbf{x}$, and choose $\tilde{\mathbf{r}}_0$
Set $\mathbf{r} = [\mathbf{r}_0]$ and $\mathbf{p} = [\mathbf{r}_0]$
while $\|\mathbf{r}_0\|_2 > tol$ **do**
  $\rho = \tilde{\mathbf{r}}_0^\top\mathbf{r}_0$
  for $j = 1, 2, \dots, L$ **do**
    $\mathbf{p} = [\mathbf{p}; A \star \mathbf{p}_{j-1}]$
    $\sigma = \tilde{\mathbf{r}}_0^\top\mathbf{p}_j, \quad \alpha = \rho/\sigma$
    $\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}_0, \quad \mathbf{r} = \mathbf{r} - \alpha[\mathbf{p}_1; \mathbf{p}_2; \dots; \mathbf{p}_j], \quad \mathbf{r} = [\mathbf{r}; A \star \mathbf{r}_{j-1}]$
    $\rho = \tilde{\mathbf{r}}_0^\top\mathbf{r}_j, \quad \beta = \rho/\sigma, \quad \mathbf{p} = \mathbf{r} - \beta\mathbf{p}$
  end for
  Compute $\boldsymbol{\zeta} = [\zeta_1; \zeta_2; \dots; \zeta_L]$ to minimize $\|\mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L]\boldsymbol{\zeta}\|_2$
  $\mathbf{x} = \mathbf{x} + [\mathbf{r}_0, \mathbf{r}_1, \dots, \mathbf{r}_{L-1}]\boldsymbol{\zeta}$
  $\mathbf{r} = \mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L]\boldsymbol{\zeta}, \quad \mathbf{p} = \mathbf{p}_0 - [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_L]\boldsymbol{\zeta}$
end while

Remark 1. There are several implementations (but mathematically equivalent methods) for Bi-CGstab(L).[11,12] Our presented scheme is based on algorithm 3.1 in Sleijpen and van Gijzen,[11] but with a minor change. Algorithm 3.1[11] generates $A\boldsymbol{p}_k$ with a vector update, whereas our variant computes it by explicitly multiplying $\boldsymbol{p}_k$ by A. The former approach is useful to express Bi-CGstab(L) in a similar fashion to the induced dimension reduction (IDR)-type methods.[11,13] However,

for a larger L, it can be one of the factors that induces a large residual gap (the difference between the recursively updated residual $\boldsymbol{r}_k$ and the explicitly computed residual $\boldsymbol{b} - A\boldsymbol{x}_k$) in finite precision arithmetic.[14] In such a case, the explicit multiplications with A in the latter approach help to reduce the residual gap. Therefore, we employ this formulation in the present study. For further discussions on the residual gap and its improvement, we refer to, for example, Aihara et al.,[14,15] Sleijpen and van der Vorst,[16] and van der Vorst and Ye.[17]

2.3 | Variant of GPBi-CG

Next, we derive a variant of the GPBi-CG algorithm. To this end, we slightly rewrite the stabilizing polynomials (3) as in the following formulation.

$$\begin{aligned} H_0(\lambda) &:= 1, \quad H_1(\lambda) := (1 - \zeta_0\lambda)H_0(\lambda), \\ H_{k+1}(\lambda) &:= (1 - \zeta_k\lambda)H_k(\lambda) - \eta_k\lambda G_{k-1}(\lambda), \end{aligned} \tag{17}$$

where $k = 1, 2, \ldots$, and the auxiliary polynomials $G_{k-1}(\lambda)$ are defined as

$$G_{k-1}(\lambda) := \frac{H_{k-1}(\lambda) - H_k(\lambda)}{\lambda}. \tag{18}$$

We now consider the updating process of iteration vectors. From (17), we have

$$\begin{aligned} \mathrm{H}_{k+1}\boldsymbol{r}_{k+1}^{bicg} &= (I - \zeta_k A)\mathrm{H}_k\boldsymbol{r}_{k+1}^{bicg} - \eta_k A\mathrm{G}_{k-1}\boldsymbol{r}_{k+1}^{bicg}, \\ \mathrm{H}_{k+1}\boldsymbol{p}_{k+1}^{bicg} &= (I - \zeta_k A)\mathrm{H}_k\boldsymbol{p}_{k+1}^{bicg} - \eta_k A\mathrm{G}_{k-1}\boldsymbol{p}_{k+1}^{bicg}, \end{aligned}$$

where $\mathrm{H}_k := H_k(A)$ and $\mathrm{G}_{k-1} := G_{k-1}(A)$. Thus, the residual and direction vectors of GPBi-CG can be expressed by

$$\begin{aligned} \boldsymbol{r}_{k+1} &= \boldsymbol{r}_k^{(1)} - \zeta_k A\boldsymbol{r}_k^{(1)} - \eta_k\boldsymbol{y}_k^{(1)}, \\ \boldsymbol{p}_{k+1} &= \boldsymbol{p}_k^{(1)} - \zeta_k A\boldsymbol{p}_k^{(1)} - \eta_k\boldsymbol{u}_k^{(1)}, \end{aligned}$$

where $\boldsymbol{r}_k^{(1)} := \mathrm{H}_k\boldsymbol{r}_{k+1}^{bicg}$ and $\boldsymbol{p}_k^{(1)} := \mathrm{H}_k\boldsymbol{p}_{k+1}^{bicg}$ as in (6), and $\boldsymbol{y}_k^{(1)} := A\mathrm{G}_{k-1}\boldsymbol{r}_{k+1}^{bicg}$ and $\boldsymbol{u}_k^{(1)} := A\mathrm{G}_{k-1}\boldsymbol{p}_{k+1}^{bicg}$ are also introduced. If these auxiliary vectors are available, the parameters ζ_k and η_k can be computed such that $\|\boldsymbol{r}_{k+1}\|_2$ is minimized (cf. section 5.3 in Zhang[8]). By introducing $\boldsymbol{z}_k^{(1)} := \mathrm{G}_{k-1}\boldsymbol{r}_{k+1}^{bicg}$, the associated approximation can be expressed by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k^{(1)} + \zeta_k\boldsymbol{r}_k^{(1)} + \eta_k\boldsymbol{z}_k^{(1)}.$$

Note that $\boldsymbol{y}_k^{(1)} = A\boldsymbol{z}_k^{(1)}$ holds by definitions.

Assume that we have $\boldsymbol{x}_k$, $\boldsymbol{r}_k := \mathrm{H}_k\boldsymbol{r}_k^{bicg}$, and $\boldsymbol{p}_k := \mathrm{H}_k\boldsymbol{p}_k^{bicg}$ at the beginning of the $(k+1)$th iteration. Then, we can obtain $\boldsymbol{x}_k^{(1)}$, $A^i\boldsymbol{r}_k^{(1)}$, and $A^i\boldsymbol{p}_k^{(1)}$ for $i = 0, 1$, by performing the Bi-CG step described in (16). Therefore, we consider how to generate the additional vectors $\boldsymbol{y}_k^{(1)}$, $\boldsymbol{u}_k^{(1)}$, and $\boldsymbol{z}_k^{(1)}$ at the $(k+1)$th iteration. From (4) and (18), we have the following relations.

$$\begin{aligned} A\mathrm{G}_{k-1}\boldsymbol{r}_k^{bicg} &= \mathrm{H}_{k-1}\boldsymbol{r}_k^{bicg} - \mathrm{H}_k\boldsymbol{r}_k^{bicg}, \\ A\mathrm{G}_{k-1}\boldsymbol{p}_k^{bicg} &= \mathrm{H}_{k-1}\boldsymbol{p}_k^{bicg} - \mathrm{H}_k\boldsymbol{p}_k^{bicg}, \\ A^2\mathrm{G}_{k-1}\boldsymbol{p}_k^{bicg} &= A\mathrm{H}_{k-1}\boldsymbol{p}_k^{bicg} - A\mathrm{H}_k\boldsymbol{p}_k^{bicg}, \\ A\mathrm{G}_{k-1}\boldsymbol{r}_{k+1}^{bicg} &= A\mathrm{G}_{k-1}\boldsymbol{r}_k^{bicg} - \alpha_k A^2\mathrm{G}_{k-1}\boldsymbol{p}_k^{bicg}, \\ A\mathrm{G}_{k-1}\boldsymbol{p}_{k+1}^{bicg} &= A\mathrm{G}_{k-1}\boldsymbol{r}_{k+1}^{bicg} - \beta_k A\mathrm{G}_{k-1}\boldsymbol{p}_k^{bicg}. \end{aligned} \tag{19}$$

By introducing the auxiliary vectors $\boldsymbol{y}_k := A\mathrm{G}_{k-1}\boldsymbol{r}_k^{bicg}$, $\boldsymbol{u}_k := A\mathrm{G}_{k-1}\boldsymbol{p}_k^{bicg}$, $\boldsymbol{v}_k := A^2\mathrm{G}_{k-1}\boldsymbol{p}_k^{bicg}$, and $\boldsymbol{q}_k := A\mathrm{H}_{k-1}\boldsymbol{p}_k^{bicg}$ into (19), we obtain the following formulas to compute the vectors $\boldsymbol{y}_k^{(1)}$ and $\boldsymbol{u}_k^{(1)}$.

$$\begin{aligned} \boldsymbol{y}_k &= \boldsymbol{r}_{k-1}^{(1)} - \boldsymbol{r}_k, \\ \boldsymbol{u}_k &= \boldsymbol{p}_{k-1}^{(1)} - \boldsymbol{p}_k, \end{aligned}$$

$$\begin{aligned}\boldsymbol{v}_k &= \boldsymbol{q}_k - A\boldsymbol{p}_k,\\ \boldsymbol{y}_k^{(1)} &= \boldsymbol{y}_k - \alpha_k \boldsymbol{v}_k,\\ \boldsymbol{u}_k^{(1)} &= \boldsymbol{y}_k^{(1)} - \beta_k \boldsymbol{u}_k.\end{aligned}$$

Here, note that $\boldsymbol{r}_{k-1}^{(1)}$, $\boldsymbol{p}_{k-1}^{(1)}$, and $\boldsymbol{q}_k(= A\boldsymbol{p}_{k-1}^{(1)})$ have already been generated in the previous iteration, and α_k, β_k, and $A\boldsymbol{p}_k$ are given in the Bi-CG step. Because $\mathrm{G}_k = \zeta_k \mathrm{H}_k + \eta_k \mathrm{G}_{k-1}$ holds from the definitions of the polynomials, we also have

$$\begin{aligned}\mathrm{G}_{k-1}\boldsymbol{r}_{k+1}^{bicg} &= \mathrm{G}_{k-1}\boldsymbol{r}_k^{bicg} - \alpha_k A\mathrm{G}_{k-1}\boldsymbol{p}_k^{bicg},\\ \mathrm{G}_k\boldsymbol{r}_{k+1}^{bicg} &= \zeta_k \mathrm{H}_k\boldsymbol{r}_{k+1}^{bicg} + \eta_k \mathrm{G}_{k-1}\boldsymbol{r}_{k+1}^{bicg},\end{aligned} \tag{20}$$

and we can compute $\boldsymbol{z}_k^{(1)}$ and $\boldsymbol{z}_k := \mathrm{G}_{k-1}\boldsymbol{r}_k^{bicg}$ recursively as follows.

$$\begin{aligned}\boldsymbol{z}_k^{(1)} &= \boldsymbol{z}_k - \alpha_k \boldsymbol{u}_k,\\ \boldsymbol{z}_{k+1} &= \zeta_k \boldsymbol{r}_k^{(1)} + \eta_k \boldsymbol{z}_k^{(1)}.\end{aligned}$$

Thus, all of the auxiliary vectors required for performing the iterations are obtained.

Algorithm 2 presents a variant of GPBi-CG. The first iteration of GPBi-CG actually corresponds to that of Bi-CGSTAB. Therefore, for ease of comprehension, we display one Bi-CGSTAB iteration separately from the iterations for GPBi-CG. For the notation, we refer to Algorithm 1. Note that the vectors $\boldsymbol{r}_{k-1}^{(1)}$, $\boldsymbol{p}_{k-1}^{(1)}$, and $\boldsymbol{q}_k(= A\boldsymbol{p}_{k-1}^{(1)})$ are saved as $\mathbf{r}'$, $\mathbf{p}'$, and $\mathbf{q}$, respectively, to be carried over to the next iteration.

Algorithm 2. Variant of GPBi-CG

Select an initial guess $\mathbf{x}$
Compute $\mathbf{r}_0 = \mathbf{b} - A \star \mathbf{x}$, and choose $\tilde{\mathbf{r}}_0$
Set $\mathbf{r} = [\mathbf{r}_0]$ and $\mathbf{p} = [\mathbf{r}_0]$
% One iteration of Bi-CGSTAB
$\mathbf{p} = [\mathbf{p}; A \star \mathbf{p}_0]$
$\sigma = \tilde{\mathbf{r}}_0^\top \mathbf{p}_1, \quad \alpha = (\tilde{\mathbf{r}}_0^\top \mathbf{r}_0)/\sigma$
$\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}_0, \quad \mathbf{r} = \mathbf{r} - \alpha \mathbf{p}_1, \quad \mathbf{r} = [\mathbf{r}; A \star \mathbf{r}_0]$
$\beta = (\tilde{\mathbf{r}}_0^\top \mathbf{r}_1)/\sigma, \quad \mathbf{p} = \mathbf{r} - \beta \mathbf{p}$
Set $\mathbf{r}' = \mathbf{r_0}$, $\mathbf{p}' = \mathbf{p}_0$, and $\mathbf{q} = \mathbf{p}_1$
Compute ζ to minimize $\|\mathbf{r}_0 - \zeta \mathbf{r}_1\|_2$
$\mathbf{z} = \zeta \mathbf{r}_0, \quad \mathbf{x} = \mathbf{x} + \mathbf{z}$
$\mathbf{r} = \mathbf{r}_0 - \zeta \mathbf{r}_1, \quad \mathbf{p} = \mathbf{p}_0 - \zeta \mathbf{p}_1$
% Iterations for GPBi-CG
while $\|\mathbf{r}_0\|_2 > tol$ **do**
    $\mathbf{y} = \mathbf{r}' - \mathbf{r}, \quad \mathbf{u} = \mathbf{p}' - \mathbf{p}$
    $\mathbf{p} = [\mathbf{p}; A \star \mathbf{p}_0], \quad \mathbf{v} = \mathbf{q} - \mathbf{p}_1$
    $\sigma = \tilde{\mathbf{r}}_0^\top \mathbf{p}_1, \quad \alpha = (\tilde{\mathbf{r}}_0^\top \mathbf{r}_0)/\sigma$
    $\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}_0, \quad \mathbf{z} = \mathbf{z} - \alpha \mathbf{u}, \quad \mathbf{y} = \mathbf{y} - \alpha \mathbf{v}$
    $\mathbf{r} = \mathbf{r} - \alpha \mathbf{p}_1, \quad \mathbf{r} = [\mathbf{r}; A \star \mathbf{r}_0]$
    $\beta = (\tilde{\mathbf{r}}_0^\top \mathbf{r}_1)/\sigma, \quad \mathbf{p} = \mathbf{r} - \beta \mathbf{p}, \quad \mathbf{u} = \mathbf{y} - \beta \mathbf{u}$
    Set $\mathbf{r}' = \mathbf{r}_0$, $\mathbf{p}' = \mathbf{p}_0$, and $\mathbf{q} = \mathbf{p}_1$
    Compute ζ and η to minimize $\|\mathbf{r}_0 - \zeta \mathbf{r}_1 - \eta \mathbf{y}\|_2$
    $\mathbf{z} = \zeta \mathbf{r}_0 + \eta \mathbf{z}, \quad \mathbf{x} = \mathbf{x} + \mathbf{z}$
    $\mathbf{r} = \mathbf{r}_0 - \zeta \mathbf{r}_1 - \eta \mathbf{y}, \quad \mathbf{p} = \mathbf{p}_0 - \zeta \mathbf{p}_1 - \eta \mathbf{u}$
end while

Remark 2. Some related methods, which are mathematically equivalent to GPBi-CG but use different implementations, have been developed.[2,10,18,19] Algorithm 2 is also a mathematically equivalent implementation to them but uses partly different formulations from either. We note that our variant is derived from a novel perspective. We see from (17) that

the stabilizing polynomial $H_{k+1}(\lambda)$ consists of two parts: one is the multiplication of one-degree polynomial $(1-\zeta_k\lambda)$ by $H_k(\lambda)$ and the other is a relaxation term $-\eta_k\lambda G_{k-1}(\lambda) = \eta_k(H_k(\lambda) - H_{k-1}(\lambda))$. This interpretation is useful to incorporate higher order polynomials into GPBi-CG and to make the new method described in the next section.

3 | GPBI-CGSTAB(L)

In this section, we propose GPBi-CGstab(L), which unifies Bi-CGstab(L) and GPBi-CG described above. The basic idea in deriving the new method is to extend the polynomials (2) and (17) to the following.

$$\begin{aligned} H_0(\lambda) &:= 1, \quad H_L(\lambda) := (1-\zeta_{0,1}\lambda - \cdots - \zeta_{0,L}\lambda^L)H_0(\lambda), \\ H_{k+L}(\lambda) &:= (1-\zeta_{k,1}\lambda - \cdots - \zeta_{k,L}\lambda^L)H_k(\lambda) - \eta_k\lambda G_{k-1}(\lambda), \end{aligned} \tag{21}$$

where k is a multiple of L and the auxiliary polynomials $G_{k-1}(\lambda)$ are defined as

$$G_{k-1}(\lambda) := \frac{H_{k-L}(\lambda) - H_k(\lambda)}{\lambda}. \tag{22}$$

The coefficients $\zeta_{k,j} \in \mathbb{R}$ for $j = 1, 2, \ldots, L$ and $\eta_k \in \mathbb{R}$ are independent parameters. Obviously, (21) simplifies to (2) when $\eta_k = 0$ (i.e., the relaxation term $-\eta_k\lambda G_{k-1}(\lambda)$ vanishes), and (21) is equivalent to (17) (and (3)) in the case of $L = 1$. GPBi-CGstab(L) is a novel LTPM that uses the stabilizing polynomials defined by (21) with the $L+1$ parameters $\zeta_{k,1}, \ldots, \zeta_{k,L}, \eta_k$.

3.1 | Recursion formulas for iteration vectors

Let $\mathrm{H}_k := H_k(A)$ and $\mathrm{G}_{k-1} := G_{k-1}(A)$ as before. As in Bi-CGstab(L), the residual $\boldsymbol{r}_k = \mathrm{H}_k\boldsymbol{r}_k^{bicg}$ and the direction vector $\boldsymbol{p}_k = \mathrm{H}_k\boldsymbol{p}_k^{bicg}$ are updated to $\boldsymbol{r}_{k+L} = \mathrm{H}_{k+L}\boldsymbol{r}_{k+L}^{bicg}$ and $\boldsymbol{p}_{k+L} = \mathrm{H}_{k+L}\boldsymbol{p}_{k+L}^{bicg}$, respectively, and this process is counted as one cycle of GPBi-CGstab(L). From (21), we have

$$\begin{aligned} \mathrm{H}_{k+L}\boldsymbol{r}_{k+L}^{bicg} &= (I - \zeta_{k,1}A - \cdots - \zeta_{k,L}A^L)\mathrm{H}_k\boldsymbol{r}_{k+L}^{bicg} - \eta_k A\mathrm{G}_{k-1}\boldsymbol{r}_{k+L}^{bicg}, \\ \mathrm{H}_{k+L}\boldsymbol{p}_{k+L}^{bicg} &= (I - \zeta_{k,1}A - \cdots - \zeta_{k,L}A^L)\mathrm{H}_k\boldsymbol{p}_{k+L}^{bicg} - \eta_k A\mathrm{G}_{k-1}\boldsymbol{p}_{k+L}^{bicg}, \end{aligned}$$

and the updated residual and direction vectors are expressed by

$$\begin{aligned} \boldsymbol{r}_{k+1} &= \boldsymbol{r}_k^{(L)} - \zeta_{k,1}A\boldsymbol{r}_k^{(L)} - \cdots - \zeta_{k,L}A^L\boldsymbol{r}_k^{(L)} - \eta_k\boldsymbol{y}_k^{(L)}, \\ \boldsymbol{p}_{k+1} &= \boldsymbol{p}_k^{(L)} - \zeta_{k,1}A\boldsymbol{p}_k^{(L)} - \cdots - \zeta_{k,L}A^L\boldsymbol{p}_k^{(L)} - \eta_k\boldsymbol{u}_k^{(L)}, \end{aligned} \tag{23}$$

where $\boldsymbol{r}_k^{(L)} := \mathrm{H}_k\boldsymbol{r}_{k+L}^{bicg}$, $\boldsymbol{p}_k^{(L)} := \mathrm{H}_k\boldsymbol{p}_{k+L}^{bicg}$, $\boldsymbol{y}_k^{(L)} := A\mathrm{G}_{k-1}\boldsymbol{r}_{k+L}^{bicg}$, and $\boldsymbol{u}_k^{(L)} := A\mathrm{G}_{k-1}\boldsymbol{p}_{k+L}^{bicg}$. The associated approximation can be expressed as

$$\boldsymbol{x}_{k+L} = \boldsymbol{x}_k^{(L)} + \zeta_{k,1}\boldsymbol{r}_k^{(L)} + \cdots + \zeta_{k,L}A^{L-1}\boldsymbol{r}_k^{(L)} + \eta_k\boldsymbol{z}_k^{(L)}, \tag{24}$$

where $\boldsymbol{z}_k^{(L)} := \mathrm{G}_{k-1}\boldsymbol{r}_{k+L}^{bicg}$, and $\boldsymbol{y}_k^{(L)} = A\boldsymbol{z}_k^{(L)}$ holds.

When the vectors $\boldsymbol{x}_k^{(0)} := \boldsymbol{x}_k$, $\boldsymbol{r}_k^{(0)} := \boldsymbol{r}_k = \mathrm{H}_k\boldsymbol{r}_k^{bicg}$, and $\boldsymbol{p}_k^{(0)} := \boldsymbol{p}_k = \mathrm{H}_k\boldsymbol{p}_k^{bicg}$ are given at the beginning of the cycle, $\boldsymbol{x}_k^{(j)}$, $A^i\boldsymbol{r}_k^{(j)} = A^i\mathrm{H}_k\boldsymbol{r}_{k+j}^{bicg}$, and $A^i\boldsymbol{p}_k^{(j)} = A^i\mathrm{H}_k\boldsymbol{p}_{k+j}^{bicg}$ for $i = 0, 1, \ldots, j$ with $j = 1, 2, \ldots, L$ can be generated by repeating the Bi-CG step as in Bi-CGstab(L) (see (9)–(15)). Therefore, here we consider computations for the additional vectors $\boldsymbol{y}_k^{(L)}$, $\boldsymbol{u}_k^{(L)}$, and $\boldsymbol{z}_k^{(L)}$. As an extension of (19), we can use the following recurrence relations.

$$\begin{aligned} A\mathrm{G}_{k-1}\boldsymbol{r}_k^{bicg} &= \mathrm{H}_{k-L}\boldsymbol{r}_k^{bicg} - \mathrm{H}_k\boldsymbol{r}_k^{bicg}, \\ A\mathrm{G}_{k-1}\boldsymbol{p}_k^{bicg} &= \mathrm{H}_{k-L}\boldsymbol{p}_k^{bicg} - \mathrm{H}_k\boldsymbol{p}_k^{bicg}. \\ &\text{for } j = 1, 2, \ldots, L \text{ do} \\ A^2\mathrm{G}_{k-1}\boldsymbol{p}_{k+j-1}^{bicg} &= A\mathrm{H}_{k-L}\boldsymbol{p}_{k+j-1}^{bicg} - A\mathrm{H}_k\boldsymbol{p}_{k+j-1}^{bicg}, \end{aligned}$$

$$
\begin{aligned}
A\mathrm{G}_{k-1}\boldsymbol{r}_{k+j}^{bicg} &= A\mathrm{G}_{k-1}\boldsymbol{r}_{k+j-1}^{bicg} - \alpha_{k+j-1}A^2\mathrm{G}_{k-1}\boldsymbol{p}_{k+j-1}^{bicg},\\
A\mathrm{G}_{k-1}\boldsymbol{p}_{k+j}^{bicg} &= A\mathrm{G}_{k-1}\boldsymbol{r}_{k+j}^{bicg} - \beta_{k+j-1}A\mathrm{G}_{k-1}\boldsymbol{p}_{k+j-1}^{bicg}.\\
&\text{end for}
\end{aligned}
\tag{25}
$$

By introducing the intermediate auxiliary vectors $\boldsymbol{y}_k^{(j)} := A\mathrm{G}_{k-1}\boldsymbol{r}_{k+j}^{bicg}$, $\boldsymbol{u}_k^{(j)} := A\mathrm{G}_{k-1}\boldsymbol{p}_{k+j}^{bicg}$, $\boldsymbol{v}_k^{(j)} := A^2\mathrm{G}_{k-1}\boldsymbol{p}_{k+j}^{bicg}$, and $\boldsymbol{q}_k^{(j)} := A\mathrm{H}_{k-L}\boldsymbol{p}_{k+j}^{bicg}$ into (25), and rewriting the coefficients as $\alpha_k^{(j)} := \alpha_{k+j}$ and $\beta_k^{(j)} := \beta_{k+j}$, we obtain the following recursions to compute $\boldsymbol{y}_k^{(L)}$ and $\boldsymbol{u}_k^{(L)}$.

$$
\begin{aligned}
\boldsymbol{y}_k^{(0)} &= \boldsymbol{r}_{k-L}^{(L)} - \boldsymbol{r}_k,\\
\boldsymbol{u}_k^{(0)} &= \boldsymbol{p}_{k-L}^{(L)} - \boldsymbol{p}_k.\\
&\text{for } j = 1, 2, \ldots, L \text{ do}\\
\boldsymbol{v}_k^{(j-1)} &= \boldsymbol{q}_k^{(j-1)} - A\boldsymbol{p}_k^{(j-1)},\\
\boldsymbol{y}_k^{(j)} &= \boldsymbol{y}_k^{(j-1)} - \alpha_k^{(j-1)}\boldsymbol{v}_k^{(j-1)},\\
\boldsymbol{u}_k^{(j)} &= \boldsymbol{y}_k^{(j)} - \beta_k^{(j-1)}\boldsymbol{u}_k^{(j-1)}.\\
&\text{end for}
\end{aligned}
\tag{26}
$$

Here, except for $\boldsymbol{q}_k^{(j-1)}$ with $j > 1$, the vectors and the Bi-CG coefficients for performing (26) are available. Note that $\boldsymbol{r}_{k-L}^{(L)}$, $\boldsymbol{p}_{k-L}^{(L)}$, and $\boldsymbol{q}_k^{(0)} (= A\boldsymbol{p}_{k-L}^{(L)})$ have been generated in the previous cycle, and $\alpha_k^{(j-1)}$, $\beta_k^{(j-1)}$, and $A\boldsymbol{p}_k^{(j-1)}$ are given at the jth repetition of the Bi-CG step.

We therefore construct additional recursions to compute $\boldsymbol{q}_k^{(j-1)}$ for $j > 1$. Based on (8), we employ the following relations.

$$
\begin{aligned}
&\text{for } j = 2, 3, \ldots, L \text{ do}\\
&A^i\mathrm{H}_{k-L}\boldsymbol{r}_{k+j-1}^{bicg} = A^i\mathrm{H}_{k-L}\boldsymbol{r}_{k+j-2}^{bicg} - \alpha_{k+j-2}A^{i+1}\mathrm{H}_{k-L}\boldsymbol{p}_{k+j-2}^{bicg},\\
&A^i\mathrm{H}_{k-L}\boldsymbol{p}_{k+j-1}^{bicg} = A^i\mathrm{H}_{k-L}\boldsymbol{r}_{k+j-1}^{bicg} - \beta_{k+j-2}A^i\mathrm{H}_{k-L}\boldsymbol{p}_{k+j-2}^{bicg}.\\
&\text{end for}
\end{aligned}
$$

Introducing $\boldsymbol{s}_k^{(j)} := A\mathrm{H}_{k-L}\boldsymbol{r}_{k+j}^{bicg}$ gives

$$
\begin{aligned}
&\text{for } j = 2, 3, \ldots, L \text{ do}\\
&A^{i-1}\boldsymbol{s}_k^{(j-1)} = A^{i-1}\boldsymbol{s}_k^{(j-2)} - \alpha_k^{(j-2)}A^i\boldsymbol{q}_k^{(j-2)}, \qquad (27)\\
&A^{i-1}\boldsymbol{q}_k^{(j-1)} = A^{i-1}\boldsymbol{s}_k^{(j-1)} - \beta_k^{(j-2)}A^{i-1}\boldsymbol{q}_k^{(j-2)}. \qquad (28)\\
&\text{end for}
\end{aligned}
$$

Here, the index i is set to $i = 1, 2, \ldots, L - j + 1$ for each $j > 1$. Note that, in addition to $\boldsymbol{q}_k^{(0)} (= A\boldsymbol{p}_{k-L}^{(L)})$, the starting vectors $A^{i-1}\boldsymbol{s}_k^{(0)} (= A^i\boldsymbol{r}_{k-L}^{(L)})$ and $A^i\boldsymbol{q}_k^{(0)} (= A^{i+1}\boldsymbol{p}_{k-L}^{(L)})$ for $i = 1, 2, \ldots, L-1$ were also obtained in the previous cycle. The computational schemes of (27) and (28) at the jth repetition ($j = 2, 3, 4$) with $L = 4$ are summarized as follows.

- Computational scheme to obtain $\boldsymbol{q}_k^{(1)}$ at the second repetition ($j = 2$) with $L = 4$:

$$
\begin{array}{ccccc}
\boldsymbol{q}_k^{(0)} & & & & \boldsymbol{q}_k^{(1)}\\
 & & & \overset{\text{Equation (28)}}{\nearrow}_{i=1} & \\
A\boldsymbol{q}_k^{(0)} & \boldsymbol{s}_k^{(0)} & \xrightarrow[i=1]{\text{Equation (27)}} & \boldsymbol{s}_k^{(1)} & A\boldsymbol{q}_k^{(1)}\\
 & & & \overset{\text{Equation (28)}}{\nearrow}_{i=2} & \\
A^2\boldsymbol{q}_k^{(0)} & A\boldsymbol{s}_k^{(0)} & \xrightarrow[i=2]{\text{Equation (27)}} & A\boldsymbol{s}_k^{(1)} & A^2\boldsymbol{q}_k^{(1)}\\
 & & & \overset{\text{Equation (28)}}{\nearrow}_{i=3} & \\
A^3\boldsymbol{q}_k^{(0)} & A^2\boldsymbol{s}_k^{(0)} & \xrightarrow[i=3]{\text{Equation (27)}} & A^2\boldsymbol{s}_k^{(1)} &
\end{array}
$$

- Computational scheme to obtain $\boldsymbol{q}_k^{(2)}$ at the third repetition ($j = 3$) with $L = 4$:

$$\begin{array}{ccccc} \boldsymbol{q}_k^{(1)} & & & & \boldsymbol{q}_k^{(2)} \\ & & & \overset{\text{Equation (28)}}{\nearrow}{}_{i=1} & \\ A\boldsymbol{q}_k^{(1)} & \boldsymbol{s}_k^{(1)} & \xrightarrow[i=1]{\text{Equation (27)}} & \boldsymbol{s}_k^{(2)} & A\boldsymbol{q}_k^{(2)} \\ & & & \overset{\text{Equation (28)}}{\nearrow}{}_{i=2} & \\ A^2\boldsymbol{q}_k^{(1)} & A\boldsymbol{s}_k^{(1)} & \xrightarrow[i=2]{\text{Equation (27)}} & A\boldsymbol{s}_k^{(2)} & \end{array}$$

- Computational scheme to obtain $\boldsymbol{q}_k^{(3)}$ at the fourth repetition ($j = 4$) with $L = 4$:

$$\begin{array}{ccccc} \boldsymbol{q}_k^{(2)} & & & & \boldsymbol{q}_k^{(3)} \\ & & & \overset{\text{Equation(28)}}{\nearrow}{}_{i=1} & \\ A\boldsymbol{q}_k^{(2)} & \boldsymbol{s}_k^{(2)} & \xrightarrow[i=1]{\text{Equation (27)}} & \boldsymbol{s}_k^{(3)} & \end{array}$$

Note that in contrast to the repetitions for (9)–(13), the number of vector updates decreases with an increase in j.

Next, we consider the recursion formulas to generate $\boldsymbol{z}_k^{(L)}$. From (21) and (22), it holds that

$$\mathrm{G}_{k+L-1} = (\zeta_{k,1}I + \zeta_{k,2}A + \cdots + \zeta_{k,L}A^{L-1})\mathrm{H}_k + \eta_k \mathrm{G}_{k-1},$$

and we can extend (20) as follows.

$$\begin{aligned} &\text{for } j = 1, 2, \ldots, L \text{ do} \\ &\quad \mathrm{G}_{k-1}\boldsymbol{r}_{k+j}^{bicg} = \mathrm{G}_{k-1}\boldsymbol{r}_{k+j-1}^{bicg} - \alpha_{k+j-1}A\mathrm{G}_{k-1}\boldsymbol{p}_{k+j-1}^{bicg}. \\ &\text{end for} \\ &\mathrm{G}_{k+L-1}\boldsymbol{r}_{k+L}^{bicg} = (\zeta_{k,1}I + \zeta_{k,2}A + \cdots + \zeta_{k,L}A^{L-1})\mathrm{H}_k\boldsymbol{r}_{k+L}^{bicg} + \eta_k \mathrm{G}_{k-1}\boldsymbol{r}_{k+L}^{bicg}. \end{aligned}$$

Therefore, staring with $\boldsymbol{z}_k^{(0)} := \mathrm{G}_{k-1}\boldsymbol{r}_k^{bicg}$, we can compute $\boldsymbol{z}_k^{(j)} := \mathrm{G}_{k-1}\boldsymbol{r}_{k+j}^{bicg}$ for $j = 1, 2, \ldots, L$ and $\boldsymbol{z}_{k+L}^{(0)} = \mathrm{G}_{k+L-1}\boldsymbol{r}_{k+L}^{bicg}$ (the next starting vector) recursively as follows.

$$\begin{aligned} &\text{for } j = 1, 2, \ldots, L \text{ do} \\ &\quad \boldsymbol{z}_k^{(j)} = \boldsymbol{z}_k^{(j-1)} - \alpha_k^{(j-1)}\boldsymbol{u}_k^{(j-1)}. \end{aligned} \tag{29}$$

$$\begin{aligned} &\text{end for} \\ &\boldsymbol{z}_{k+L}^{(0)} = \zeta_{k,1}\boldsymbol{r}_k^{(L)} + \zeta_{k,2}A\boldsymbol{r}_k^{(L)} + \cdots + \zeta_{k,L}A^{L-1}\boldsymbol{r}_k^{(L)} + \eta_k\boldsymbol{z}_k^{(L)}. \end{aligned} \tag{30}$$

Here, $\boldsymbol{u}_k^{(j-1)}$ for $j = 1, 2, \ldots, L$ are given in (26). We now have all the recursion formulas for updating the iteration vectors.

3.2 | Choice of parameters and a specific algorithm

We describe a practical choice for the parameters $\zeta_{k,1}, \ldots, \zeta_{k,L}, \eta_k$. As in Bi-CGstab($L$) and GPBi-CG, it is natural to determine the $L + 1$ parameters to minimize $\|\boldsymbol{r}_{k+L}\|_2$ at each cycle; that is, we solve

$$\min_{\zeta_k, \eta_k} \|\boldsymbol{r}_k^{(L)} - [A\boldsymbol{r}_k^{(L)}, \ldots, A^L\boldsymbol{r}_k^{(L)}]\boldsymbol{\zeta}_k - \eta_k\boldsymbol{y}_k^{(L)}\|_2, \tag{31}$$

for $\boldsymbol{\zeta}_k := [\zeta_{k,1}, \dots, \zeta_{k,L}]^\top \in \mathbb{R}^L$ and $\eta_k \in \mathbb{R}$. It is well-known that the least squares problem (31) can be converted into the normal equation

$$B_k^\top B_k \begin{bmatrix} \boldsymbol{\zeta}_k \\ \eta_k \end{bmatrix} = B_k^\top \boldsymbol{r}_k^{(L)}, \quad B_k := [A\boldsymbol{r}_k^{(L)}, \dots, A^L \boldsymbol{r}_k^{(L)}, \boldsymbol{y}_k^{(L)}] \in \mathbb{R}^{n\times(L+1)}. \tag{32}$$

For a modest value of L, (32) is easy to solve with a direct method, such as the Cholesky factorization method. As noted in Sleijpen et al.,[12] for stability reasons, it would also be useful to solve (31) directly with the QR factorization method if B_k is ill-conditioned.

Algorithm 3 presents a GPBi-CGstab(L) algorithm. Because the first cycle of GPBi-CGstab(L) corresponds to that of Bi-CGstab(L), it is displayed separately, as in Algorithm 2. To perform the cycles of GPBi-CGstab(L), the vector updates 26–29 are combined with the Bi-CG steps of Bi-CGstab(L) (Algorithm 1). Then, after the determination of the parameters by (31), the updates (23), (24), and (30) are performed. Note that the vectors $\boldsymbol{r}_{k-L}^{(L)}$, $\boldsymbol{p}_{k-L}^{(L)}$, $A^{i-1}\boldsymbol{s}_k^{(0)} (= A^i \boldsymbol{r}_{k-L}^{(L)})$, and $A^i \boldsymbol{q}_k^{(0)} (= A^{i+1}\boldsymbol{p}_{k-L}^{(L)})$ are saved as $\mathbf{r}'$, $\mathbf{p}'$, $\mathbf{s}_{i-1}$, and $\mathbf{q}_i$, respectively, to be carried over to the next cycle.

Algorithm 3. GPBi-CGstab(L)

Select an initial guess $\mathbf{x}$
Compute $\mathbf{r}_0 = \mathbf{b} - A \star \mathbf{x}$, and choose $\tilde{\mathbf{r}}_0$
Set $\mathbf{r} = [\mathbf{r}_0]$ and $\mathbf{p} = [\mathbf{r}_0]$
% One cycle of Bi-CGstab(L)
$\rho = \tilde{\mathbf{r}}_0^\top \mathbf{r}_0$
for $j = 1, 2, \dots, L$ **do**
  $\mathbf{p} = [\mathbf{p}; A \star \mathbf{p}_{j-1}]$
  $\sigma = \tilde{\mathbf{r}}_0^\top \mathbf{p}_j, \quad \alpha = \rho/\sigma$
  $\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}_0, \quad \mathbf{r} = \mathbf{r} - \alpha[\mathbf{p}_1; \mathbf{p}_2; \dots; \mathbf{p}_j], \quad \mathbf{r} = [\mathbf{r}; A \star \mathbf{r}_{j-1}]$
  $\rho = \tilde{\mathbf{r}}_0^\top \mathbf{r}_j, \quad \beta = \rho/\sigma, \quad \mathbf{p} = \mathbf{r} - \beta \mathbf{p}$
end for
Set $\mathbf{r}' = \mathbf{r}_0$, $\mathbf{p}' = \mathbf{p}_0$, $\mathbf{s} = [\mathbf{r}_1; \mathbf{r}_2; \dots; \mathbf{r}_{L-1}]$, and $\mathbf{q} = [\mathbf{p}_1; \mathbf{p}_2; \dots; \mathbf{p}_L]$
Compute $\boldsymbol{\zeta} = [\zeta_1; \zeta_2; \dots; \zeta_L]$ to minimize $\|\mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L]\boldsymbol{\zeta}\|_2$
$\mathbf{z} = [\mathbf{r}_0, \mathbf{r}_1, \dots, \mathbf{r}_{L-1}]\boldsymbol{\zeta}, \quad \mathbf{x} = \mathbf{x} + \mathbf{z}$
$\mathbf{r} = \mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L]\boldsymbol{\zeta}, \quad \mathbf{p} = \mathbf{p}_0 - [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_L]\boldsymbol{\zeta}$
% Cycles for GPBi-CGstab(L)
while $\|\mathbf{r}_0\|_2 > tol$ **do**
  $\mathbf{y} = \mathbf{r}' - \mathbf{r}, \quad \mathbf{u} = \mathbf{p}' - \mathbf{p}$
  $\rho = \tilde{\mathbf{r}}_0^\top \mathbf{r}_0$
  for $j = 1, 2, \dots, L$ **do**
    if $j > 1$ **then**
      $\mathbf{s} = [\mathbf{s}_0; \mathbf{s}_1; \dots; \mathbf{s}_{L-j}] - \alpha[\mathbf{q}_1; \mathbf{q}_2; \dots; \mathbf{q}_{L-j+1}]$
      $\mathbf{q} = \mathbf{s} - \beta[\mathbf{q}_0; \mathbf{q}_1; \dots; \mathbf{q}_{L-j}]$
    end if
    $\mathbf{p} = [\mathbf{p}; A \star \mathbf{p}_{j-1}], \quad \mathbf{v} = \mathbf{q}_0 - \mathbf{p}_1$
    $\sigma = \tilde{\mathbf{r}}_0^\top \mathbf{p}_j, \quad \alpha = \rho/\sigma$
    $\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}_0, \quad \mathbf{z} = \mathbf{z} - \alpha \mathbf{u}, \quad \mathbf{y} = \mathbf{y} - \alpha \mathbf{v}$
    $\mathbf{r} = \mathbf{r} - \alpha[\mathbf{p}_1; \mathbf{p}_2; \dots; \mathbf{p}_j], \quad \mathbf{r} = [\mathbf{r}; A \star \mathbf{r}_{j-1}]$
    $\rho = \tilde{\mathbf{r}}_0^\top \mathbf{r}_j, \quad \beta = \rho/\sigma, \quad \mathbf{p} = \mathbf{r} - \beta \mathbf{p}, \quad \mathbf{u} = \mathbf{y} - \beta \mathbf{u}$
  end for
  Set $\mathbf{r}' = \mathbf{r}_0$, $\mathbf{p}' = \mathbf{p}_0$, $\mathbf{s} = [\mathbf{r}_1; \mathbf{r}_2; \dots; \mathbf{r}_{L-1}]$, and $\mathbf{q} = [\mathbf{p}_1; \mathbf{p}_2; \dots; \mathbf{p}_L]$
  Compute $\boldsymbol{\zeta} = [\zeta_1; \zeta_2; \dots; \zeta_L]$ and η to minimize $\|\mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L]\boldsymbol{\zeta} - \eta \mathbf{y}\|_2$
  $\mathbf{z} = [\mathbf{r}_0, \mathbf{r}_1, \dots, \mathbf{r}_{L-1}]\boldsymbol{\zeta} + \eta \mathbf{z}, \quad \mathbf{x} = \mathbf{x} + \mathbf{z}$
  $\mathbf{r} = \mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_L]\boldsymbol{\zeta} - \eta \mathbf{y}, \quad \mathbf{p} = \mathbf{p}_0 - [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_L]\boldsymbol{\zeta} - \eta \mathbf{u}$
end while

It is evident that GPBi-CGstab(L) (Algorithm 3) simplifies to Bi-CGstab(L) (Algorithm 1) if $\eta_k = 0$, and to GPBi-CG (Algorithm 2) if $L = 1$. Consequently, CGS, Bi-CGSTAB, and Bi-CGStab2, can also be viewed as special cases of the presented framework; for these derivations, we refer to Zhang.[8]

Our variants of Bi-CGstab(L) (Algorithm 1) and GPBi-CG (Algorithm 2) are mathematically equivalent methods to the original Bi-CGstab(L)[6] and GPBi-CG,[8] respectively, but the variants use different implementations and require slightly more computational costs as shown in Section 3.4. However, the essence of the presented variants is to enable us to unify the two conventional frameworks (i.e., Bi-CGstab(L) and GPBi-CG), and to derive the mathematically novel method GPBi-CGstab(L). Because our main purpose is to provide this new framework of LTPMs, we will not discuss enhanced (but mathematically equivalent) implementations of Bi-CGstab(L) and GPBi-CG themselves; for refined variants of them, see, for example, Sleijpen et al.[12] and Abe and Sleijpen.[10]

3.3 | Advantage of the comprehensive stabilizing polynomials

We discuss advantages of the stabilizing polynomials of GPBi-CGstab(L) compared to those of the underlying Bi-CGstab(L) and GPBi-CG. Let $\boldsymbol{r}_k^{-stab(L)}$, $\boldsymbol{r}_k^{gp-}$, and $\boldsymbol{r}_k^{gp-stab(L)}$ be residuals obtained by Bi-CGstab(L), GPBi-CG, and GPBi-CGstab(L), respectively, where k indicates a multiple of L. Similarly, we distinguish the stabilizing polynomials of the methods by using the superscripts "$-stab(L)$," "$gp-$," and "$gp-stab(L)$."

We first confirm the relations between the above residuals and the Krylov subspace. We recall from the definitions (21) and (22) that the residual of GPBi-CGstab(L) can be expressed by

$$\mathrm{H}_{k+L}^{gp-stab(L)}\boldsymbol{r}_{k+L}^{bicg} = (I - \zeta_{k,1}A - \cdots - \zeta_{k,L}A^L)\mathrm{H}_k^{gp-stab(L)}\boldsymbol{r}_{k+L}^{bicg} - \eta_k(\mathrm{H}_{k-L}^{gp-stab(L)} - \mathrm{H}_k^{gp-stab(L)})\boldsymbol{r}_{k+L}^{bicg}. \tag{33}$$

The degree of the polynomial increases by L with updating from $\mathrm{H}_k^{gp-stab(L)}$ to $\mathrm{H}_{k+L}^{gp-stab(L)}$, because $\mathrm{H}_{k+L}^{gp-stab(L)}$ consists of multiplying a Lth degree polynomial by the previous $\mathrm{H}_k^{gp-stab(L)}$ and adding the relaxation term of the same degree as $\mathrm{H}_k^{gp-stab(L)}$. In other words, when we have the residual $\boldsymbol{r}_k^{gp-stab(L)} = \mathrm{H}_k^{gp-stab(L)}\boldsymbol{r}_k^{bicg} \in \mathcal{K}_{2k+1}(A, \boldsymbol{r}_0)$, it is updated to $\boldsymbol{r}_{k+L}^{gp-stab(L)} = \mathrm{H}_{k+L}^{gp-stab(L)}\boldsymbol{r}_{k+L}^{bicg} \in \mathcal{K}_{2(k+L)+1}(A, \boldsymbol{r}_0)$ by performing one cycle of the method. This property holds for Bi-CGstab(L) deriving from the choice $\eta_k = 0$; that is, if $\boldsymbol{r}_k^{-stab(L)} = \mathrm{H}_k^{-stab(L)}\boldsymbol{r}_k^{bicg} \in \mathcal{K}_{2k+1}(A, \boldsymbol{r}_0)$, then it holds that $\boldsymbol{r}_{k+L}^{-stab(L)} = \mathrm{H}_{k+L}^{-stab(L)}\boldsymbol{r}_{k+L}^{bicg} \in \mathcal{K}_{2(k+L)+1}(A, \boldsymbol{r}_0)$. Thus, the residuals of Bi-CGstab(L) and GPBi-CGstab(L) are updated in the same Krylov subspace. In GPBi-CG (= GPBi-CGstab(1)), because the degree of the stabilizing polynomial increases by one at each iteration, the residual $\boldsymbol{r}_k^{gp-} = \mathrm{H}_k^{gp-}\boldsymbol{r}_k^{bicg} \in \mathcal{K}_{2k+1}(A, \boldsymbol{r}_0)$ is updated to $\boldsymbol{r}_{k+L}^{gp-} = \mathrm{H}_{k+L}^{gp-}\boldsymbol{r}_{k+L}^{bicg} \in \mathcal{K}_{2(k+L)+1}(A, \boldsymbol{r}_0)$ by performing L iterations.

A strength of GPBi-CGstab(L), compared to GPBi-CG, is that it can exploit higher degree polynomials like Bi-CGstab(L). The GPBi-CG residual is updated from $\boldsymbol{r}_k^{gp-} \in \mathcal{K}_{2k+1}(A, \boldsymbol{r}_0)$ to $\boldsymbol{r}_{k+L}^{gp-} \in \mathcal{K}_{2(k+L)+1}(A, \boldsymbol{r}_0)$ with L iterations as mentioned above, where residual minimization over two-dimensional space is performed L times. On the other hand, GPBi-CGstab(L) performs $L + 1$-dimensional minimization when updating $\boldsymbol{r}_k^{gp-stab(L)} \in \mathcal{K}_{2k+1}(A, \boldsymbol{r}_0)$ to $\boldsymbol{r}_{k+L}^{gp-stab(L)} \in \mathcal{K}_{2(k+L)+1}(A, \boldsymbol{r}_0)$. We expect that increasing the parameter L enhances the convergence of GPBi-CGstab(L) as in Bi-CGstab(L).

Then, we consider effects of the relaxation term that is a common part with GPBi-CG (but not in Bi-CGstab(L)). In (33), if the Lth degree polynomial $(I - \zeta_{k,1}A - \cdots - \zeta_{k,L}A^L)$ is a significant factor to minimize the residual, η_k is necessarily selected to be small and the influence of the relaxation term would be almost eliminated. Otherwise, the $L + 1$ parameters including η_k are appropriately selected to locally minimize the residual; the updated GPBi-CGstab(L) residual can be smaller compared to the case of choosing $\eta_k = 0$ like Bi-CGstab(L). In this connection, we can compare the residual norms obtained in the second cycle of GPBi-CGstab(L) and Bi-CGstab(L). The first cycle of GPBi-CGstab(L) is the same as that of Bi-CGstab(L) as displayed in Algorithm 3 (i.e., $\mathrm{H}_L^{gp-stab(L)} = \mathrm{H}_L^{-stab(L)}$), and then we have $\mathrm{H}_L^{gp-stab(L)}\boldsymbol{r}_{2L}^{bicg} = \mathrm{H}_L^{-stab(L)}\boldsymbol{r}_{2L}^{bicg}$ for the intermediate residuals at the second cycle. Therefore, it holds from the property of residual minimization that $\|\boldsymbol{r}_{2L}^{gp-stab(L)}\|_2 \leq \|\boldsymbol{r}_{2L}^{-stab(L)}\|_2$ at the end of the second cycle. Similar situations could be occur if $\mathrm{H}_k^{gp-stab(L)}\boldsymbol{r}_{k+L}^{bicg} \approx \mathrm{H}_k^{-stab(L)}\boldsymbol{r}_{k+L}^{bicg}$ for a larger k.

As a result, GPBi-CGstab(L) has both advantages of Bi-CGstab(L) and GPBi-CG, and this may lead to higher performance than the conventional ones. Note that, we here focused on the differences of the stabilizing polynomials when updating the residuals, because it seems to be difficult to compare $\boldsymbol{r}_k^{-stab(L)}$, $\boldsymbol{r}_k^{gp-}$, and $\boldsymbol{r}_k^{gp-stab(L)}$, directly. We will analyze the theoretical relations between the residuals in the future. In Section 4, we demonstrate by numerical experiments that GPBi-CGstab(L) with $L > 1$ actually achieves better convergence than Bi-CGstab(L) and GPBi-CG.

TABLE 1 Computational costs per MV and memory requirements

Methods	Algorithms	AXPYs	DOTs	Memory
Bi-CGstab(L)	Algorithm 3.1 in Sleijpen and Fokkema[6]	$0.75(L+3)$	$0.25(L+7)$	$2L+4$
	Algorithm 1	$0.5(L+6)$	$0.25(L+7)+0.5/L$	$2L+4$
GPBi-CG	Algorithm 5 in Zhang[8]	7	3.5	11
	Algorithm 2	7.25	4	12
GPBi-CGstab(L)	Algorithm 3	$L+4.25+2/L$	$0.25(L+7)+0.5(L+3)/L$	$4L+8$

Remark 3. We note that, although GPBi-CGstab(L) performs residual minimization with $L+1$ parameters, it should not be confused with Bi-CGstab($L+1$). Because the residuals of GPBi-CGstab(L) and Bi-CGstab($L+1$) are generated in different Krylov subspaces at each cycle, they are viewed as essentially different methods. However, similar to the relations between GPBi-CG, Bi-CGStab2, and Bi-CGstab(2), the residuals of Bi-CGstab($L+1$) can be generated from a variation of GPBi-CGstab(L) in theory. If the parameters $\zeta_{k,j}$ for $j=2,3,\dots,L$ and η_k are set to 0 at odd cycles, the stabilizing polynomials (21) are converted as follows.

$$H_0(\lambda) := 1,$$
$$\begin{cases} H_{k+1}(\lambda) := (1-\zeta_k\lambda)H_k(\lambda), \\ H_{k+L+1}(\lambda) := (1-\zeta_{k,1}\lambda-\cdots-\zeta_{k,L}\lambda^L)H_{k+1}(\lambda)-\eta_k\lambda G_k(\lambda), \end{cases}$$

where $k=0,(L+1),2(L+1),\dots$ and $G_k(\lambda) := (H_k(\lambda)-H_{k+1}(\lambda))/\lambda$. Then, the resulting algorithm generates the Bi-CGstab($L+1$) residuals at even cycles when the parameters are selected to locally minimize the residuals. However, this is just a theoretical aspect and it might be unsuitable as an implementation of Bi-CGstab($L+1$) for systems with a complex spectrum, because the resulting algorithm may have a numerical instability caused by the polynomials of degree one at odd cycles as in Bi-CGStab2*.

3.4 | Computational costs and memory requirements

Table 1 summarizes the computational costs per MV and memory requirements for Bi-CGstab(L), GPBi-CG, and GPBi-CGstab(L). Throughout this paper, "MV" stands for a matrix–vector multiplication with the coefficient matrix A. Note that, Bi-CGstab(L) and GPBi-CGstab(L) require $2L$ MVs per cycle, while GPBi-CG requires 2 MVs per iteration, but, on average, all the methods require 1 MV per Krylov dimension. For Bi-CGstab(L) and GPBi-CG, we show the costs for the original implementations in Sleijpen and Fokkema[6] and Zhang[8] and for our variants (Algorithms 1 and 2). In the table, AXPY and DOT stand for a vector update of the form $a\boldsymbol{x}+\boldsymbol{y}$ and an inner product $\boldsymbol{x}^\top\boldsymbol{y}$, respectively, with $a\in\mathbb{R}$ and $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n$. Following the related studies,[10,13] the form $a\boldsymbol{x}$ or $\boldsymbol{x}+\boldsymbol{y}$ is counted as 0.5 AXPYs in order to compare the theoretical number of arithmetic operations. The costs for checking the stopping rule are not included.

Table 1 indicates that, for a modest value of L (e.g., $L\le 4$), the computational costs for AXPYs and DOTs of GPBi-CGstab(L) are not very high compared to those of Bi-CGstab(L) and GPBi-CG. As mentioned above, there are several formulations to compute the Bi-CG coefficients, and they have different numbers of DOTs. For instance, the numerator of α_k can be converted by $(\tilde{\boldsymbol{r}}_0,\boldsymbol{r}_k)=-\zeta_{k-L,L}(\tilde{\boldsymbol{r}}_0,A^L\boldsymbol{r}_{k-L}^{(L)})$ by using the bi-orthogonality conditions, which reduces $0.5/L$ DOTs in our variants.

The last column of Table 1 displays the amount of n-vectors that have to be stored in the algorithms. Here, the memory for the coefficient matrix ($O(n)$ storage in many cases of sparse matrices) and for the right-hand side vector are not counted. In Algorithms 2 and 3, the vector $\mathbf{v}$ can be stored at the same storage of $\mathbf{r}'$ (or $\mathbf{p}'$), and we count the amount based on such an implementation. In theory, the number of n-vectors can be reduced by one by overwriting $\mathbf{q}_0$ with the same storage of $\mathbf{v}$, but this requires modifications of the algorithms and leads to a slightly complicated implementation.

*The residuals obtained at even iterations of Bi-CGStab2 is the same as those generated by Bi-CGstab(2) in exact arithmetic. From this perspective, Bi-CGStab2 and Bi-CGstab(2) can be viewed as mathematically equivalent methods. However, their numerical behavior is significantly different for systems with a complex spectrum, because Bi-CGStab2 uses stabilizing polynomials of degree one at odd iterations as used in Bi-CGSTAB. Actually, Bi-CGstab(2) is numerically more stable than Bi-CGStab2 in our experiments; see Section 4.1.

4 | NUMERICAL EXPERIMENTS

In this section, we present numerical experiments to demonstrate the effectiveness of the proposed GPBi-CGstab(L) algorithm. Through some model problems, we compare the convergences of CGS, Bi-CGSTAB, Bi-CGStab2, Bi-CGstab(L), GPBi-CG, and GPBi-CGstab(L).

Numerical calculations were conducted by double-precision floating-point arithmetic on a PC (Intel Xeon E3-1270 v5 CPU with 32 GB of RAM) equipped with MATLAB R2018a. Bi-CGstab(L) and GPBi-CGstab(L) were implemented based on Algorithms 1 and 3, respectively, and the others were based on algorithms given by Zhang.[8] The iterations were started with $\mathbf{0}$ and were stopped when the relative residual 2-norm $\|\boldsymbol{r}_k\|_2/\|\boldsymbol{b}\|_2$ became less than 10^{-12} for all the methods. The right-hand side vector $\boldsymbol{b}$ was given as $\boldsymbol{b} = A\boldsymbol{x}^*$ with the exact solution $\boldsymbol{x}^* := (1, 1, \dots, 1)^\top$ for given matrices. The initial shadow residual $\tilde{\boldsymbol{r}}_0$ was set to $\boldsymbol{r}_0$ ($= \boldsymbol{b}$). Different conditions were set for each example below.

Note that Bi-CGstab(L) and GPBi-CGstab(L) can be implemented with different ways which may produce different numerical results. In particular, there are several approaches to perform residual minimization. We have used the backslash command in MATLAB for the normal equation based on (32). The other parts of the methods have been implemented naively based on Algorithms 1 and 3. We will not further discuss the details of implementations in the present paper, but will seek more effective implementations in the future.

4.1 | Numerical example 1

We first show that the proposed method can be a promising alternative to the conventional LTPMs when solving systems with a complex spectrum. To this end, we employ the following two Toeplitz matrices:

$$(\textbf{Toeplitz 1}) \quad A := \begin{bmatrix} 2 & 1 & & \\ 0 & 2 & 1 & \\ 0 & 0 & 2 & \ddots \\ 0 & 0 & 0 & \ddots \\ 1.4 & 0 & 0 & \ddots \\ & 1.4 & 0 & \ddots \\ & & 1.4 & \ddots \\ & & & \ddots \end{bmatrix} \in \mathbb{R}^{500\times 500},$$

$$(\textbf{Toeplitz 2}) \quad A := \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & & & \\ -1 & 1 & 1 & 1 & 1 & 1 & 1 & & \\ & -1 & 1 & 1 & 1 & 1 & 1 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \in \mathbb{R}^{250\times 250}.$$

Toeplitz 1 is motivated by the test matrix in Gutknecht,[5] and the eigenvalues of this matrix are known to be distributed in a radial fashion on the complex plane.[20] The specific spectrum is given in Figure 1. Toeplitz 2 is well-known as Grcar's matrix, and Figure 2 displays its eigenvalues computed by the MATLAB function **eig**. For these types of problems, higher-order stabilizing polynomials are expected to be more effective.

Figures 3 and 4 display the convergence histories of the conventional and proposed LTPMs for Toeplitz 1 and 2, respectively. The plots show the number of MVs on the horizontal axis versus the $\log_{10}$ of the relative residual 2-norm ($\|\boldsymbol{r}_k\|_2/\|\boldsymbol{b}\|_2$) on the vertical axis. Here, the parameter L was set to 2 in Bi-CGstab(L) and GPBi-CGstab(L).

From Figure 3, we can make the following observations for Toeplitz 1. The residual norms of CGS and Bi-CGSTAB diverge and stagnate, respectively, and their iterations cannot proceed owing to breakdown. Bi-CGStab2, which performs second-degree minimizations, converges, although it exhibits large oscillations in the residual norms. Bi-CGstab(2) and GPBi-CG converge slightly faster than Bi-CGStab2; their speed of convergence are roughly the same. Subsequently, the residual norms of the proposed GPBi-CGstab(2) decrease significantly faster than those of the other methods and is the only one to converge within $2n$ MVs (i.e., the theoretical upper bound of MVs required for successful convergence).

For Toeplitz 2, all the methods do not converge within $2n$ MVs, but Bi-CGstab(2) and GPBi-CGstab(2) converge as displayed in Figure 4. Actually, CGS and Bi-CGSTAB have breakdown, and both Bi-CGStab2 and GPBi-CG converge at around 3000 MVs. Thus, together with the above results, Bi-CGstab(2) seems to be numerically more stable than Bi-CGStab2 although they produce the same residuals at even iterations in exact arithmetic, and is also more effective than GPBi-CG for this problem. Nevertheless, GPBi-CGstab(2) (the new method unifying Bi-CGstab(2) and GPBi-CG) achieves much faster convergence than Bi-CGstab(2).

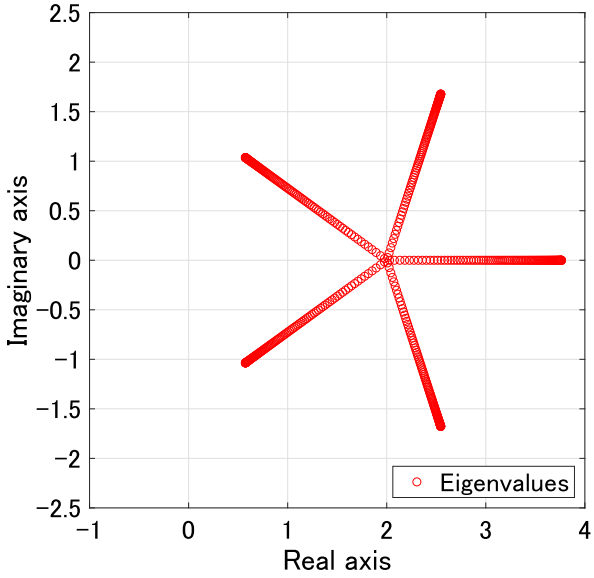


FIGURE 1 Spectrum of Toeplitz 1

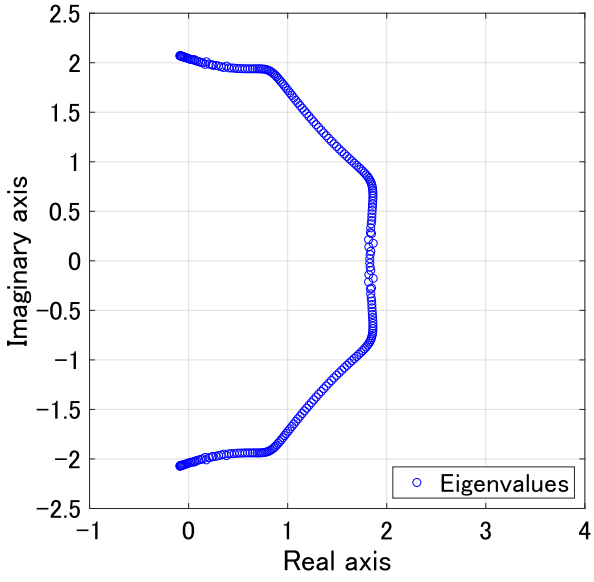


FIGURE 2 Spectrum of Toeplitz 2

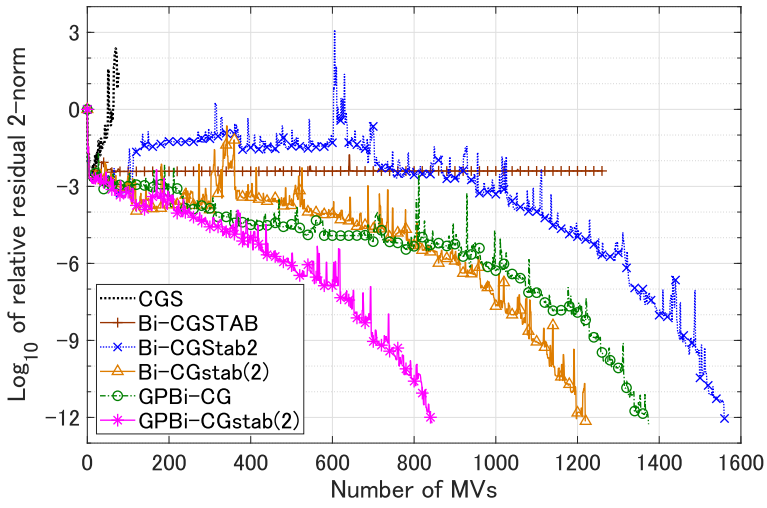


FIGURE 3 Convergence histories of Lanczos-type product methods for Toeplitz 1

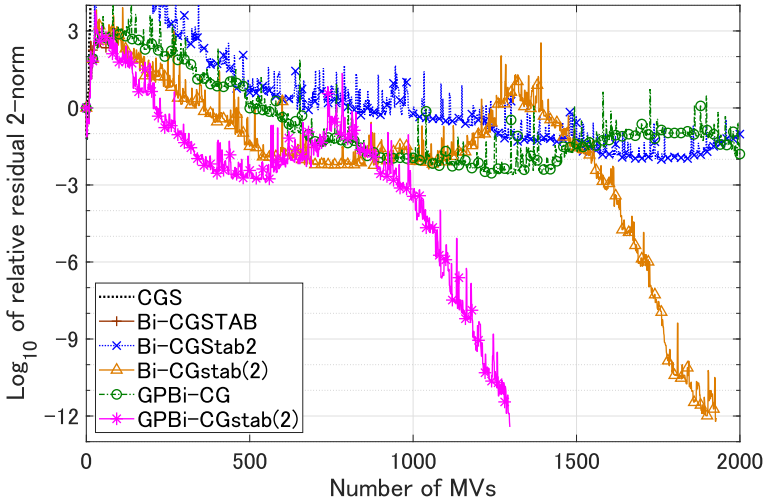


FIGURE 4 Convergence histories of Lanczos-type product methods for Toeplitz 2

TABLE 2 Number of MVs for Bi-CGstab(L) and GPBi-CGstab(L) for the Toeplitz matrices

Matrices	Solvers	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	L=10	Ave.	SD
Toeplitz 1	Bi-CGstab(L)	1,220	810	704	710	720	728	704	720	720	782	158
	GPBi-CGstab(L)	844	750	752	740	732	728	800	702	760	756	40
Toeplitz 2	Bi-CGstab(L)	1,928	1,440	1,088	1,040	972	966	976	1,008	980	1,155	307
	GPBi-CGstab(L)	1,296	1,224	1,056	1,030	1,044	994	992	990	1,040	1,074	103

We now proceed to compare the convergences of Bi-CGstab(L) and GPBi-CGstab(L) for $L \geq 2$. Table 2 shows the number of MVs required for the successful convergence of the two methods with $L = 2, 3, \ldots, 10$ for the Toeplitz matrices. The last two columns display the average (Ave.) and the *SD* of the number of MVs.

From Table 2, we can observe the following. Comparing to the case with $L = 2$, the speed of convergence of Bi-CGstab(L) and GPBi-CGstab(L) can be enhanced by increasing L, but it seems to be limited for modest values of L. GPBi-CGstab(L) has smaller *SD* than Bi-CGstab(L), while the two methods exhibit similar convergence for $L \geq 4$. For small values of L (i.e., $L = 2, 3$), GPBi-CGstab(L) has clearly faster convergence than Bi-CGstab(L) and achieves closer to the optimal number of MVs. This is an advantage of the proposed method, because the computational costs per MV increase with increasing L and it is more efficient to use as small L as possible if we obtain similar convergence behavior.

Remark 4. We briefly remark on the validity of the considerations in Section 3.3. Figure 5 displays the relative residual 2-norms of Bi-CGstab(2), GPBi-CG, GPBi-CGstab(2), and GPBi-CGstab(4) during the early iterations for Toeplitz 1. The plots show the number "k" on the horizontal axis versus the $\log_{10}$ of the relative residual 2-norm ($\|\boldsymbol{r}_k\|_2/\|\boldsymbol{b}\|_2$) on the vertical axis. Bi-CGstab(2), GPBi-CG, GPBi-CGstab(2) generate the same residuals for $k = 2$. (Note that the second residual of GPBi-CG is the same as that of Bi-CGStab2 in theory.) The residuals obtained by GPBi-CGstab(2) are smaller than those obtained by GPBi-CG for $k = 4, 6, \ldots$, and GPBi-CGstab(4) generates even smaller residuals for $k = 4, 8, \ldots$. Thus, the $L + 1$-dimensional minimization in GPBi-CGstab(L) seems to be more effective than performing residual minimization over two-dimensional space L times as in GPBi-CG. To compare Bi-CGstab(2) and GPBi-CGstab(2) in detail, we also show the coefficients of the stabilizing polynomials and the residual norms of the two methods in Table 3. As described in Section 3.3, we have $\|\boldsymbol{r}_4^{gp-stab(L)}\|_2 \leq \|\boldsymbol{r}_4^{-stab(L)}\|_2$ at the second cycle. In this case, however, $(I - \zeta_{2,1}A - \zeta_{2,2}A^2)$ is a good factor to reduce the residual and η_2 is selected to be small (i.e., we have $\boldsymbol{r}_4^{gp-stab(L)} \approx \boldsymbol{r}_4^{-stab(L)}$). Then, at the third cycle, the parameters $\zeta_{4,1}$, $\zeta_{4,2}$, and η_4 are appropriately selected to minimize the residual, and the updated residual of GPBi-CGstab(2) is visibly smaller than that of Bi-CGstab(2).

4.2 | Numerical example 2

Next, we consider various types of nonsymmetric matrices, that have different size, sparsity, condition number, and application discipline, to show the reliability and robustness of the proposed method. Specifically, motivated by several

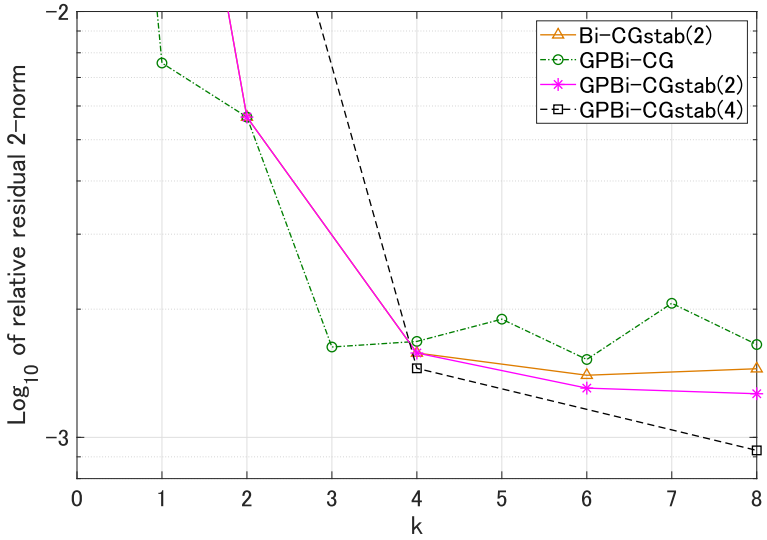


FIGURE 5 Comparison of the residual norms during the early iterations for Toeplitz 1

TABLE 3 Coefficients of the stabilizing polynomials and the residual norms of Bi-CGstab(2) and GPBi-CGstab(2) during the early iterations for Toeplitz 1

	Bi-CGstab(2)	**GPBi-CGstab(2)**
$\|\boldsymbol{r}_2\|_2/\|\boldsymbol{b}\|_2$	0.005649…	0.005649…
$\zeta_{2,1}$	0.409521…	0.409731…
$\zeta_{2,2}$	−0.096541…	−0.097285…
η_2	—	0.002435…
$\|\boldsymbol{r}_4\|_2/\|\boldsymbol{b}\|_2$	0.001578…	0.001577…
$\zeta_{4,1}$	0.300737…	0.437486…
$\zeta_{4,2}$	−0.096728…	−0.139714…
η_4	—	−0.310830…
$\|\boldsymbol{r}_6\|_2/\|\boldsymbol{b}\|_2$	0.001399…	0.001305…

papers (e.g., Sogabe et al.,[21] Tanio and Sugihara,[22] and van der Vorst and Ye[17]), we take test matrices from the SuiteSparse Matrix Collection.[23] Table 4 lists the characteristics of the test matrices; for each matrix, it shows its dimension ($\boldsymbol{n}$), the number of nonzero entries ($\boldsymbol{nnz}$), the 2-norm condition number ($\boldsymbol{\kappa_2(A)}$), and the application discipline. The condition number is displayed only if it is given in the above collection. Here, 14 matrices were employed, but we note that similar numerical results can be obtained for many other matrices.

Table 5 shows the number of MVs required for the successful convergence of CGS, Bi-CGSTAB, Bi-CGStab2, Bi-CGstab(L), GPBi-CG, and GPBi-CGstab(L) for the test matrices. In the table, these methods are expressed as **CGS**, **–STAB**, **–Stab2**, **–stab**(L), **GP–**, and **GP–stab**(L), respectively. For each matrix, the smallest number of MVs is displayed in boldface with an underline, and the second one is displayed only with an underline. We set the maximum number of MVs to $2n$, and the symbol † indicates that there was no convergence within this bound. Table 6 shows the explicitly computed relative residual 2-norm ($\|\boldsymbol{b}-A\boldsymbol{x}_k\|_2/\|\boldsymbol{b}\|_2$) at termination. Here, the value with an underline denotes that $\|\boldsymbol{b}-A\boldsymbol{x}_k\|_2/\|\boldsymbol{b}\|_2 \geq 10^{-9}$ holds despite the fact that the stopping criterion $\|\boldsymbol{r}_k\|_2/\|\boldsymbol{b}\|_2 < 10^{-12}$ is satisfied; that is, there is a large residual gap. The symbol ‡ indicates that the residual norms diverge or that breakdown occurs.

We here comment on the choice of the parameter L for Bi-CGstab(L) and GPBi-CGstab(L). As mentioned in Section 3.3, the residuals of the two methods belong to the same Krylov subspace after performing the same number of cycles. From this perspective, we regard Bi-CGstab(L) and GPBi-CGstab(L) as the homogeneous short-recurrence Krylov subspace methods, and we especially compare the convergences of them. However, in terms of the number of parameters, because GPBi-CGstab(L) performs residual minimization using $L+1$ parameters at each cycle, it might also be fair to compare it with Bi-CGstab($L+1$) in which a residual is minimized over a $L+1$ dimensional space. Therefore, although

TABLE 4 Characteristics of the test matrices

Matrices	n	nnz	$\kappa_2(A)$	Application discipline problem
orsreg_1	2,205	14,133	6.75e+03	Computational fluid dynamics
sherman5	3,312	20,793	1.88e+05	Computational fluid dynamics
poisson3Da	13,514	352,762	1.12e+03	Computational fluid dynamics
poisson3Db	85,623	2,374,949	—	Computational fluid dynamics
add20	2,395	13,151	1.20e+04	Circuit simulation
memplus	17,758	99,147	1.29e+05	Circuit simulation
epb2	25,228	175,027	2.62e+03	Thermal
epb3	84,617	463,625	—	Thermal
wang4	26,068	177,196	4.03e+05	Semiconductor device
big	13,209	91,465	4.44e+07	Directed weighted graph
chipcool0	20,082	281,150	8.33e+06	Model reduction
chipcool1	20,082	281,150	3.55e+06	Model reduction
sme3Da	12,504	874,887	5.22e+07	Structural
sme3Db	29,067	2,081,063	3.63e+07	Structural

TABLE 5 Number of MVs for Lanczos-type product methods for the test matrices

				–stab(L)				GP–stab(L)		
Matrices	**CGS**	**–STAB**	**–Stab2**	$L=2$	$L=3$	$L=4$	**GP–**	$L=2$	$L=3$	$L=4$
orsreg_1	712	2,642	956	620	600	544	780	672	546	**496**
sherman5	3,134	5,938	4,152	4,572	3,804	3,256	4,740	3,720	3,462	**3,088**
poisson3Da	314	296	298	280	288	280	266	**256**	258	**256**
poisson3Db	810	550	586	588	522	584	534	508	504	**496**
add20	**666**	1,324	1,554	1,252	1,290	1,136	918	880	768	864
memplus	2,606	4,698	3,460	3,848	3,426	3,000	2,894	2,588	**2,460**	2,544
epb2	886	926	828	952	792	824	810	784	804	**768**
epb3	†	23,342	19,172	22,504	16,392	16,640	17,712	13,180	**11,658**	13,152
wang4	1,096	1,044	1,140	988	978	1,072	1,018	964	1284	**888**
big	†	8,218	6,242	5,176	4,350	3,656	4,448	3,680	3,660	**3,408**
chipcool0	†	†	†	†	†	†	26,296	21,560	**19,980**	22,056
chipcool1	†	†	†	†	†	30,184	14,852	14,056	**13,716**	13,808
sme3Da	†	†	†	†	†	15,056	†	14,520	**11,934**	12,424
sme3Db	†	16,566	43,800	49,096	29,724	28,184	33,850	16,080	17,142	**13,368**

Abbreviation:CGS, conjugate gradient squared method.

the parameter L is normally set to an even number in Bi-CGstab(L), we set $L = 2, 3, 4$ in the present experiments, in order to also compare the convergences of GPBi-CGstab(L) and Bi-CGstab($L + 1$).

From Tables 5 and 6, we can make the following observations. GPBi-CGstab(L) converges within $2n$ MVs for all the matrices, and there are matrices for which some of the conventional LTPMs do not converge. The number of MVs required for successful convergence of GPBi-CGstab(L) is smaller than that for the conventional LTPMs, for almost all the matrices. In particular, although Bi-CGstab(L) and GPBi-CG appear to be more effective than the other conventional LTPMs, a faster convergence can be achieved by GPBi-CGstab(L). Moreover, as in Bi-CGstab(L), increasing L often

TABLE 6 True relative residual 2-norm of Lanczos-type product methods for the test matrices

				–stab(L)				GP–stab(L)		
Matrices	**CGS**	**–STAB**	**–Stab2**	$L=2$	$L=3$	$L=4$	**GP–**	$L=2$	$L=3$	$L=4$
orsreg_1	2.6e−09	1.0e−11	7.5e−10	4.6e−12	4.5e−12	4.3e−12	5.7e−12	4.3e−12	3.6e−12	3.6e−12
sherman5	7.9e−11	8.3e−13	4.3e−13	8.6e−13	5.9e−13	6.9e−13	8.0e−13	9.2e−13	9.8e−13	6.5e−13
poisson3Da	9.0e−13	6.7e−13	9.1e−13	9.1e−13	5.9e−13	2.3e−13	8.3e−13	9.9e−13	7.4e−13	9.7e−13
poisson3Db	2.5e−09	9.8e−13	9.5e−13	6.8e−13	8.6e−13	7.7e−13	9.6e−13	8.6e−13	9.0e−13	9.1e−13
add20	9.0e−13	6.6e−13	9.9e−13	9.4e−13	5.1e−13	9.4e−13	9.8e−13	5.5e−13	8.7e−13	8.6e−13
memplus	6.6e−11	1.0e−12	4.1e−13	4.8e−13	5.5e−13	6.3e−13	1.0e−12	9.9e−13	9.3e−13	3.8e−12
epb2	2.5e−09	5.6e−13	6.2e−13	3.9e−13	9.4e−13	3.6e−13	9.3e−13	8.8e−13	7.8e−13	9.1e−13
epb3	‡	9.4e−13	9.9e−13	6.5e−13	1.0e−12	9.0e−13	8.7e−13	8.0e−13	9.6e−13	9.2e−13
wang4	1.8e−10	9.8e−13	4.0e−13	7.5e−13	5.9e−13	6.5e−13	5.8e−13	6.3e−13	9.2e−13	7.6e−13
big	‡	7.9e−13	3.7e−05	5.4e−13	4.9e−13	7.6e−13	1.2e−08	9.2e−13	9.8e−13	9.6e−13
chipcool0	‡	2.9e−09	4.3e−08	1.1e−09	2.0e−10	2.8e−10	5.8e−13	8.8e−13	7.8e−13	5.8e−13
chipcool1	‡	8.3e−11	1.9e−10	4.5e−11	7.4e−11	8.8e−13	9.2e−13	1.0e−12	1.0e−12	1.0e−12
sme3Da	‡	9.8e−10	3.6e−07	1.0e−06	3.3e−05	9.8e−11	2.3e−09	5.4e−12	1.5e−12	9.0e−12
sme3Db	‡	9.4e−13	6.7e−11	2.7e−10	8.8e−13	1.5e−12	4.9e−10	3.7e−12	3.2e−12	6.9e−13

Abbreviation:CGS, conjugate gradient squared method.

helps to enhance the convergence in GPBi-CGstab(L). Even when comparing GPBi-CGstab(L) and Bi-CGstab($L+1$), the number of MVs of the proposed method is smaller in many cases. The approximate solutions obtained by GPBi-CGstab(L) are sufficiently accurate for all the matrices, although there are cases where the conventional LTPMs have a large residual gap. We note that, in case there is a large residual gap in the proposed method, the refined techniques presented in, for example, Aihara et al.,[15] Sleijpen and van der Vorst,[16] and van der Vorst and Ye,[17] may be useful to improve it.

We also note that the modest value of L, such as $L = 2, 3$, and 4 above, is a suitable choice for the proposed method in terms of not only computational costs but also numerical stability. In our experience, choosing a larger L (e.g., $L > 10$) often leads to a loss of convergence speed.

4.3 | Numerical example 3

Finally, we present numerical results with preconditioning. We use the preconditioned algorithms, which can be derived by applying the methods to the right-preconditioned system $\tilde{A}\tilde{\boldsymbol{x}} = \boldsymbol{b}$, where $\tilde{A} = AK^{-1}$ and $\tilde{\boldsymbol{x}} = K\boldsymbol{x}$ for a preconditioner K. In CGS, Bi-CGSTAB, Bi-CGStab2, and GPBi-CG, we can update the approximations $\boldsymbol{x}_k$ recursively, instead of $\tilde{\boldsymbol{x}}_k$, through some changes of variables (cf. van der Vorst[4]). Although Bi-CGstab(L) and GPBi-CGstab(L) can also be implemented to update the approximations $\boldsymbol{x}_k$, they require many additional vector updates for obtaining auxiliary vectors such as $K^{-1}(AK^{-1})^j\boldsymbol{r}_k^{(j)}$. Therefore, for the two methods, we use simpler algorithms that update the approximations $\tilde{\boldsymbol{x}}_k$, and $\boldsymbol{x}_k = K^{-1}\tilde{\boldsymbol{x}}_k$ is computed once when the updates are terminated. This approach can be easily implemented with lower computational costs and, in our experience, is slightly less affected by rounding errors than the former approach.

Tables 7 and 8 show the number of MVs and the explicitly computed relative residual 2-norm, respectively, of the methods with preconditioning for the test matrices in Table 4. Here, ILU(0)[24] was employed as the preconditioner, and we refer to the previous subsection for other computational conditions. Note that the same number of multiplications by K^{-1} as the number of MVs are required for each preconditioned algorithm. From Tables 5 to 8, we can observe that the convergence of GPBi-CGstab(L) can be significantly enhanced by preconditioning, as in the conventional LTPMs. Then, the convergence speed and the attainable accuracy of the approximate solutions for the preconditioned GPBi-CGstab(L) compare favorably with those for the conventional LTPMs with preconditioning.

TABLE 7 Number of MVs for Lanczos-type product methods with preconditioning for the test matrices

				–stab(L)				GP–stab(L)		
Matrices	**CGS**	**–STAB**	**–Stab2**	$L = 2$	$L = 3$	$L = 4$	**GP–**	$L = 2$	$L = 3$	$L = 4$
orsreg_1	116	102	102	104	102	104	**98**	100	102	104
sherman5	62	64	**52**	**52**	60	56	54	**52**	60	56
poisson3Da	100	102	90	92	90	**88**	90	**88**	90	**88**
poisson3Db	236	200	208	208	204	208	192	**188**	192	192
add20	332	470	380	368	336	352	368	308	324	**288**
memplus	586	836	686	648	606	576	666	564	558	**512**
epb2	66	64	**60**	**60**	**60**	64	62	**60**	**60**	64
epb3	234	222	218	220	222	224	210	224	222	**208**
wang4	104	102	102	104	102	**96**	104	100	**96**	**96**
big	6,980	2,556	2,230	1,904	1,758	**1,656**	2,472	1,992	1,764	1,752
chipcool0	154	124	118	120	120	112	114	108	108	**104**
chipcool1	106	102	100	100	102	104	**98**	100	102	112
sme3Da	8,616	3,704	4,884	4,172	2,100	2,184	5,298	2,584	2,184	**1,968**
sme3Db	†	†	6,606	2,456	2,742	2,192	4,212	3,324	2,190	**1,888**

Abbreviation:CGS, conjugate gradient squared method.

TABLE 8 True relative residual 2-norm of Lanczos-type product methods with preconditioning for the test matrices

				–stab(L)				GP–stab(L)		
Matrices	**CGS**	**–STAB**	**–Stab2**	$L = 2$	$L = 3$	$L = 4$	**GP–**	$L = 2$	$L = 3$	$L = 4$
orsreg_1	1.5e−09	8.3e−12	9.2e−12	1.0e−11	2.4e−11	3.2e−10	1.2e−11	1.0e−11	2.4e−11	3.2e−10
sherman5	2.1e−14	2.3e−13	8.4e−13	8.4e−13	9.2e−16	1.1e−14	5.0e−13	5.2e−13	7.5e−16	1.0e−14
poisson3Da	2.3e−13	3.0e−13	7.2e−13	4.6e−13	6.7e−13	8.1e−13	6.1e−13	9.7e−13	5.0e−13	7.7e−13
poisson3Db	1.2e−11	9.0e−13	5.6e−13	9.1e−13	4.9e−13	4.3e−13	8.3e−13	9.2e−13	5.5e−13	6.1e−13
add20	7.7e−13	3.8e−13	9.9e−13	8.9e−13	9.3e−13	4.5e−13	8.7e−13	4.0e−13	7.6e−13	9.9e−13
memplus	2.0e−12	7.6e−13	1.2e−12	9.9e−13	8.3e−13	8.5e−13	8.4e−13	8.2e−13	9.6e−13	8.5e−13
epb2	2.2e−13	6.0e−13	8.4e−13	8.4e−13	3.9e−13	1.4e−13	8.1e−13	9.4e−13	5.8e−13	1.8e−13
epb3	4.8e−13	4.4e−13	4.5e−13	5.3e−13	2.7e−13	1.1e−13	9.3e−13	3.6e−13	6.4e−13	9.7e−13
wang4	9.3e−13	3.9e−13	6.5e−13	6.3e−14	1.2e−13	9.0e−13	1.2e−13	2.3e−13	4.9e−13	4.8e−13
big	2.9e−06	4.3e−13	2.2e−12	2.3e−13	8.9e−13	3.5e−13	5.5e−12	1.6e−13	7.0e−13	9.6e−13
chipcool0	1.9e−13	7.9e−13	3.8e−13	1.5e−13	4.9e−13	8.3e−13	7.2e−13	5.9e−13	4.5e−13	8.2e−13
chipcool1	4.6e−13	5.0e−13	8.5e−14	8.4e−14	1.1e−13	2.0e−14	2.9e−13	2.1e−13	4.9e−14	1.0e−14
sme3Da	2.3e−06	4.7e−13	9.1e−10	3.6e−13	3.2e−12	5.4e−11	1.0e−10	4.7e−12	7.0e−12	7.1e−11
sme3Db	‡	‡	3.9e−10	2.1e−13	2.6e−13	9.0e−13	1.4e−12	9.2e−12	1.9e−12	5.0e−13

Abbreviation:CGS, conjugate gradient squared method.

5 | CONCLUDING REMARKS

We have proposed comprehensive stabilizing polynomials to develop a novel framework of LTPMs. The presented polynomials have $L+1$ parameters, from which both the conventional LTPMs, such as CGS, Bi-CGSTAB, and Bi-CGStab2, and their extensions, namely Bi-CGstab(L) and GPBi-CG, can be derived. We have also designed a new method, named GPBi-CGstab(L), in which the $L+1$ parameters are determined to minimize the residual at each cycle. Numerical experiments demonstrated that GPBi-CGstab(L) with $L>1$ can exhibit better convergence properties than the conventional LTPMs for nonsymmetric linear systems. In particular, GPBi-CGstab(L) unifies Bi-CGstab(L) and GPBi-CG, and can be superior to them, especially for systems with a complex spectrum.

Finally, we briefly describe some future prospects. As with conventional LTPMs, there are many possibilities to improve and extend the presented method, and some of them are listed below.

- A refined choice of parameters in stabilizing polynomials, referred to as the strategy of "vanilla", might be incorporated. This may lead to more accurate Bi-CG coefficients and may result in more numerically stable convergence. For details of the strategy, we refer to Sleijpen and van der Vorst[25] and Abe and Sleijpen.[10]
- GPBi-CGstab(L) can also be regarded as a method based on the Sonneveld subspace.[11] Thus, based on a similar analogy of the extension from Bi-CGstab(L) to IDR(s)stab(L)[11] (or equivalently, GBi-CGSTAB(s,L)[22]), the presented method can be extended to be in part of a family of IDR(s)[13] by extending the shadow Krylov subspace $\mathcal{K}_k(A^\top, \tilde{\boldsymbol{r}}_0)$ to its block version $\mathcal{K}_k(A^\top, \tilde{R}_0)$, where $\tilde{R}_0 \in \mathbb{R}^{n\times s}$.
- Block or shifted versions of Bi-CGstab(L) and GPBi-CG have also been developed, and GPBi-CGstab(L) may be naturally extended to this class of methods. For details of the related methods, we refer to Saito et al.[26] and Zhang and Dai[27] (for the block or related global methods) and to Frommer[28] and Dehghan and Mohammadi-Arani[29] (for the shifted methods).

These variations of GPBi-CGstab(L) were not elaborated in the present paper but will be discussed in future studies.

ACKNOWLEDGEMENTS

The author would like to thank the reviewers for their careful reading and constructive comments. This study was supported by grant number JP18K18064 from the Grants-in-Aid for Scientific Research Program (KAKENHI) of the Japan Society for the Promotion of Science (JSPS). This work does not have any conflicts of interest.

ORCID

Kensuke Aihara https://orcid.org/0000-0002-4802-4090

REFERENCES

1. Fletcher R. Conjugate gradient methods for indefinite systems. In: Watson GA, editor. Numerical Analysis, Proceedings of the Dundee Conference on Numerical Analysis. Lecture Notes in Mathematics. Volume 506. Berlin, Germany: Springer, 1976; p. 73–89.
2. Gutknecht MH. Lanczos-type solvers for nonsymmetric linear systems of equations. Acta Numer. 1997;6:271–397.
3. Sonneveld P. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. SIAM J Sci Stat Comput. 1989;10(1):36–52.
4. van der Vorst HA. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J Sci Stat Comput. 1992;13(2):631–644.
5. Gutknecht MH. Variants of BiCGSTAB for matrices with complex spectrum. SIAM J Sci Comput. 1993;14(5):1020–1033.
6. Sleijpen GLG, Fokkema DR. BiCGstab(L) for linear equations involving unsymmetric matrices with complex spectrum. Electron Trans Numer Anal. 1993;1:11–32.
7. Fokkema DR, Sleijpen GLG, van der Vorst HA. Generalized conjugate gradient squared. J Comput Appl Math. 1996;71(1):125–146.
8. Zhang SL. GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems. SIAM J Sci Comput. 1997;18(2):537–551.
9. Abe K, Sleijpen GLG. Hybrid Bi-CG methods with a Bi-CG formulation closer to the IDR approach. Appl Math Comput. 2012;218(22):10889–10899.
10. Abe K, Sleijpen GLG. Solving linear equations with a stabilized GPBiCG method. Appl Numer Math. 2013;67:4–16.
11. Sleijpen GLG, van Gijzen MB. Exploiting BiCGstab(ℓ) strategies to induce dimension reduction. SIAM J Sci Comput. 2010;32(5):2687–2709.
12. Sleijpen GLG, vander Vorst HA, Fokkema DR. BiCGstab(l) and other hybrid Bi-CG methods. Numer Alg. 1994;7(1):75–109.
13. Sonneveld P, van Gijzen MB. IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. SIAM J Sci Comput. 2008;31(2):1035–1062.

14. Aihara K, Abe K, Ishiwata E. A variant of IDRstab with reliable update strategies for solving sparse linear systems. J Comput Appl Math. 2014;259:244–258.
15. Aihara K, Komeyama R, Ishiwata E. Variants of residual smoothing with a small residual gap. Bit Numer Math. 2019;59:565–584.
16. Sleijpen GLG, vander Vorst HA. Reliable updated residuals in hybrid Bi-CG methods. Computing. 1996;56(2):141–163.
17. vander Vorst HA, Ye Q. Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals. SIAM J Sci Comput. 2000;22(3):835–852.
18. Cao ZH. On the QMR approach for iterative methods including coupled three-term recurrences for solving nonsymmetric linear systems. Appl Numer Math. 1998;27(2):123–140.
19. Röllin S, Gutknecht MH. Variations of Zhang's Lanczos-type product method. Appl Numer Math. 2002;41(1):119–133.
20. Fukuda A, Ishiwata E, Iwasaki M, Nakamura Y. The discrete hungry Lotka-Volterra system and a new algorithm for computing matrix eigenvalues. Inverse Prob. 2009;25(1):015007.
21. Sogabe T, Sugihara M, Zhang SL. An extension of the conjugate residual method to nonsymmetric linear systems. J Comput Appl Math. 2009;226(1):103–113.
22. Tanio M, Sugihara M. GBi-CGSTAB(s; L): IDR(s) with higher-order stabilization polynomials. J Comput Appl Math. 2010;235(3):765–784.
23. Davis TA, Hu Y. The university of Florida sparse matrix collection. ACM Trans Math Softw. 2011;38(1):1–25.
24. Saad Y. Iterative Methods for Sparse Linear Systems. 2nd ed. Philadelphia, PA: SIAM, 2003.
25. Sleijpen GLG, vander Vorst HA. Maintaining convergence properties of BiCGstab methods in finite precision arithmetic. Numer Alg. 1995;10(2):203–223.
26. Saito S, Tadano H, Imakura A. Development of the block BiCGSTAB(ℓ) method for solving linear systems with multiple right hand sides. JSIAM Lett. 2014;6:65–68.
27. Zhang J, Dai H. Global GPBiCG method for complex non-Hermitian linear systems with multiple right-hand sides. Comp Appl Math. 2016;35(1):171–185.
28. Frommer A. BiCGStab(ℓ) for families of shifted linear systems. Computing. 2003;70(2):87–109.
29. Dehghan M, Mohammadi-Arani R. Generalized product-type methods based on bi-conjugate gradient (GPBiCG) for solving shifted linear systems. Comp Appl Math. 2017;36(4):1591–1606.

How to cite this article: Aihara K. GPBi-CGstab(L): A Lanczos-type product method unifying Bi-CGstab(L) and GPBi-CG. *Numer Linear Algebra Appl*. 2020;27:e2298. https://doi.org/10.1002/nla.2298