Numerical Algorithms (2023) 93:295–319
https://doi.org/10.1007/s11075-022-01415-7

ORIGINAL PAPER

Global GPBiCGstab(L) method for solving linear matrix equations

Itsuki Horiuchi[1] · Kensuke Aihara[2] · Toshio Suzuki[3] · Emiko Ishiwata[3]

Received: 17 May 2022 / Accepted: 12 September 2022 / Published online: 18 November 2022

Abstract
Global Krylov subspace methods are effective iterative solvers for large linear matrix equations. Several Lanczos-type product methods (LTPMs) for solving standard linear systems of equations have been extended to their global versions. However, the GPBiCGstab(L) method, which unifies two well-known LTPMs (i.e., BiCGstab(L) and GPBiCG methods), has been developed recently, and it has been shown that this novel method has superior convergence when compared to the conventional LTPMs. In the present study, we therefore extend the GPBiCGstab(L) method to its global version. Herein, we present not only a naive extension of the original GPBiCGstab(L) algorithm but also its alternative implementation. This variant enables the preconditioning technique to be applied stably and efficiently. Numerical experiments were performed, and the results demonstrate the effectiveness of the proposed global GPBiCGstab(L) method.

Keywords Linear matrix equation · Global Krylov subspace method · Lanczos-type product method · GPBiCGstab(L) · Preconditioning

Toshio Suzuki and Emiko Ishiwata contributed equally to this work.

✉ Itsuki Horiuchi
Horiuchi.i@jcity.maeda.co.jp

✉ Kensuke Aihara
aiharak@tcu.ac.jp

Toshio Suzuki
tosuzuki@rs.tus.ac.jp

Emiko Ishiwata
ishiwata@rs.tus.ac.jp

1 MAEDA CORPORATION, 2-10-2 Fujimi, Chiyoda-ku, 102-8151, Tokyo, Japan

2 Department of Computer Science, Tokyo City University, 1-28-1 Tamazutsumi, Setagaya-ku, 158-8557, Tokyo, Japan

3 Department of Applied Mathematics, Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, 162-8601, Tokyo, Japan

Mathematics Subject Classification (2010) 65F10 · 65F45

1 Introduction

In many fields of scientific computing, it is important to solve linear matrix equations of the form

$$\mathcal{A}(X) = B, \tag{1}$$

where $\mathcal{A} : \mathbb{R}^{n\times s} \to \mathbb{R}^{n\times s}$ is a linear operator, and $X, B \in \mathbb{R}^{n\times s}$. In the present work, we focus mainly on the case where the representation of $\mathcal{A}$ is a large sparse nonsymmetric matrix and $s \ll n$.

The most basic form of (1) is the standard linear system of equations $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^{n}$. For this problem, short-recurrence Krylov subspace methods, such as the conjugate gradient squared (CGS) method [1], bi-conjugate gradient stabilized (BiCGSTAB) method [2], and BiCGStab2 method [3], are typical iterative solvers that are widely used. The CGS, BiCGSTAB, and BiCGStab2 methods are known to belong to the so-called Lanczos-type product methods (LTPMs) (also called hybrid BiCG methods), wherein the residuals are defined by the product of the stabilizing polynomials and BiCG residuals. The BiCGstab(L) method [4] and GPBiCG method [5] also belong to LTPMs and can be viewed as two different generalizations of the abovementioned typical methods. Moreover, the GPBiCGstab(L) method [6], which unifies BiCGstab(L) and GPBiCG, has been developed recently. Numerical experiments have been performed to demonstrate that the GPBiCGstab(L) method has superior convergence when compared to the conventional LTPMs, especially for linear systems with complex spectra.

Because $\mathcal{A}$ is defined on the finite-dimensional space, there always exists a standard linear system of equations that is mathematically equivalent to (1), to which LTPMs can be applied theoretically. However, it is often difficult to construct the representation matrix of $\mathcal{A}$, i.e., the coefficient matrix of the converted linear system. For example, it is well known that the Sylvester equation

$$\mathcal{A}(X) = AX - XC = B, \quad A \in \mathbb{R}^{n\times n}, \quad C \in \mathbb{R}^{s\times s} \tag{2}$$

can be converted to a standard linear system $\tilde{A}\boldsymbol{x} = \tilde{\boldsymbol{b}}$ with $\tilde{A} := I_s \otimes A - C^\top \otimes I_n \in \mathbb{R}^{ns\times ns}$ and $\tilde{\boldsymbol{b}} := \mathbf{vec}(B) \in \mathbb{R}^{ns}$, but it is costly to construct $\tilde{A}$ explicitly when ns is large. Here, $\otimes$ denotes the Kronecker product, I_k is the identity matrix of order k, and $\mathbf{vec} : \mathbb{R}^{n\times s} \to \mathbb{R}^{ns}$ is the vectorization operator, i.e., $\mathbf{vec}(V) = [\boldsymbol{v}_1^\top, \boldsymbol{v}_2^\top, \dots, \boldsymbol{v}_s^\top]^\top \in \mathbb{R}^{ns}$ for $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_s] \in \mathbb{R}^{n\times s}$. More generally, there are also cases where the representation matrix of $\mathcal{A}$ cannot be expressed explicitly. Therefore, application of iterative solvers to the matrix equations without explicitly using the representation matrix is desirable. The global Krylov subspace methods (cf., e.g., [7–12]) are known to be such matrix-free approaches and generate approximate solutions using the matrix Krylov subspace $\mathcal{K}_k(\mathcal{A}, R_0) := \left\{\sum_{i=0}^{k-1} c_i \mathcal{A}^i(R_0) \mid c_i \in \mathbb{R}\right\}$, where $R_0 := B - \mathcal{A}(X_0)$ is the initial residual with an initial guess X_0, and $\mathcal{A}^i$ indicates that $\mathcal{A}$ acts i times. Thus, the global methods can be implemented without using the representation matrix if the linear transformation $\mathcal{A}(V)$ is available for an arbitrary $V \in \mathbb{R}^{n\times s}$.

Several LTPMs have been extended thus far to their global versions. For example, the Gl-CGS-type methods [13, 14], Gl-BiCGSTAB method [9], and Gl-GPBiCG method [15], have been proposed, where "Gl-" represents the global version of the method. Moreover, relating to the methods based on the global Lanczos process [9], the global quasi-minimal residual (Gl-QMR) method [16] and the global bi-conjugate residual (Gl-BiCR) type methods [17] have also been developed. However, to the best of our knowledge, the global versions of the classical BiCGstab(L) and recent GPBiCGstab(L) methods have not been studied previously. We therefore develop a novel global GPBiCGstab(L) method (including Gl-BiCGstab(L)) for further convergence improvement. Then, based on the aforementioned works, we restrict the target matrix equation to a large sparse nonsymmetric linear system with multiple right-hand sides $AX = B$ and discuss the preconditioned algorithms of Gl-GPBiCGstab(L). It is comparatively easy to extend the original GPBiCGstab(L) algorithm [6, Algorithm 3] to its global version naively. However, when applying the so-called right preconditioning to this naive Gl-GPBiCGstab(L), several concerns regarding numerical stability and computational costs are noted. To overcome these problems, we also derive a refined variant of Gl-GPBiCGstab(L), which is mathematically equivalent to the naive version but uses alternative recursion formulas to update the iteration matrices. This refined variant enables right preconditioning to be applied with greater robustness and fewer computational costs. Through numerical experiments on model problems (i.e., linear system $AX = B$ and the Sylvester equation $AX - XC = B$), we show that the refined Gl-GPBiCGstab(L) (with or without preconditioning) is more effective than other typical Gl-LTPMs.

The remainder of this paper is organized as follows. In Section 2, we describe a naive Gl-GPBiCGstab(L) algorithm for (1) by extending the original GPBiCGstab(L). In Section 3, we derive a refined variant of Gl-GPBiCGstab(L) by partly using different recursion formulas. In Section 4, we present the preconditioned algorithms of the naive and refined Gl-GPBiCGstab(L) for the representative matrix equation $AX = B$ as well as note their advantages and disadvantages. In Section 5, numerical experiments are presented to demonstrate that the proposed methods have superior convergence when compared to the conventional ones. Finally, some concluding remarks and a note on the future directions are given in Section 6.

2 Global GPBiCGstab(L) algorithm

This section presents the extension of the GPBiCGstab(L) algorithm proposed in [6] to its global version for solving linear matrix equations of the form (1).

2.1 Extension to global algorithms

We first describe a generic method for constructing global algorithms. When an algorithm of the LTPMs for the standard linear system of equations is given, its global version can be derived directly through the following simple steps (see also [9, 10]):

1. Applying the given LTPM algorithm to $\tilde{A}\boldsymbol{x} = \tilde{\boldsymbol{b}}$ that is equivalent to (1), where $\tilde{A} \in \mathbb{R}^{ns \times ns}$ is a representation matrix of $\mathcal{A}$ and $\tilde{\boldsymbol{b}} := \mathbf{vec}(B)$.
2. Rewriting the operations in the algorithm using the following equivalent reformulations. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{ns}$ and $a \in \mathbb{R}$,

$$\text{linear transformation: } \tilde{A}\boldsymbol{x} \longrightarrow \mathcal{A}(X), \tag{3}$$

$$\text{vector update (AXPY): } a\boldsymbol{x} + \boldsymbol{y} \longrightarrow aX + Y, \tag{4}$$

$$\text{inner product (DOT): } (\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{x}^\top \boldsymbol{y} \longrightarrow \langle X, Y \rangle_F := \mathrm{tr}(X^\top Y), \tag{5}$$

   where $X, Y \in \mathbb{R}^{n \times s}$ are the matrices satisfying $\boldsymbol{x} = \mathbf{vec}(X)$ and $\boldsymbol{y} = \mathbf{vec}(Y)$. Note that the 2-norm $\|\boldsymbol{x}\|_2$ is also converted to the associated Frobenius norm (F-norm) $\|X\|_F := \sqrt{\langle X, X \rangle_F}$.

From (3), the resulting global algorithm can be implemented without using the representation matrix $\tilde{A}$ explicitly. This is the main advantage of the global methods in actual computations. Moreover, when the transformation by $\mathcal{A}$ is based mainly on matrix–matrix products, we can also implement it with high-performance level-3 BLAS routines.

2.2 Simple derivation

Herein, we describe an algorithm of the Gl-GPBiCGstab(L) method obtained by the simple transformations noted above. For details regarding the GPBiCGstab(L) method itself, we refer to [6] and the next section.

By applying the original GPBiCGstab(L) algorithm [6, Algorithm 3] to $\tilde{A}\boldsymbol{x} = \tilde{\boldsymbol{b}}$ and rewriting the iteration process with (3)–(4), the naive extension to the Gl-GPBiCGstab(L) algorithm is obtained as shown in Algorithm 1. The notations in the algorithm are based on MATLAB conventions. For example, the variables R and P consist of $n \times s$ submatrices R_i and P_i for $i = 0, 1, \ldots, j$, respectively; that is, $\mathrm{R} = [\mathrm{R}_0; \mathrm{R}_1; \ldots; \mathrm{R}_j]$ and $\mathrm{P} = [\mathrm{P}_0; \mathrm{P}_1; \ldots; \mathrm{P}_j]$, respectively, where $[\mathrm{V}_0; \mathrm{V}_1; \ldots; \mathrm{V}_j] := [\mathrm{V}_0^\top, \mathrm{V}_1^\top, \ldots, \mathrm{V}_j^\top]^\top$, and the submatrix V_i theoretically corresponds to $\mathcal{A}^i(\mathrm{V}_0)$.

Remark 1 When using MATLAB, there are several approaches for storing the variables comprising $n \times s$ submatrices, such as R and P above. One method is to construct $\mathrm{R} = [\mathrm{R}_0; \mathrm{R}_1; \ldots; \mathrm{R}_j]$ directly as an $n(j+1) \times s$ long rectangular matrix. Another approach is to use a multidimensional array. Because the matrix R can be viewed as a three-dimensional tensor, it can be stored in an $n \times s \times (j+1)$ three-dimensional array. In this case, the ith submatrix R_i is accessed as $\mathrm{R}(:, :, i)$ in MATLAB. Alternatively, we can use a structure array. For example, the matrix R can be defined using the MATLAB command 'struct' as follows:

$$\mathrm{R\_struct} = \mathrm{struct}(\text{`R'}, \{\mathrm{R}_0, \mathrm{R}_1, \ldots, \mathrm{R}_j\}),$$

where the first argument 'R' is the field name and second argument is a cell array containing the submatrices. In this case, the ith submatrix R_i is accessed as $\mathrm{R\_struct}(i).\mathrm{R}$. Because the implementation using the structure array is faster than the other approaches in our experience, we use this in the actual computations.

Line 19 of Algorithm 1 requires solving a minimization problem

$$\min_{\zeta_i,\eta}\left\|\mathrm{R}_0-\sum_{i=1}^{L}\zeta_i\mathrm{R}_i-\eta\mathrm{Y}\right\|_F$$

for $L+1$ scalar variables $\zeta_1,\dots,\zeta_L$ and η, where $\mathrm{R}_i\in\mathbb{R}^{n\times s}$ for $i=0,1,\dots,L$ and $\mathrm{Y}\in\mathbb{R}^{n\times s}$ are given in the iteration process. Note that the integer L is a predetermined constant in Gl-GPBiCGstab(L). The above problem is equivalent to the standard least-squares problem

$$\min_{\boldsymbol{\zeta},\eta}\left\|\mathbf{vec}(\mathrm{R}_0)-M\begin{bmatrix}\boldsymbol{\zeta}\\ \eta\end{bmatrix}\right\|_2,$$
$$M:=[\mathbf{vec}(\mathrm{R}_1),\dots,\mathbf{vec}(\mathrm{R}_L),\mathbf{vec}(\mathrm{Y})],\quad \boldsymbol{\zeta}:=[\zeta_1,\dots,\zeta_L]^\top, \tag{6}$$

and can be converted to the normal equation

$$M^\top M\begin{bmatrix}\boldsymbol{\zeta}\\ \eta\end{bmatrix}=M^\top\mathbf{vec}(\mathrm{R}_0). \tag{7}$$

As is well known, we can use a direct method such as Cholesky factorization to solve (7), and a more stable approach such as QR factorization is useful for solving the least-squares problem (6) directly when M is ill-conditioned.

Note that the first iteration of GPBiCGstab(L) corresponds to that of BiCGstab(L) and that η needs to be 0. Hence, the first iteration (i.e., one BiCGstab(L) iteration)

```
1: Select an initial guess $\mathrm{X}$.
2: Compute $\mathrm{R}_0=\mathrm{B}-\mathcal{A}(\mathrm{X})$ and choose $\tilde{\mathrm{R}}_0$.
3: Set $\mathrm{R}=[\mathrm{R}_0]$, $\mathrm{P}=[\mathrm{R}_0]$, and $\mathrm{Q}=[\mathrm{O}_{n\times s}]$.
4: $\mathrm{Z}=\mathrm{U}=\mathrm{Y}=\mathrm{O}_{n\times s}$
5: **for** $k=0,1,\dots,$ until convergence **do**
6:     $\rho=\langle\tilde{\mathrm{R}}_0,\mathrm{R}_0\rangle_F$
7:     **for** $j=1,2,\dots,L$ **do**
8:         **if** $k\neq 0$ && $j>1$ **then**
9:             $\mathrm{S}=[\mathrm{S}_0;\dots;\mathrm{S}_{L-j}]-\alpha[\mathrm{Q}_1;\dots;\mathrm{Q}_{L-j+1}]$
10:            $\mathrm{Q}=\mathrm{S}-\beta[\mathrm{Q}_0;\dots;\mathrm{Q}_{L-j}]$
11:        **end if**
12:        $\mathrm{P}=[\mathrm{P};\mathcal{A}(\mathrm{P}_{j-1})]$,   $\mathrm{V}=\mathrm{Q}_0-\mathrm{P}_1$
13:        $\sigma=\langle\tilde{\mathrm{R}}_0,\mathrm{P}_j\rangle_F$,   $\alpha=\rho/\sigma$
14:        $\mathrm{X}=\mathrm{X}+\alpha\mathrm{P}_0$,   $\mathrm{Z}=\mathrm{Z}-\alpha\mathrm{U}$,   $\mathrm{Y}=\mathrm{Y}-\alpha\mathrm{V}$
15:        $\mathrm{R}=\mathrm{R}-\alpha[\mathrm{P}_1;\dots;\mathrm{P}_j]$,   $\mathrm{R}=[\mathrm{R};\mathcal{A}(\mathrm{R}_{j-1})]$
16:        $\rho=\langle\tilde{\mathrm{R}}_0,\mathrm{R}_j\rangle_F$,   $\beta=\rho/\sigma$,   $\mathrm{P}=\mathrm{R}-\beta\mathrm{P}$,   $\mathrm{U}=\mathrm{Y}-\beta\mathrm{U}$
17:    **end for**
18:    Set $\mathrm{S}=[\mathrm{R}_1;\dots;\mathrm{R}_{L-1}]$, $\mathrm{Q}=[\mathrm{P}_1;\dots;\mathrm{P}_L]$, $\mathrm{R}'=\mathrm{R}_0$, and $\mathrm{P}'=\mathrm{P}_0$.
19:    Solve $\min_{\zeta_i,\eta}\left\|\mathrm{R}_0-\sum_{i=1}^{L}\zeta_i\mathrm{R}_i-\eta\mathrm{Y}\right\|_F$   (**if** $k=0$ **then** $\eta=0$ **end if**).
20:    $\mathrm{Z}=\zeta_1\mathrm{R}_0+\cdots+\zeta_L\mathrm{R}_{L-1}+\eta\mathrm{Z}$,   $\mathrm{X}=\mathrm{X}+\mathrm{Z}$
21:    $\mathrm{R}=\mathrm{R}_0-\zeta_1\mathrm{R}_1-\cdots-\zeta_L\mathrm{R}_L-\eta\mathrm{Y}$,   $\mathrm{Y}=\mathrm{R}'-\mathrm{R}$
22:    $\mathrm{P}=\mathrm{P}_0-\zeta_1\mathrm{P}_1-\cdots-\zeta_L\mathrm{P}_L-\eta\mathrm{U}$,   $\mathrm{U}=\mathrm{P}'-\mathrm{P}$
23: **end for**
```

Algorithm 1 Gl-GPBiCGstab(L) for (1) (naive)

is displayed separately from the main GPBiCGstab(L) iterations in the original algorithm [6, Algorithm 3], whereas we display all iterations in the same form using a branch at line 19 in Algorithm 1.

3 Alternative implementation of Gl-GPBiCGstab(L)

In this section, we present the derivation of another Gl-GPBiCGstab(L) algorithm that is mathematically equivalent to Algorithm 1 but exploits alternative recursion formulas for updating the iteration matrices. The resulting algorithm uses a slightly simpler expression and is also useful for designing a robust right preconditioned Gl-GPBiCGstab(L) with fewer additional costs, as shown in the next section.

3.1 Basic concepts

Similar to standard LTPMs, the global versions of the LTPMs can be characterized by generating the residuals R_k defined by the form

$$R_k := H_k(\mathcal{A}) \left[R_k^{gbcg} \right], \tag{8}$$

where $H_k(\lambda)$ is the so-called stabilizing polynomial of degree k satisfying $H_k(0) = 1$, and R_k^{gbcg} is the kth Gl-BiCG residual that is generated by the coupled two-term recurrences

$$R_{k+1}^{gbcg} = R_k^{gbcg} - \alpha_k \mathcal{A}\left(P_k^{gbcg}\right), \tag{9}$$

$$P_{k+1}^{gbcg} = R_{k+1}^{gbcg} - \beta_k P_k^{gbcg} \tag{10}$$

with recurrence coefficients $\alpha_k, \beta_k \in \mathbb{R}$ and direction matrix $P_k^{gbcg} \in \mathbb{R}^{n\times s}$ of Gl-BiCG [9]. Note that $H_k(\mathcal{A})$ is a linear operator of the polynomial form, i.e., $H_k(\mathcal{A}) = \mathcal{I} - \omega_1\mathcal{A} - \cdots - \omega_k\mathcal{A}^k$ with $\omega_j \in \mathbb{R}$ and an identity operator $\mathcal{I}$, and we define its operation for $X \in \mathbb{R}^{n\times s}$ as

$$H_k(\mathcal{A})[X] := X - \omega_1\mathcal{A}(X) - \cdots - \omega_k\mathcal{A}^k(X).$$

Specific choices of the stabilizing polynomials give specific Gl-LTPMs.

The original GPBiCGstab(L) [6] is a novel framework of the standard LTPMs using the following comprehensive stabilizing polynomials.

$$H_{k+L}(\lambda) := \left(1 - \zeta_{k,1}\lambda - \cdots - \zeta_{k,L}\lambda^L\right) H_k(\lambda) - \eta_k\lambda G_{k-1}(\lambda), \tag{11}$$

$$G_{k-1}(\lambda) := \frac{H_{k-L}(\lambda) - H_k(\lambda)}{\lambda}, \tag{12}$$

where $H_0(\lambda) := 1$, $\eta_0 := 0$, and k is a multiple of L. The $L + 1$ coefficients $\zeta_{k,1}, \ldots, \zeta_{k,L}, \eta_k \in \mathbb{R}$ are independent parameters. The stabilizing polynomials are reduced to those of BiCGstab(L) if η_k is always set to 0 and can be equivalent to those of GPBiCG when $L = 1$. Thus, we can also derive CGS, BiCGSTAB, and BiCGStab2 from the above framework. In the practical computation of GPBiCGstab(L), the $L+1$ parameters are determined by locally minimizing the residual norms in each iteration.

In the following subsections, we show derivation of the Gl-GPBiCGstab(L) algorithm that generates residuals (8) with stabilizing polynomials (11) and (12). The derivation is somewhat different from the one in [6].

3.2 Recursion formulas for updating the iteration matrices

Herein, the operators $H_k(\mathcal{A})$ and $G_{k-1}(\mathcal{A})$ are denoted as H_k and G_{k-1}, respectively, and we omit the parentheses for simplicity; specifically, $H_k(\mathcal{A})[X]$, $G_{k-1}(\mathcal{A})[X]$, and $\mathcal{A}(X)$ are expressed as $\mathrm{H}_k X$, $\mathrm{G}_{k-1}X$, and $\mathcal{A}X$, respectively.

Let X_k, $R_k = \mathrm{H}_k R_k^{gbcg}$, and $P_k = \mathrm{H}_k P_k^{gbcg}$ be the approximation, residual, and direction matrix of Gl-GPBiCGstab(L), respectively. These matrices are updated to X_{k+L}, $R_{k+L} = \mathrm{H}_{k+L} R_{k+L}^{gbcg}$, and $P_{k+L} = \mathrm{H}_{k+L} P_{k+L}^{gbcg}$, respectively, and the processes are counted as one cycle below. We now introduce the following auxiliary matrices for $j = 1, 2, \dots, L$.

$$R_k^{(j)} := \mathrm{H}_k R_{k+j}^{gbcg}, \quad P_k^{(j)} := \mathrm{H}_k P_{k+j}^{gbcg}, \tag{13}$$

$$Y_k^{(j)} := \mathcal{A}\mathrm{G}_{k-1} R_{k+j}^{gbcg}, \quad U_k^{(j)} := \mathcal{A}\mathrm{G}_{k-1} P_{k+j}^{gbcg}, \quad Z_k^{(j)} := \mathrm{G}_{k-1} R_{k+j}^{gbcg}. \tag{14}$$

From (11), (13), and (14), the updated residual and direction matrices can be expanded as

$$R_{k+L} = R_k^{(L)} - \zeta_{k,1}\mathcal{A}R_k^{(L)} - \cdots - \zeta_{k,L}\mathcal{A}^L R_k^{(L)} - \eta_k Y_k^{(L)}, \tag{15}$$

$$P_{k+L} = P_k^{(L)} - \zeta_{k,1}\mathcal{A}P_k^{(L)} - \cdots - \zeta_{k,L}\mathcal{A}^L P_k^{(L)} - \eta_k U_k^{(L)}, \tag{16}$$

and the associated approximation can be expressed as

$$X_{k+L} = X_k^{(L)} + \zeta_{k,1} R_k^{(L)} + \cdots + \zeta_{k,L}\mathcal{A}^{L-1} R_k^{(L)} + \eta_k Z_k^{(L)}, \tag{17}$$

where we note that $Y_k^{(L)} = \mathcal{A}Z_k^{(L)}$ holds. When the auxiliary matrices in (15)–(17) are obtained, we can determine the $L+1$ parameters $\zeta_{k,1}, \dots, \zeta_{k,L}, \eta_k$ to minimize the residual norm $\|R_{k+L}\|_F$, as described in Section 2.2.

As mentioned in [6], the stabilizing polynomial $H_{k+L}(\lambda)$ consists of two parts: multiplication of an Lth degree polynomial with the previous $H_k(\lambda)$ and relaxation term $-\eta_k \lambda G_{k-1}(\lambda)$. We therefore describe the generation of auxiliary matrices for each part separately.

3.2.1 Multiplication of the Lth degree polynomial

We describe an iteration process to generate the auxiliary matrices $\mathcal{A}^i R_k^{(L)}$ and $\mathcal{A}^i P_k^{(L)}$ for $i = 0, 1, \dots, L$, and the associated approximation $X_k^{(L)}$ in (15)–(17).

This process corresponds to the well-known L-times-BiCG steps in BiCGstab(L), and the description below is based on that in [6, Section 2.2].

Assuming that the approximation $X_k^{(0)} := X_k$, the corresponding residual $R_k^{(0)} := R_k$, and direction matrix $P_k^{(0)} := P_k$ are given at the beginning of the cycle. Then, the jth repetition ($j = 1, 2, \ldots, L$) of the BiCG steps is described as follows.

From the Gl-BiCG recursions (9) and (10), applying operators $\mathcal{A}^i \mathrm{H}_k$ to R_{k+j}^{gbcg} for $i = 0, 1, \ldots, j-1$ and to P_{k+j}^{gbcg} for $i = 0, 1, \ldots, j$, we have

$$\begin{aligned}
\mathcal{A}^i \mathrm{H}_k R_{k+j}^{gbcg} &= \mathcal{A}^i \mathrm{H}_k R_{k+j-1}^{gbcg} - \alpha_{k+j-1} \mathcal{A}^{i+1} \mathrm{H}_k P_{k+j-1}^{gbcg}, \quad i = 0, 1, \ldots, j-1,\\
\mathcal{A}^i \mathrm{H}_k P_{k+j}^{gbcg} &= \mathcal{A}^i \mathrm{H}_k R_{k+j}^{gbcg} - \beta_{k+j-1} \mathcal{A}^i \mathrm{H}_k P_{k+j-1}^{gbcg}, \quad i = 0, 1, \ldots, j,
\end{aligned}$$

where it is noted that $\mathrm{H}_k \mathcal{A} V = \mathcal{A} \mathrm{H}_k V$ holds for an arbitrary $V \in \mathbb{R}^{n \times s}$. Rewriting the above recursions with the auxiliary matrices (13) gives

$$\begin{aligned}
\mathcal{A}^i R_k^{(j)} &= \mathcal{A}^i R_k^{(j-1)} - \alpha_k^{(j-1)} \mathcal{A}^{i+1} P_k^{(j-1)}, \quad i = 0, 1, \ldots, j-1, && (18)\\
\mathcal{A}^i P_k^{(j)} &= \mathcal{A}^i R_k^{(j)} - \beta_k^{(j-1)} \mathcal{A}^i P_k^{(j-1)}, \quad i = 0, 1, \ldots, j, && (19)
\end{aligned}$$

where $\alpha_k^{(j)} := \alpha_{k+j}$ and $\beta_k^{(j)} := \beta_{k+j}$. The approximation $X_k^{(j)}$ associated with $R_k^{(j)}$ can be expressed as

$$X_k^{(j)} = X_k^{(j-1)} + \alpha_k^{(j-1)} P_k^{(j-1)}. \tag{20}$$

Following [6, Eqs. (14) and (15)] and using the iteration matrices, we compute the Gl-BiCG coefficients $\alpha_k^{(j)}$ and $\beta_k^{(j)}$ as follows:

$$\begin{gathered}
\alpha_k^{(j)} = \frac{\rho_k^{(j)}}{\sigma_k^{(j)}}, \quad \beta_k^{(j)} = \frac{\rho_k^{(j+1)}}{\sigma_k^{(j)}},\\
\rho_k^{(j)} := \langle \tilde{R}_0, \mathcal{A}^j R_k^{(j)} \rangle, \quad \sigma_k^{(j)} := \langle \tilde{R}_0, \mathcal{A}^{j+1} P_k^{(j)} \rangle,
\end{gathered} \tag{21}$$

where $\tilde{R}_0$ is the initial shadow residual.

Thus, when $X_k^{(j-1)}$, $\mathcal{A}^i R_k^{(j-1)}$, and $\mathcal{A}^i P_k^{(j-1)}$ for $i = 0, 1, \ldots, j-1$ are given at the beginning of the jth repetition; the new matrices $X_k^{(j)}$, $\mathcal{A}^i R_k^{(j)}$, and $\mathcal{A}^i P_k^{(j)}$ for $i = 0, 1, \ldots, j$ can be generated using (18)–(21). This updating scheme is summarized as follows:

$$\begin{tikzcd}
P_k^{(j-1)} & X_k^{(j-1)} \arrow[r, "\text{Eq.(20)}"] & X_k^{(j)} & & P_k^{(j)} \\
 & & & \overset{\text{Eq.(19)}}{\nearrow}\ {\scriptstyle i=0} & \\
\mathcal{A}P_k^{(j-1)} & R_k^{(j-1)} \arrow[r, "\text{Eq.(18)}", "i=0"'] & R_k^{(j)} & & \mathcal{A}P_k^{(j)} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\mathcal{A}^{j-1}P_k^{(j-1)} \arrow[dd, "\text{Apply}\mathcal{A}"] & \mathcal{A}^{j-2}R_k^{(j-1)} \arrow[r, "\text{Eq.(18)}", "i=j-2"'] & \mathcal{A}^{j-2}R_k^{(j)} & & \mathcal{A}^{j-1}P_k^{(j)} \\
 & & & \overset{\text{Eq.(19)}}{\nearrow}\ {\scriptstyle i=j-1} & \\
\mathcal{A}^{j}P_k^{(j-1)} & \mathcal{A}^{j-1}R_k^{(j-1)} \arrow[r, "\text{Eq.(18)}", "i=j-1"'] & \mathcal{A}^{j-1}R_k^{(j)} \arrow[dd, "\text{Apply}\mathcal{A}"] & & \mathcal{A}^{j}P_k^{(j)} \\
 & & & \overset{\text{Eq.(19)}}{\nearrow}\ {\scriptstyle i=j} & \\
 & & \mathcal{A}^{j}R_k^{(j)} & &
\end{tikzcd} \tag{22}$$

Note that $\mathcal{A}^j P_k^{(j-1)}$ and $\mathcal{A}^j R_k^{(j)}$ are obtained by explicitly applying $\mathcal{A}$ to $\mathcal{A}^{j-1} P_k^{(j-1)}$ and $\mathcal{A}^{j-1} R_k^{(j)}$, respectively. By repeating scheme (22) for $j = 1, 2, \ldots, L$, we obtain the auxiliary matrices $X_k^{(L)}$, $\mathcal{A}^i R_k^{(L)}$, and $\mathcal{A}^i P_k^{(L)}$ for $i = 0, 1, \ldots, L$. This process is identical to that used in Algorithm 1.

3.2.2 Relaxation

Next, we describe the iterations to generate the auxiliary matrices $Y_k^{(L)}$, $U_k^{(L)}$, and $Z_k^{(L)}$ for (15)–(17). This process corresponds to [6, Section 3.1], but we exploit different recursion formulas, and the resulting algorithm will thus be different from Algorithm 1.

For now, we consider the generation of $Y_k^{(L)} = \mathcal{A}\mathrm{G}_{k-1} R_{k+L}^{gbcg}$ and $U_k^{(L)} = \mathcal{A}\mathrm{G}_{k-1} P_{k+L}^{gbcg}$. We introduce additional auxiliary matrices

$$S_k^{(j)} := \mathrm{H}_{k-L} R_{k+j}^{gbcg}, \quad Q_k^{(j)} := \mathrm{H}_{k-L} P_{k+j}^{gbcg}$$

for $j = 0, 1, \ldots, L$, where $S_k^{(0)} = R_{k-L}^{(L)}$ and $Q_k^{(0)} = P_{k-L}^{(L)}$ hold. Using the relation $\mathcal{A}\mathrm{G}_{k-1} = \mathrm{H}_{k-L} - \mathrm{H}_k$, we have

$$\begin{aligned}
\mathcal{A}\mathrm{G}_{k-1} R_{k+L}^{gbcg} &= \mathrm{H}_{k-L} R_{k+L}^{gbcg} - \mathrm{H}_k R_{k+L}^{gbcg}, \\
\mathcal{A}\mathrm{G}_{k-1} P_{k+L}^{gbcg} &= \mathrm{H}_{k-L} P_{k+L}^{gbcg} - \mathrm{H}_k P_{k+L}^{gbcg},
\end{aligned}$$

which can be rewritten as follows:

$$Y_k^{(L)} = S_k^{(L)} - R_k^{(L)}, \tag{23}$$

$$U_k^{(L)} = Q_k^{(L)} - P_k^{(L)}. \tag{24}$$

Because $R_k^{(L)}$ and $P_k^{(L)}$ are generated by L repetitions of (22) with $i = 0$, we consider the method of obtaining $S_k^{(L)}$ and $Q_k^{(L)}$. At the jth repetition, applying $\mathcal{A}^i \mathrm{H}_{k-L}$ for

$i = 0, 1, \dots, L - j$ to the Gl-BiCG recursions, we have

$$\begin{aligned}\mathcal{A}^i \mathrm{H}_{k-L} R_{k+j}^{gbcg} &= \mathcal{A}^i \mathrm{H}_{k-L} R_{k+j-1}^{gbcg} - \alpha_{k+j-1} \mathcal{A}^{i+1} \mathrm{H}_{k-L} P_{k+j-1}^{gbcg},\\ \mathcal{A}^i \mathrm{H}_{k-L} P_{k+j}^{gbcg} &= \mathcal{A}^i \mathrm{H}_{k-L} R_{k+j}^{gbcg} - \beta_{k+j-1} \mathcal{A}^i \mathrm{H}_{k-L} P_{k+j-1}^{gbcg}.\end{aligned}$$

Rewriting these recursions with the auxiliary matrices gives

$$\mathcal{A}^i S_k^{(j)} = \mathcal{A}^i S_k^{(j-1)} - \alpha_k^{(j-1)} \mathcal{A}^{i+1} Q_k^{(j-1)}, \tag{25}$$
$$\mathcal{A}^i Q_k^{(j)} = \mathcal{A}^i S_k^{(j)} - \beta_k^{(j-1)} \mathcal{A}^i Q_k^{(j-1)}. \tag{26}$$

Thus, when $\mathcal{A}^i S_k^{(j-1)}$ for $i = 0, 1, \dots, L - j$ and $\mathcal{A}^i Q_k^{(j-1)}$ for $i = 0, 1, \dots, L - j + 1$ are given at the beginning of the jth repetition, the new matrices $\mathcal{A}^i S_k^{(j)}$ and $\mathcal{A}^i Q_k^{(j)}$ for $i = 0, 1, \dots, L - j$ can be generated using (21), (25), and (26). This is summarized by the following scheme:

$$\begin{tikzcd}
Q_k^{(j-1)} & & & & Q_k^{(j)} \\
& & & \overset{\text{Eq.(26)}}{\nearrow}\ {\scriptstyle i=0} & \\
\mathcal{A}Q_k^{(j-1)} & S_k^{(j-1)} \arrow[r, "\text{Eq.(25)}", "i=0"'] & S_k^{(j)} & & \mathcal{A}Q_k^{(j)} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\mathcal{A}^{L-j} Q_k^{(j-1)} & \mathcal{A}^{L-j-1} S_k^{(j-1)} \arrow[r, "\text{Eq.(25)}", "i=L-j-1"'] & \mathcal{A}^{L-j-1} S_k^{(j)} & & \mathcal{A}^{L-j} Q_k^{(j)} \\
& & & \overset{\text{Eq.(26)}}{\nearrow}\ {\scriptstyle i=L-j} & \\
\mathcal{A}^{L-j+1} Q_k^{(j-1)} & \mathcal{A}^{L-j} S_k^{(j-1)} \arrow[r, "\text{Eq.(25)}", "i=L-j"'] & \mathcal{A}^{L-j} S_k^{(j)} & &
\end{tikzcd} \tag{27}$$

Here, the starting matrices $\mathcal{A}^i S_k^{(0)} = \mathcal{A}^i R_{k-L}^{(L)}$ for $i = 0, 1, \dots, L - 1$ and $\mathcal{A}^i Q_k^{(0)} = \mathcal{A}^i P_{k-L}^{(L)}$ for $i = 0, 1, \dots, L$ are generated in the previous cycle. We note that when j increases, the number of matrix updates decreases in the scheme (27), whereas it increases in the scheme (22). Thus, repeating (27) for $j = 1, 2, \dots, L$ gives $S_k^{(L)}$ and $Q_k^{(L)}$.

We now consider the computation of $Z_k^{(L)} = \mathrm{G}_{k-1} R_{k+L}^{gbcg}$. From $\mathcal{A}\mathrm{G}_{k-1} = \mathrm{H}_{k-L} - \mathrm{H}_k$ and the Gl-BiCG recursion for the residuals, we have

$$\begin{aligned}\mathcal{A}\mathrm{G}_{k-1} P_{k+j-1}^{gbcg} &= \mathrm{H}_{k-L} P_{k+j-1}^{gbcg} - \mathrm{H}_k P_{k+j-1}^{gbcg},\\ \mathrm{G}_{k-1} R_{k+j}^{gbcg} &= \mathrm{G}_{k-1} R_{k+j-1}^{gbcg} - \alpha_{k+j-1} \mathcal{A}\mathrm{G}_{k-1} P_{k+j-1}^{gbcg},\end{aligned}$$

and rewriting them gives

$$U_k^{(j-1)} = Q_k^{(j-1)} - P_k^{(j-1)}, \tag{28}$$
$$Z_k^{(j)} = Z_k^{(j-1)} - \alpha_k^{(j-1)} U_k^{(j-1)}. \tag{29}$$

Therefore, we generate $Z_k^{(L)}$ by repeating (29) with (28) for $j = 1, 2, \ldots, L$, where $U_k^{(0)} := Q_k^{(0)} - P_k^{(0)}$ and $Z_k^{(0)} := \mathrm{G}_{k-1} R_k^{gbcg}$. Because it holds that

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

application to R_{k+L}^{gbcg} produces a recursion formula for computing the next starting matrix $Z_{k+L}^{(0)}$ as follows:

$$Z_{k+L}^{(0)} = \zeta_{k,1} R_k^{(L)} + \zeta_{k,2}\mathcal{A}R_k^{(L)} + \cdots + \zeta_{k,L}\mathcal{A}^{L-1}R_k^{(L)} + \eta_k Z_k^{(L)}. \qquad (30)$$

We now have all the auxiliary matrices required for the cycles.

3.3 Refined Gl-GPBiCGstab(L) algorithm

By combining (15)–(30), we obtain the refined Gl-GPBiCGstab(L) algorithm shown in Algorithm 2. We refer to Section 2 for the notation. Algorithms 1 and 2 generate the same approximations in exact arithmetic, but they use different formulas, especially for computing Y and U, and their numerical behaviors are different in finite-precision arithmetic. The total computational costs of Algorithms 1 and 2 are identical. However, Algorithm 2 has slightly simpler implementation because unlike Algorithm 1, there is no branch in the loops with j and no recursions of Y and U. Moreover, this simplicity is useful when applying preconditioning. As we show in later sections, Algorithm 2 with right preconditioning is numerically more stable and has a lower computational cost than that of Algorithm 1.

```
1:  Select an initial guess X.
2:  Compute $\mathrm{R}_0 = \mathrm{B} - \mathcal{A}(\mathrm{X})$ and choose $\tilde{\mathrm{R}}_0$.
3:  Set $\mathrm{R} = [\mathrm{R}_0]$, $\mathrm{P} = [\mathrm{R}_0]$, $\mathrm{S} = \left[\mathrm{O}_{n\times s}^{(0)}; \ldots; \mathrm{O}_{n\times s}^{(L-1)}\right]$, and $\mathrm{Q} = \left[\mathrm{O}_{n\times s}^{(0)}; \ldots; \mathrm{O}_{n\times s}^{(L)}\right]$.
4:  $\mathrm{Z} = \mathrm{O}_{n\times s}$
5:  for $k = 0, 1, \ldots,$ until convergence do
6:      $\rho = \langle \tilde{\mathrm{R}}_0, \mathrm{R}_0\rangle_F$
7:      for $j = 1, 2, \ldots, L$ do
8:          $\mathrm{P} = [\mathrm{P}; \mathcal{A}(\mathrm{P}_{j-1})]$,    $\mathrm{U} = \mathrm{Q}_0 - \mathrm{P}_0$
9:          $\sigma = \langle \tilde{\mathrm{R}}_0, \mathrm{P}_j\rangle_F$,    $\alpha = \rho/\sigma$
10:         $\mathrm{X} = \mathrm{X} + \alpha\mathrm{P}_0$,    $\mathrm{Z} = \mathrm{Z} - \alpha\mathrm{U}$
11:         $\mathrm{R} = \mathrm{R} - \alpha[\mathrm{P}_1; \ldots; \mathrm{P}_j]$,    $\mathrm{R} = [\mathrm{R}; \mathcal{A}(\mathrm{R}_{j-1})]$
12:         $\rho = \langle \tilde{\mathrm{R}}_0, \mathrm{R}_j\rangle_F$,    $\beta = \rho/\sigma$,    $\mathrm{P} = \mathrm{R} - \beta\mathrm{P}$
13:         $\mathrm{S} = [\mathrm{S}_0; \ldots; \mathrm{S}_{L-j}] - \alpha[\mathrm{Q}_1; \ldots; \mathrm{Q}_{L-j+1}]$
14:         $\mathrm{Q} = \mathrm{S} - \beta[\mathrm{Q}_0; \ldots; \mathrm{Q}_{L-j}]$
15:     end for
16:     $\mathrm{Y} = \mathrm{S}_0 - \mathrm{R}_0$,    $\mathrm{U} = \mathrm{Q}_0 - \mathrm{P}_0$
17:     Set $\mathrm{S} = [\mathrm{R}_0; \ldots; \mathrm{R}_{L-1}]$ and $\mathrm{Q} = \mathrm{P}$.
18:     Solve $\min_{\zeta_i,\eta} \left\| \mathrm{R}_0 - \sum_{i=1}^{L} \zeta_i \mathrm{R}_i - \eta\mathrm{Y} \right\|_F$    (if $k = 0$ then $\eta = 0$ end if).
19:     $\mathrm{Z} = \zeta_1\mathrm{R}_0 + \cdots + \zeta_L\mathrm{R}_{L-1} + \eta\mathrm{Z}$,    $\mathrm{X} = \mathrm{X} + \mathrm{Z}$
20:     $\mathrm{R} = \mathrm{R}_0 - \zeta_1\mathrm{R}_1 - \cdots - \zeta_L\mathrm{R}_L - \eta\mathrm{Y}$
21:     $\mathrm{P} = \mathrm{P}_0 - \zeta_1\mathrm{P}_1 - \cdots - \zeta_L\mathrm{P}_L - \eta\mathrm{U}$
22: end for
```

Algorithm 2 Gl-GPBiCGstab(L) for (1) (refined)

We briefly describe the related Gl-LTPMs derived from Gl-GPBiCGstab(L). If the parameter η is set to 0 in Algorithms 1 and 2, the computations of relaxation vanish, and both algorithms reduce to the same Gl-BiCGstab(L) algorithm. To the best of our knowledge, this is also a new Gl-LTPM. However, Algorithms 1 and 2 with $L = 1$ are mathematically equivalent to Gl-GPBiCG [15, Algorithm 2] (and also Gl-GPBiCG-plus [15, Algorithm 4]), but they use different implementations. As in the case of standard LTPMs, the global versions of CGS, BiCGSTAB, and BiCGStab2 can be derived from the framework of Gl-GPBiCG; therefore, Gl-GPBiCGstab(L) includes them as well. Thus, Algorithms 1 and 2 can be reduced to various global versions and have different implementations. However, because the main purpose of this study is to develop an effective method for Gl-GPBiCGstab(L), we will not discuss the details of the simplified algorithms further or their implementations. We consider several key algorithms and compare their convergence later in numerical experiments.

4 Preconditioning for linear systems with multiple right-hand sides

In this section, we focus on linear systems with multiple right-hand sides

$$AX = B, \quad A \in \mathbb{R}^{n\times n}, \quad B \in \mathbb{R}^{n\times s}, \tag{31}$$

which constitute one of the most important types of linear matrix equations [7–10]. Here, A is a large sparse nonsymmetric and nonsingular matrix. Because the preconditioning technique is useful for enhancing the convergence of Gl-LTPMs when solving (31), we discuss the preconditioned algorithms of Gl-GPBiCGstab(L).

There are several approaches for converting the above system into a well-conditioned system using a preconditioner $K \approx A$. Multiplying (31) by K^{-1} from the left side gives the left preconditioned system

$$\hat{A}X = \hat{B}, \quad \hat{A} := K^{-1}A, \quad \hat{B} := K^{-1}B.$$

In general, the left preconditioned algorithm is easy to implement by replacing only the multiplications by A with those by $K^{-1}A$, but the residuals generated in the iterations change to $\hat{R}_k = K^{-1}R_k$; we therefore need special care in setting the stopping rule. Conversely, the right preconditioned system is given in the form

$$\hat{A}\hat{X} = B, \quad \hat{A} := AK^{-1}, \quad \hat{X} := KX. \tag{32}$$

In this case, the residuals coincide with the standard ones, i.e., $\hat{R}_k = R_k$, and the approximations generated in the iterations change to $\hat{X}_k = KX_k$. The approximation $X_k = K^{-1}\hat{X}_k$ needs to be computed accurately once the iterations are terminated. Alternatively, to capture the approximations during the iterations, it is well known that we can update X_k recursively instead of $\hat{X}_k$ through some changes in the variables (cf., e.g., [2, 18]). We describe such algorithms of Gl-GPBiCGstab(L) with right preconditioning.

```
1: Select an initial guess X.
2: Compute $\mathrm{R}_0 = \mathrm{B} - A\mathrm{X}$ and choose $\tilde{\mathrm{R}}_0$.
3: Set $\mathrm{R} = [\mathrm{R}_0]$, $\hat{\mathrm{R}} = [K^{-1}\mathrm{R}_0]$, $\hat{\mathrm{P}} = [\hat{\mathrm{R}}_0]$, and $\mathrm{Q} = \hat{\mathrm{Q}} = [\mathrm{O}_{n\times s}]$.
4: $\hat{\mathrm{Z}} = \hat{\mathrm{U}} = \hat{\mathrm{Y}} = \mathrm{Y} = \mathrm{O}_{n\times s}$
5: for $k = 0, 1, \ldots,$ until convergence do
6:     $\mathrm{P} = [\,]$,   $\rho = \langle \tilde{\mathrm{R}}_0, \mathrm{R}_0\rangle_F$
7:     for $j = 1, 2, \ldots, L$ do
8:         if $k \neq 0$ && $j > 1$ then
9:             $\mathrm{S} = [\mathrm{S}_0; \ldots; \mathrm{S}_{L-j}] - \alpha[\mathrm{Q}_1; \ldots; \mathrm{Q}_{L-j+1}]$
10:            $\hat{\mathrm{S}} = [\hat{\mathrm{S}}_0; \ldots; \hat{\mathrm{S}}_{L-j}] - \alpha[\hat{\mathrm{Q}}_1; \ldots; \hat{\mathrm{Q}}_{L-j+1}]$
11:            $\mathrm{Q} = \mathrm{S} - \beta[\mathrm{Q}_0; \ldots; \mathrm{Q}_{L-j}]$,   $\hat{\mathrm{Q}} = \hat{\mathrm{S}} - \beta[\hat{\mathrm{Q}}_0; \ldots; \hat{\mathrm{Q}}_{L-j}]$
12:        end if
13:        $\mathrm{P} = [\mathrm{P}; A\hat{\mathrm{P}}_{j-1}]$,   $\hat{\mathrm{P}} = [\hat{\mathrm{P}}; K^{-1}\mathrm{P}_{j-1}]$,   $\mathrm{V} = \mathrm{Q}_0 - \mathrm{P}_0$,   $\hat{\mathrm{V}} = \hat{\mathrm{Q}}_0 - \hat{\mathrm{P}}_1$
14:        $\sigma = \langle \tilde{\mathrm{R}}_0, \mathrm{P}_{j-1}\rangle_F$,   $\alpha = \rho/\sigma$
15:        $\mathrm{X} = \mathrm{X} + \alpha\hat{\mathrm{P}}_0$,   $\hat{\mathrm{Z}} = \hat{\mathrm{Z}} - \alpha\hat{\mathrm{U}}$,   $\hat{\mathrm{Y}} = \hat{\mathrm{Y}} - \alpha\hat{\mathrm{V}}$,   $\mathrm{Y} = \mathrm{Y} - \alpha\mathrm{V}$
16:        $\mathrm{R} = \mathrm{R} - \alpha\mathrm{P}$,   $\hat{\mathrm{R}} = \hat{\mathrm{R}} - \alpha[\hat{\mathrm{P}}_1; \ldots; \hat{\mathrm{P}}_j]$
17:        $\mathrm{R} = [\mathrm{R}; A\hat{\mathrm{R}}_{j-1}]$,   $\hat{\mathrm{R}} = [\hat{\mathrm{R}}; K^{-1}\mathrm{R}_j]$
18:        $\rho = \langle \tilde{\mathrm{R}}_0, \mathrm{R}_j\rangle_F$,   $\beta = \rho/\sigma$
19:        $\mathrm{P} = [\mathrm{R}_1; \ldots; \mathrm{R}_j] - \beta\mathrm{P}$,   $\hat{\mathrm{P}} = \hat{\mathrm{R}} - \beta\hat{\mathrm{P}}$,   $\hat{\mathrm{U}} = \hat{\mathrm{Y}} - \beta\hat{\mathrm{U}}$
20:    end for
21:    Set $\mathrm{S} = [\mathrm{R}_1; \ldots; \mathrm{R}_{L-1}]$, $\mathrm{Q} = [\mathrm{P}_0; \ldots; \mathrm{P}_{L-1}]$, and $\mathrm{R}' = \mathrm{R}_0$.
22:    Set $\hat{\mathrm{S}} = [\hat{\mathrm{R}}_1; \ldots; \hat{\mathrm{R}}_{L-1}]$, $\hat{\mathrm{Q}} = [\hat{\mathrm{P}}_1; \ldots; \hat{\mathrm{P}}_L]$, $\hat{\mathrm{R}}' = \hat{\mathrm{R}}_0$, and $\hat{\mathrm{P}}' = \hat{\mathrm{P}}_0$.
23:    Solve $\min_{\zeta_i,\eta} \left\| \mathrm{R}_0 - \sum_{i=1}^{L} \zeta_i \mathrm{R}_i - \eta\mathrm{Y} \right\|_F$   (if $k = 0$ then $\eta = 0$ end if).
24:    $\hat{\mathrm{Z}} = \zeta_1\hat{\mathrm{R}}_0 + \cdots + \zeta_L\hat{\mathrm{R}}_{L-1} + \eta\hat{\mathrm{Z}}$,   $\mathrm{X} = \mathrm{X} + \hat{\mathrm{Z}}$
25:    $\mathrm{R} = \mathrm{R}_0 - \zeta_1\mathrm{R}_1 - \cdots - \zeta_L\mathrm{R}_L - \eta\mathrm{Y}$,   $\mathrm{Y} = \mathrm{R}' - \mathrm{R}$
26:    $\hat{\mathrm{R}} = \hat{\mathrm{R}}_0 - \zeta_1\hat{\mathrm{R}}_1 - \cdots - \zeta_L\hat{\mathrm{R}}_L - \eta\hat{\mathrm{Y}}$,   $\hat{\mathrm{Y}} = \hat{\mathrm{R}}' - \hat{\mathrm{R}}$
27:    $\hat{\mathrm{P}} = \hat{\mathrm{P}}_0 - \zeta_1\hat{\mathrm{P}}_1 - \cdots - \zeta_L\hat{\mathrm{P}}_L - \eta\hat{\mathrm{U}}$,   $\hat{\mathrm{U}} = \hat{\mathrm{P}}' - \hat{\mathrm{P}}$
28: end for
```

Algorithm 3 Right preconditioned Gl-GPBiCGstab(L) for (31) (naive)

4.1 Naive right preconditioned algorithm

Algorithm 3 is a naive right preconditioned Gl-GPBiCGstab(L) algorithm that is obtained by applying Algorithm 1 to (32). Similar to [2, 18], several variables are changed to update X_k recursively, and the variables with the hat symbol '$\hat{\ }$' denote that K^{-1} acts on its underlying variables (without hats). For example, we have the following relations in Algorithm 3:

$$\mathrm{R} = \left[\mathrm{R}_0; \mathrm{R}_1; \ldots; \mathrm{R}_j\right] = \left[R_k^{(j)}; \left(AK^{-1}\right) R_k^{(j)}; \ldots; \left(AK^{-1}\right)^j R_k^{(j)}\right],$$

$$\hat{\mathrm{R}} = \left[\hat{\mathrm{R}}_0; \hat{\mathrm{R}}_1; \ldots; \hat{\mathrm{R}}_j\right] = \left[K^{-1}R_k^{(j)}; K^{-1}\left(AK^{-1}\right) R_k^{(j)}; \ldots; K^{-1}\left(AK^{-1}\right)^j R_k^{(j)}\right],$$

$$\mathrm{P} = [\mathrm{P}_0; \mathrm{P}_1; \ldots; \mathrm{P}_{j-1}] = \left[\left(AK^{-1}\right) P_k^{(j)}; \left(AK^{-1}\right)^2 P_k^{(j)}; \ldots; \left(AK^{-1}\right)^j P_k^{(j)}\right],$$

$$\hat{\mathrm{P}} = \left[\hat{\mathrm{P}}_0; \hat{\mathrm{P}}_1; \ldots; \hat{\mathrm{P}}_j\right] = \left[K^{-1} P_k^{(j)}; K^{-1}\left(AK^{-1}\right) P_k^{(j)}; \ldots; K^{-1}\left(AK^{-1}\right)^j P_k^{(j)}\right],$$

where we note that $P_k^{(j)}$ itself is not needed for updating X_k, and we set $\mathrm{P}_0 := (AK^{-1})P_k^{(j)}$. Similar relationships also hold between other variables with and without hats.

We briefly describe some concerns regarding Algorithm 3. Because $\hat{\mathrm{R}}$ is computed by a linear combination of $\hat{\mathrm{R}}_i$ for $i = 0, 1, \ldots, L$ and $\hat{\mathrm{Y}}$ in line 26, it may differ significantly from $K^{-1}\mathrm{R}$ owing to accumulation of the rounding errors. In our experience, this causes numerical instabilities, such as stagnation and divergence of the residual norms in the late stages of the iterations (see Section 5.1 below). In our preliminary experiments, we confirmed that the convergence can be improved by computing $\hat{\mathrm{R}}$ explicitly with the form $K^{-1}\mathrm{R}$. However, this approach is costly because it requires additional multiplications by K^{-1}. We note that because $\hat{\mathrm{Y}}$ and $\hat{\mathrm{U}}$ are computed by their coupled recursions with the starting matrix $\hat{\mathrm{Y}} = \hat{\mathrm{R}}' - \hat{\mathrm{R}}$ given in line 26, it is difficult to remove this line without using additional multiplications with K^{-1}.

4.2 Refined right preconditioned algorithm

We can overcome the above difficulty by exploiting Algorithm 2. Algorithm 4 presents the refined right preconditioned Gl-GPBiCGstab(L) algorithm from applying Algorithm 2 to (32). Here, because $\hat{\mathrm{U}}$ is computed directly in the form $\hat{\mathrm{U}} = \hat{\mathrm{Q}}_0 - \hat{\mathrm{P}}_0$, we do not need $\hat{\mathrm{Y}}$, which enables removal of the linear combination for $\hat{\mathrm{R}}$. Then, unlike Algorithm 3, multiplication by K^{-1} can be used to compute the following variables in the jth repetition.

$$\begin{aligned}\hat{\mathrm{R}}_{j-1} &= K^{-1}\mathrm{R}_{j-1} = K^{-1}\left(AK^{-1}\right)^{j-1} R_k^{(j)},\\ \hat{\mathrm{P}}_j &= K^{-1}\mathrm{P}_{j-1} = K^{-1}\left(AK^{-1}\right)^j P_k^{(j)}.\end{aligned}$$

Hence, $\hat{\mathrm{R}}$ is always a good approximation of $K^{-1}\mathrm{R}$ in finite-precision arithmetic, and we expect that Algorithm 4 is numerically more stable than Algorithm 3.

Note that because multiplications with K^{-1} are used in different parts between Algorithms 3 and 4, the algorithms with $\eta = 0$ result in different formulations of the right preconditioned Gl-BiCGstab(L).

4.3 Computational costs and memory requirements

We compare the computational costs and memory requirements between the related right preconditioned algorithms. We consider the original Gl-GPBiCG and

```
1: Select an initial guess X.
2: Compute $\mathrm{R}_0 = \mathrm{B} - A\mathrm{X}$ and choose $\tilde{\mathrm{R}}_0$.
3: Set $\mathrm{R} = [\mathrm{R}_0]$ and $\hat{\mathrm{P}} = [K^{-1}\mathrm{R}_0]$.
4: Set $\mathrm{S} = \hat{\mathrm{S}} = \mathrm{Q} = \left[\mathrm{O}_{n\times s}^{(0)}; \dots; \mathrm{O}_{n\times s}^{(L-1)}\right]$ and $\hat{\mathrm{Q}} = \left[\mathrm{O}_{n\times s}^{(0)}; \dots; \mathrm{O}_{n\times s}^{(L)}\right]$.
5: $\hat{\mathrm{Z}} = \mathrm{O}_{n\times s}$
6: for $k = 0, 1, \dots,$ until convergence do
7:     $\mathrm{P} = [\,], \quad \hat{\mathrm{R}} = [\,], \quad \rho = \langle \tilde{\mathrm{R}}_0, \mathrm{R}_0 \rangle_F$
8:     for $j = 1, 2, \dots, L$ do
9:         $\mathrm{P} = [\mathrm{P}; A\hat{\mathrm{P}}_{j-1}], \quad \hat{\mathrm{U}} = \hat{\mathrm{Q}}_0 - \hat{\mathrm{P}}_0$
10:        $\sigma = \langle \tilde{\mathrm{R}}_0, \mathrm{P}_{j-1} \rangle_F, \quad \alpha = \rho/\sigma$
11:        $\mathrm{X} = \mathrm{X} + \alpha\hat{\mathrm{P}}_0, \quad \hat{\mathrm{Z}} = \hat{\mathrm{Z}} - \alpha\hat{\mathrm{U}}, \quad \mathrm{R} = \mathrm{R} - \alpha\mathrm{P}$
12:        if $j \neq 1$ then $\hat{\mathrm{R}} = \hat{\mathrm{R}} - \alpha\left[\hat{\mathrm{P}}_1; \dots; \hat{\mathrm{P}}_{j-1}\right]$ end if
13:        $\hat{\mathrm{R}} = \left[\hat{\mathrm{R}}; K^{-1}\mathrm{R}_{j-1}\right], \quad \mathrm{R} = \left[\mathrm{R}; A\hat{\mathrm{R}}_{j-1}\right]$
14:        $\rho = \langle \tilde{\mathrm{R}}_0, \mathrm{R}_j \rangle_F, \quad \beta = \rho/\sigma$
15:        $\mathrm{P} = \left[\mathrm{R}_1; \dots; \mathrm{R}_j\right] - \beta\mathrm{P}, \quad \hat{\mathrm{P}} = \hat{\mathrm{R}} - \beta\hat{\mathrm{P}}, \quad \hat{\mathrm{P}} = \left[\hat{\mathrm{P}}; K^{-1}\mathrm{P}_{j-1}\right]$
16:        $\mathrm{S} = \left[\mathrm{S}_0; \dots; \mathrm{S}_{L-j}\right] - \alpha\left[\mathrm{Q}_0; \dots; \mathrm{Q}_{L-j}\right]$
17:        $\hat{\mathrm{S}} = \left[\hat{\mathrm{S}}_0; \dots; \hat{\mathrm{S}}_{L-j}\right] - \alpha\left[\hat{\mathrm{Q}}_1; \dots; \hat{\mathrm{Q}}_{L-j+1}\right]$
18:        if $j \neq L$ then $\mathrm{Q} = \left[\mathrm{S}_1; \dots; \mathrm{S}_{L-j}\right] - \beta\left[\mathrm{Q}_0; \dots; \mathrm{Q}_{L-j-1}\right]$ end if
19:        $\hat{\mathrm{Q}} = \left[\hat{\mathrm{S}}_0; \dots; \hat{\mathrm{S}}_{L-j}\right] - \beta\left[\hat{\mathrm{Q}}_0; \dots; \hat{\mathrm{Q}}_{L-j}\right]$
20:    end for
21:    $\mathrm{Y} = \mathrm{S}_0 - \mathrm{R}_0, \quad \hat{\mathrm{U}} = \hat{\mathrm{Q}}_0 - \hat{\mathrm{P}}_0$
22:    Set $\mathrm{S} = \left[\mathrm{R}_0; \dots; \mathrm{R}_{L-1}\right]$, $\mathrm{Q} = \mathrm{P}$, $\hat{\mathrm{S}} = \hat{\mathrm{R}}$, and $\hat{\mathrm{Q}} = \hat{\mathrm{P}}$.
23:    Solve $\min_{\zeta_i,\eta} \left\| \mathrm{R}_0 - \sum_{i=1}^{L} \zeta_i \mathrm{R}_i - \eta\mathrm{Y} \right\|_F$ (if $k = 0$ then $\eta = 0$ end if).
24:    $\hat{\mathrm{Z}} = \zeta_1\hat{\mathrm{R}}_0 + \dots + \zeta_L\hat{\mathrm{R}}_{L-1} + \eta\hat{\mathrm{Z}}, \quad \mathrm{X} = \mathrm{X} + \hat{\mathrm{Z}}$
25:    $\mathrm{R} = \mathrm{R}_0 - \zeta_1\mathrm{R}_1 - \dots - \zeta_L\mathrm{R}_L - \eta\mathrm{Y}$
26:    $\hat{\mathrm{P}} = \hat{\mathrm{P}}_0 - \zeta_1\hat{\mathrm{P}}_1 - \dots - \zeta_L\hat{\mathrm{P}}_L - \eta\hat{\mathrm{U}}$
27: end for
```

Algorithm 4 Right preconditioned Gl-GPBiCGstab(L) for (31) (refined)

Gl-GPBiCG-plus with right preconditioning given in [15] as well as the naive and refined variants of Gl-BiCGstab(L) and Gl-GPBiCGstab(L) with right preconditioning. Note that the evaluations for the naive and refined Gl-GPBiCG derived from Gl-GPBiCGstab(L) are obtained by substituting $L = 1$ in the results of Algorithms 3 and 4, respectively.

Below, "MM" denotes a matrix–matrix product with A. Although Gl-GPBiCG and Gl-GPBiCG-plus require two MMs per iteration while the others require $2L$ MMs per cycle, all methods require one MM per increase by one Krylov dimension on average. A single multiplication by K^{-1} is also required per MM for all the methods. The number of AXPYs and DOTs per MM as well as their memory requirements are summarized in Table 1, where AXPY and DOT correspond to the forms (4) and (5), respectively. Following [6], the form aX or $X + Y$ is counted as 1/2 AXPYs.

Table 1 Computational cost per MM and memory requirements with right preconditioning

Algorithms	**AXPYs**	**DOTs**	**Memory**
Gl-GPBiCG [15, Algorithm 3]	$\frac{31}{4}$	$\frac{7}{2}$	14
Gl-GPBiCG-plus [15, Algorithm 5]	$\frac{27}{4}$	$\frac{7}{2}$	13
Gl-BiCGstab(L) (naive) (Algorithm 3 with $\eta = 0$)	$L + 4$	$\frac{L}{4} + \frac{7}{4} + \frac{1}{2L}$	$4L + 5$
Gl-BiCGstab(L) (refined) (Algorithm 4 with $\eta = 0$)	$L + \frac{5}{2}$	$\frac{L}{4} + \frac{7}{4} + \frac{1}{2L}$	$4L + 4$
Gl-GPBiCGstab(L) (naive) (Algorithm 3)	$2L + \frac{11}{2} + \frac{11}{4L}$	$\frac{L}{4} + \frac{9}{4} + \frac{3}{2L}$	$8L + 10$
Gl-GPBiCGstab(L) (refined) (Algorithm 4)	$2L + \frac{15}{4} + \frac{2}{L}$	$\frac{L}{4} + \frac{9}{4} + \frac{3}{2L}$	$8L + 8$

The computational costs for checking the stopping rule are not included. The memory requirements indicate the amount of $n \times s$ matrices that must be stored in the algorithms. The memories for A and B are not counted.

From Table 1, we see that Algorithm 4 has an additional advantage that it requires lower computational cost and memory requirements than Algorithm 3. The costs and memories of Algorithm 4 for a modest value of L, e.g., $L \leq 4$, are not high compared with those of the Gl-GPBiCG and Gl-BiCGstab(L) algorithms.

5 Numerical experiments

We present numerical experiments to show the effectiveness of the proposed Gl-GPBiCGstab(L) method. Numerical calculations were carried out in double-precision floating-point arithmetic on a PC (Intel Core i7-1185G7 CPU with 32 GB of RAM) equipped with MATLAB 2021a. The right-hand side $B \in \mathbb{R}^{n\times s}$ was given by a random matrix. The initial guess X_0 and initial shadow residual $\tilde{R}_0$ were set to O and R_0 $(= B)$, respectively. The least-squares problem (6) was converted to a normal equation (7) and solved using the backslash command in MATLAB. The other computational conditions were set individually for each example, as noted below.

5.1 Comparison of the naive and refined algorithms

We first compare convergence between the naive and refined Gl-GPBiCGstab(L) algorithms with and without preconditioning. Algorithms 1–4 were applied to (31).

Following [6], the coefficient matrix A was given by the following Toeplitz matrix:

$$A := \begin{bmatrix} 2 & 1 & & \\ 0 & 2 & 1 & \\ 0 & 0 & 2 & \ddots \\ 0 & 0 & 0 & \ddots \\ 1.4 & 0 & 0 & \ddots \\ & 1.4 & 0 & \ddots \\ & & 1.4 & \ddots \\ & & & \ddots \end{bmatrix} \in \mathbb{R}^{500\times 500}.$$

The iterations were stopped when the relative residual norms $\|R_k\|_F/\|B\|_F$ were less than 10^{-14}. The parameters s and L were set to $s = 1, 2, 4, 8, 16, 32$ and $L = 2, 4, 8$, respectively. The maximum number of MMs was set to $2n$. ILU(0) [19] was used as the preconditioner in Algorithms 3 and 4.

Figure 1 shows the convergence histories of the relative residual norms of Algorithms 3 and 4, i.e., the naive and refined Gl-GPBiCGstab(L) with right preconditioning, for $s = 1$ and 32. The plots indicate the number of MMs along the horizontal axis versus $\log_{10}$ of the relative residual F-norm along the vertical axis. Note that "MV" (matrix–vector product) and 2-norm are utilized for $s = 1$ instead of MM and F-norm, respectively. Table 2 shows the number of MMs required for successful convergence of the four algorithms. The symbol † in Table 2 indicates that the residual norms stagnate or diverge at the late stages of the iterations, as displayed in Fig. 1.

From Fig. 1 and Table 2, we observe the following. The numbers of MMs required for successful convergence of Algorithms 1 and 2 (without preconditioning) are comparable for each value of s and L. In contrast, in the preconditioned case, Algorithm 4 achieves faster convergence than in the non-preconditioned case for all s and L, whereas Algorithm 3 often does not converge. As noted in Section 4.1, $\hat{\mathrm{R}}$ in Algorithm 3 may differ from $K^{-1}\mathrm{R}$ because of rounding errors, and this difference is expected to increase with increasing L. Actually, Algorithm 3 becomes unstable as parameter L increases for a fixed s. Similar results have been observed for

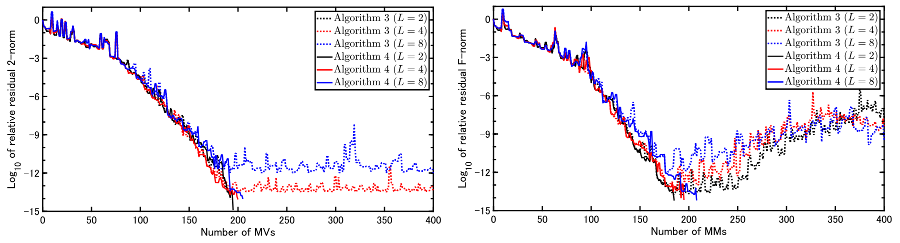

Fig. 1 Convergence histories of Algorithms 3 and 4 (naive and refined Gl-GPBiCGstab(L) with right preconditioning, respectively) for $s = 1$ (left) and $s = 32$ (right)

Table 2 Number of MMs for Algorithms 1–4 for the Toeplitz matrix

Solvers	$L \setminus s$	1	2	4	8	16	32
Algorithm 1	2	761	696	693	752	688	720
	4	677	648	672	663	653	624
	8	641	640	656	653	640	653
Algorithm 2	2	755	701	707	707	676	685
	4	641	665	671	683	632	624
	8	641	641	635	640	640	649
Algorithm 3	2	193	196	193	192	188	†
	4	†	200	193	†	†	†
	8	†	†	†	†	†	†
Algorithm 4	2	195	204	191	197	184	185
	4	200	199	192	208	200	195
	8	205	208	208	208	208	208

other matrices. Moreover, because stagnation of the residual norms occurs even when $s = 1$, there is an inherent problem in the original GPBiCGstab(L) with right preconditioning, but the numerical instability seems to grow with increasing s. For the above reasons, we conclude that our refined Gl-GPBiCGstab(L) algorithm is more robust than the naive version with right preconditioning.

5.2 Experiments for linear systems with multiple right-hand sides

Next, we apply several Gl-LTPMs with preconditioning to (31) and compare their convergences. We use Gl-BiCGSTAB, Gl-GPBiCG, Gl-BiCGstab(L), and Gl-GPBiCGstab(L), which can be obtained from Algorithm 4. For Gl-GPBiCG, we also use the Gl-GPBiCG-plus implementation [15, Algorithm 5]. Table 3 displays the abbreviations of the solvers and their corresponding algorithms. Right preconditioning was applied to all methods, and the ILU(0) preconditioner was used.

We consider test matrices from the SuiteSparse Matrix Collection [20]. Table 4 shows the dimension ($\boldsymbol{n}$), number of nonzero entries ($\boldsymbol{nnz}$), and 2-norm condition number ($\boldsymbol{\kappa_2}$) of each matrix. The condition number is displayed only if it is given in the above collection. The iterations are stopped when the relative residual norms $\|R_k\|_F/\|B\|_F$ are less than 10^{-10}. The maximum number of MMs was set to $2n$. For Gl-BiCGstab(L) and Gl-GPBiCGstab(L), we use $L = 2, 4$. The number of right-hand sides s is set to 16.

Figure 2 displays the convergence histories of the relative residual norms of Gl-LTPMs with right preconditioning for sme3Db and garon2. We refer to Section 5.1 for the plots of the figures. Table 5 shows the number of MMs required for successful convergence (**MMs**), computation time (**Time**), and explicitly computed relative residual norm (referred to as the true relative residual norm) $\|B - AX_k\|_F/\|B\|_F$ (**TRR**) at the time of termination of the solvers. Following [6], the smallest **MMs**

Table 3 Solvers used in Section 5.2

Abbreviations	Solvers[1]	Algorithms
-STAB	Gl-BiCGSTAB	Algorithm 4 with $L = 1$ and $\eta = 0$
GP-plus	Gl-GPBiCG-plus	[15, Algorithm 5]
GP-	Gl-GPBiCG	Algorithm 4 with $L = 1$
-stab($\boldsymbol{L}$)	Gl-BiCGstab(L)	Algorithm 4 with $\eta = 0$
GP-stab($\boldsymbol{L}$)	Gl-GPBiCGstab(L)	Algorithm 4

[1]Right preconditioning is applied to all solvers

and **Time** are displayed in boldface font in italics, and the second smallest values are displayed only in italics for each problem. “Inf”’ and “NaN” in the table indicate that the iteration values are “infinity” and “not a number”, respectively, when iterating with MATLAB and that the iterations cannot proceed thereafter.

From Fig. 2 and Table 5, we observe the following. The fastest convergence in terms of number of MMs is achieved by the proposed Gl-GPBiCGstab(L) for all problems. In particular, Gl-GPBiCGstab(4) often converges faster than not only the other Gl-LTPMs but also Gl-GPBiCGstab(2). As noted in [6], moderately increasing L seems to be useful for reducing the number of MMs required for successful convergence. These are important metrics of the effectiveness of Gl-GPBiCGstab(L) as a global Krylov subspace method because the number of MMs coincides with the dimensions of the matrix Krylov subspace. In terms of the computational time, Gl-BiCGstab(L) and Gl-GPBiCGstab(L) are comparable to the conventional Gl-GPBiCG-plus on average. Comparing the true relative residual norms, there are cases where Gl-BiCGstab(L) and Gl-GPBiCGstab(L) generate slightly more accurate approximate solutions than Gl-GPBiCG-plus.

We also remark on the numerical results briefly when the stopping criterion was set to $\|R_k\|_F/\|B\|_F < 10^{-12}$, because we obtained slightly different observations from the above. With respect to the convergence speed in terms of the number of MMs, Gl-BiCGstab(4) and Gl-GPBiCGstab(4) are superior to other Gl-LTPMs. Moreover,

Table 4 Characteristics of test matrices for linear systems with multiple right-hand sides (31)

No.	Matrices	n	nnz	κ_2
#1	memplus	17,758	99,147	1.3e+05
#2	utm1700b	1,700	21,509	5.6e+06
#3	cavity16	4,562	137,887	9.4e+06
#4	sme3Db	29,067	2,081,063	3.6e+07
#5	dc1	116,835	766,396	-
#6	ex28	2,603	77,031	2.0e+05
#7	garon2	13,535	373,235	6.1e+07
#8	Goodwin_054	32,510	1,030,878	-

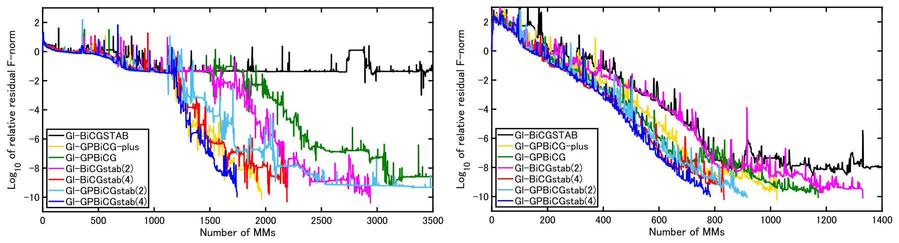

Fig. 2 Convergence histories of the relative residual norms of Gl-LTPMs with right preconditioning for sme3Db (left) and garon2 (right)

Table 5 Numbers of MMs, computation times, and true relative residual norms for Gl-LTPMs with right preconditioning for linear systems with multiple right-hand sides (31)

					-stab(L)		GP-stab(L)	
No.	Items	-STAB	GP-plus	GP-	$L = 2$	$L = 4$	$L = 2$	$L = 4$
#1	**MMs**	948	843	772	863	*752*	761	***680***
	Time	6.142	***5.113***	5.937	6.164	*5.915*	6.879	7.164
	TRR	4.5e−11	4.5e−09	8.3e−11	8.8e−11	9.8e−11	9.7e−11	7.7e−11
#2	**MMs**	438	189	198	185	*167*	173	***160***
	Time	0.314	*0.134*	0.165	0.145	***0.127***	0.162	0.148
	TRR	2.6e−10	4.3e−09	1.1e−09	5.7e−10	6.0e−09	1.0e−09	6.0e−09
#3	**MMs**	1836	594	672	725	*488*	588	***464***
	Time	5.466	1.920	2.354	2.337	***1.561***	2.130	*1.690*
	TRR	1.1e−10	2.1e−10	1.6e−10	8.5e−11	4.1e−11	6.4e−10	9.6e−11
#4	**MMs**	17115	*1963*	6333	2939	2191	5005	***1744***
	Time	1100	*124.9*	419.0	191.6	150.3	339.9	***121.7***
	TRR	3.3e−08	2.7e−09	2.9e−08	3.3e−09	2.7e−09	9.3e−08	3.7e−09
#5	**MMs**	2195	*799*	839	1419	816	***705***	845
	Time	94.04	***33.40***	41.67	64.98	40.37	*39.11*	52.37
	TRR	2.2e−07	7.6e−06	5.2e−04	7.1e−08	7.0e−08	7.3e−08	7.0e−08
#6	**MMs**	212	165	172	188	155	*152*	***151***
	Time	0.377	*0.301*	0.357	0.344	***0.288***	0.315	0.327
	TRR	3.7e−11	2.4e−10	7.6e−11	4.2e−11	1.7e−10	8.0e−11	3.3e−10
#7	**MMs**	Inf	1023	1171	1331	*833*	915	***784***
	Time	-	*10.45*	13.56	14.76	***9.816***	11.65	10.97
	TRR	-	2.8e−10	9.8e−11	9.1e−11	6.2e−11	8.2e−11	9.6e−11
#8	**MMs**	NaN	4451	NaN	*2587*	4513	2879	***2328***
	Time	-	128.6	-	***79.18***	176.0	118.1	*99.26*
	TRR	-	1.1e−10	-	8.0e−11	7.3e−11	1.1e−10	1.2e−10

Table 6 Solvers used in Section 5.3

Abbreviations	Solvers[2]	Algorithms
-STAB	Gl-BiCGSTAB	Algorithm 2 with $L = 1$ and $\eta = 0$
GP-plus	Gl-GPBiCG-plus	[15, Algorithm 4]
GP-	Gl-GPBiCG	Algorithm 2 with $L = 1$
-stab($\boldsymbol{L}$)	Gl-BiCGstab(L)	Algorithm 2 with $\eta = 0$
GP-stab($\boldsymbol{L}$)	Gl-GPBiCGstab(L)	Algorithm 2

[2]No preconditioning is used in all the solvers

only these two solvers converge for all the test matrices; other conventional Gl-LTPMs fail to converge at the late stages of the iterations for some problems. On the other hand, the attainable accuracy in terms of the true residual norm is limited for all the solvers, that is, a so-called large residual gap (the difference between R_k and $B - AX_k$) appears in most cases. This problem would be improved by combining Gl-LTPMs with the techniques described in [21].

5.3 Experiments for the Sylvester equation

Herein, we present the numerical results of the Gl-LTPMs for the Sylvester equation (2). The solvers shown in Table 6 are applied to (2) without preconditioning.

Note that linear operation by $\mathcal{A}$ to $X \in \mathbb{R}^{n\times s}$ is defined as $\mathcal{A}(X) := AX - XC$ for the given matrices $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{s\times s}$, and "MM" is replaced by "OP" to denote an operation with $\mathcal{A}$. Motivated by [15, 22], we set the test matrices A and C as displayed in Table 7. The matrices are derived from the SuiteSparse Matrix Collection [20], except for the tridiagonal matrix in problems #11 and #12. The tridiagonal matrix $C = [c_{ij}]$ is defined as $c_{i+1,i} := 11$, $c_{ii} := -2$, and $c_{i,i+1} := -9$ for each i (otherwise, $c_{ij} := 0$). The value s is determined by the size of C, and the maximum number of OPs was set to $2sn$. All other computational conditions were similar to those in Section 5.2.

Table 7 Characteristics of the test matrices for the Sylvester equation (2)

No.	Matrices	n	nnz	κ_2
#9	$A =$ fs_680_1	680	2,184	1.5e+04
	$C =$ can_24	24	160	7.8e+01
#10	$A =$ fs_680_1	680	2,184	1.5e+04
	$C =$ ibm32	32	126	4.0e+02
#11	$A =$ sherman1	1,000	3,750	1.6e+04
	$C =$ tridiag$(11, -2, -9)$	10	28	8.4e+00
#12	$A =$ sherman4	1,104	3,786	2.2e+03
	$C =$ tridiag$(11, -2, -9)$	10	28	8.4e+00

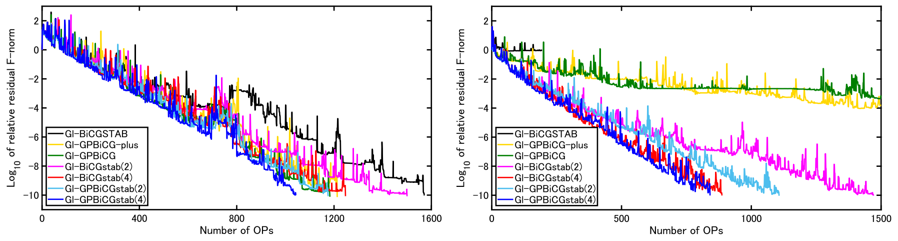

Fig. 3 Convergence histories of the relative residual norms of Gl-LTPMs for problems #9 (left) and #11 (right)

Figure 3 displays the convergence histories of the relative residual norms of the Gl-LTPMs for problems #9 and #11. The number of OPs and $\log_{10}$ of the relative residual F-norms are plotted on the horizontal and vertical axes, respectively. Table 8 shows the number of OPs required for successful convergence (**OPs**), computation time (**Time**), and true relative residual norm $\|B - \mathcal{A}(X_k)\|_F / \|B\|_F$ (**TRR**) at the time of termination. The other notations are as defined in Table 5.

From Fig. 3 and Table 8, we observe the following. Similar to the results in Section 5.2, Gl-GPBiCGstab(L) (especially, with $L = 4$) often converges faster than the other Gl-LTPMs with respect to the number of OPs. The conventional Gl-GPBiCG-plus is efficient in terms of the computational time. However, Gl-GPBiCG-plus does not converge for problem #11, whereas Gl-BiCGstab(L) and Gl-GPBiCGstab(L) converge. Similar situations have occasionally occurred in our

Table 8 Numbers of OPs, computation times, and true relative residual norms for Gl-LTPMs for the Sylvester equation (2)

					-stab(L)		GP-stab(L)	
No.	Items	-STAB	GP-plus	GP-	$L = 2$	$L = 4$	$L = 2$	$L = 4$
#9	**OPs**	1572	1213	1183	1500	1247	*1176*	***1043***
	Time	0.301	***0.212***	0.386	0.338	*0.286*	0.345	0.313
	TRR	1.0e−10	1.4e−10	7.7e−11	8.4e−11	9.4e−11	8.5e−11	1.0e−10
#10	**OPs**	1773	1269	1241	1392	1312	***1184***	*1200*
	Time	0.383	***0.263***	0.479	0.368	*0.315*	0.388	0.370
	TRR	8.5e−11	1.3e−10	7.5e−11	8.4e−11	4.7e−11	9.7e−11	1.5e−09
#11	**OPs**	Inf	NaN	9430	1471	*887*	1107	***841***
	Time	-	-	1.569	0.194	***0.126***	0.212	*0.150*
	TRR	-	-	9.6e−11	9.4e−11	9.6e−11	9.2e−11	9.7e−11
#12	**OPs**	1531	347	360	333	***311***	317	*312*
	Time	0.187	***0.044***	0.080	*0.049*	0.051	0.068	0.065
	TRR	7.6e−11	6.8e−11	9.7e−11	6.5e−11	5.9e−11	7.8e−11	9.8e−11

experience, and the proposed methods appear to be more robust than the conventional Gl-LTPMs.

We also note that the numerical results with the stopping criterion $\|R_k\|_F/\|B\|_F < 10^{-12}$ are similar to the above. However, Gl-BiCGstab(4) and Gl-GPBiCGstab(4) have a slightly larger residual gap for problems #9 and #10. We will not further discuss the residual gap in the present study, but will seek more refined algorithms of Gl-LTPMs based on [21] in the future.

6 Concluding remarks

We propose a novel global Lanczos-type method Gl-GPBiCGstab(L) for solving linear matrix equations. The original GPBiCGstab(L) can be easily extended to its global version naively, but such a method has numerical instabilities when right preconditioning is applied for solving linear systems with multiple right-hand sides. Therefore, we reconstruct Gl-GPBiCGstab(L) using alternative recursion formulas to update the iteration matrices. The resulting refined algorithm with right preconditioning has greater robustness and lower computational cost than the naive version. Moreover, the results of numerical experiments show that the refined Gl-GPBiCGstab(L) converges quickly and stably compared with other Gl-LTPMs for linear systems with multiple right-hand sides and the Sylvester equation.

Based on the above results, there are two main prospects for the proposed approach. First, we can apply the proposed approach to more difficult classes of matrix equations, such as the general coupled matrix equations including the generalized Sylvester equation [11]. Second, the proposed method can be used to discuss block Krylov subspace methods, such as the block BiCGstab(L) [23] and block GPBiCG [24] methods. Because these block-type methods are effective approaches, especially when solving linear systems with multiple right-hand sides, it is natural that a block version of GPBiCGstab(L) be developed, as noted in [6]. We have not elaborated on these points in the present work but expect to discuss them in future works.

Acknowledgements The authors would like to thank the reviewers for their careful reading and constructive comments.

Funding This study was supported by grant number JP21K11925 from the Grants-in-Aid for Scientific Research Program (KAKENHI) of the Japan Society for the Promotion of Science (JSPS).

Data availability All data generated or analyzed during this study are available in this published article or in the references.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Sonneveld, P.: CGS, a fast Lanczos-type solver for nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **10**(1), 36–52 (1989). https://doi.org/10.1137/0910004
2. van der Vorst, H.A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **13**(2), 631–644 (1992). https://doi.org/10.1137/0913035
3. Gutknecht, M.H.: Variants of BiCGSTAB for matrices with complex spectrum. SIAM J. Sci. Comput. **14**(5), 1020–1033 (1993). https://doi.org/10.1137/0914062
4. Sleijpen, G.L.G., Fokkema, D.R.: BiCGstab(L) for linear equations involving unsymmetric matrices with complex spectrum. Electron. Trans. Numer. Anal. **1**, 11–32 (1993)
5. Zhang, S.-L.: GPBi-CG: Generalized product-type methods based on Bi-CG for solving non-symmetric linear systems. SIAM J. Sci. Comput. **18**(2), 537–551 (1997). https://doi.org/10.1137/S1064827592236313
6. Aihara, K.: GPBi-CGstab(L): A Lanczos-type product method unifying Bi-CGstab(L) and GPBi-CG. Numer. Linear Algebra Appl. **27**(3), 2298 (2020). https://doi.org/10.1002/nla.2298
7. Jbilou, K., Messaoudi, A., Sadok, H.: Global FOM and GMRES algorithms for matrix equations. Appl. Numer. Math. **31**(1), 49–63 (1999). https://doi.org/10.1016/S0168-9274(98)00094-4
8. Heyouni, M.: The global Hessenberg and CMRH methods for linear systems with multiple right-hand sides. Numer. Algorithms **26**(4), 317–332 (2001). https://doi.org/10.1023/A:1016603612931
9. Jbilou, K., Sadok, H., Tinzefte, A.: Oblique projection methods for linear systems with multiple right-hand sides. Electron. Trans. Numer. Anal. **20**, 119–138 (2005)
10. Heyouni, M., Essai, A.: Matrix Krylov subspace methods for linear systems with multiple right-hand sides. Numer. Algorithms **40**(2), 137–156 (2005). https://doi.org/10.1007/s11075-005-1526-2
11. Beik, F.P.A., Salkuyeh, D.K.: On the global Krylov subspace methods for solving general coupled matrix equations. Comput. Math. Appl. **62**(12), 4605–4613 (2011). https://doi.org/10.1016/j.camwa.2011.10.043
12. Ebadi, G., Alipour, N., Vuik, C.: Deflated and augmented global Krylov subspace methods for the matrix equations. Appl. Numer. Math. **99**, 137–150 (2016). https://doi.org/10.1016/j.apnum.2015.08.010
13. Zhang, J., Dai, H.: Global CGS algorithm for linear systems with multiple right-hand sides (in Chinese). Numer. Math. A: J. Chin. Univ. **30**(4), 390–399 (2008)
14. Zhang, J., Dai, H., Zhao, J.: Generalized global conjugate gradient squared algorithm. Appl. Math. Comput. **216**(12), 3694–3706 (2010). https://doi.org/10.1016/j.amc.2010.05.026
15. Zhang, J., Dai, H.: Global GPBiCG method for complex non-Hermitian linear systems with multiple right-hand sides. Comp. Appl. Math. **35**(1), 171–185 (2016). https://doi.org/10.1007/s40314-014-0188-x
16. Wang, Y., Gu, G.: Global quasi-minimal residual method for the Sylvester equations. J. Shanghai Univ. **11**(1), 52–57 (2007). https://doi.org/10.1007/s11741-007-0109-y
17. Zhang, J., Dai, H., Zhao, J.: A new family of global methods for linear systems with multiple right-hand sides. J. Comput. Appl. Math. **236**(6), 1562–1575 (2011). https://doi.org/10.1016/j.cam.2011.09.020
18. Aihara, K., Abe, K., Ishiwata, E.: Preconditioned IDRStab algorithms for solving nonsymmetric linear systems. IAENG Int. J. Appl. Math. **45**(3), 164–174 (2015)
19. Saad, Y. Iterative Methods for Sparse Linear Systems, 2nd edn. SIAM, Philadelphia (2003)
20. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Trans. Math. Softw. **38**(1), 1–25 (2011). https://doi.org/10.1145/2049662.2049663

21. Aihara, K., Imakura, A., Morikuni, K.: Cross-interactive residual smoothing for global and block Lanczos-type solvers for linear systems with multiple right-hand sides. SIAM J. Matrix Anal. Appl. **43**(3), 1308–1330 (2022). https://doi.org/10.1137/21M1436774
22. El Guennouni, A., Jbilou, K., Riquet, A.J.: Block Krylov subspace methods for solving large Sylvester equations. Numer. Algorithms **29**(1–3), 75–96 (2002)
23. Saito, S., Tadano, H., Imakura, A.: Development of the block BiCGSTAB(ℓ) method for solving linear systems with multiple right hand sides. JSIAM Lett. **6**, 65–68 (2014). https://doi.org/10.14495/jsiaml.6.65
24. Taherian, A., Toutounian, F.: Block GPBi-CG method for solving nonsymmetric linear systems with multiple right-hand sides and its convergence analysis. Numer. Algorithms **88**(4), 1831–1850 (2021). https://doi.org/10.1007/s11075-021-01097-7