CSIAM Trans. Appl. Math.
doi: 10.4208/csiam-am.SO-2024-0053

Vol. **x**, No. x, pp. 1-28
xx 2025

Generalized Product-Type Variants of RBiCG for Solving Families of Linear Systems

Yuhao Zhong[1], Yanfei Jing[1,*], Li Xu[2] and Hao Wang[2]

[1] *School of Mathematical Sciences/Institute of Computational Science, University of Electronic Science and Technology of China, Chengdu 611731, China.*
[2] *School of Electronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.*

Received 23 November 2024; Accepted 28 July 2025

Abstract. Families of slowly changing nonsingular large sparse linear systems arise frequently in many simulation problems in science and engineering. We consider iterative solution with recycling techniques for a general case where both left-hand sides and right-hand sides of the systems change from one family to the next. We firstly develop a generalized product-type method in the framework of recycling biconjugate gradient method (RBiCG), referred to as RGPBiCG, which can also be considered as a recycling variant of GPBiCG. However, as the same situation in RBiCG stabilized method (RBiCGSTAB), the construction of recycling spaces in RGPBiCG requires expensive computational costs due to invoking other algorithms (like RBiCG) to compute approximate eigenspaces. In order to further reduce such computational costs, we alternatively form the recycling spaces in RGPBiCG with difference vectors of approximate solutions, as employed for loose GMRES (LGMRES), resulting in a more promising algorithm termed as LR-GPBiCG. Numerical experiments on both a set of academic problems and engineering simulation problems demonstrate the efficiency of our proposed algorithms.

AMS subject classifications: 65F10, 65N22, 15A06

Key words: Sequences of linear systems, subspace recycling, RBiCG, GPBiCG, difference vectors of approximate solutions.

1 Introduction

We consider solving a sequence of families of linear systems with both possibly slowly-

*Corresponding author. *Email addresses:* yanfeijing@uestc.edu.cn, 00jyfvictory@163.com (Y. F. Jing), zhongyuhao119@foxmail.com (Y. H. Zhong), lixu@uestc.edu.cn (L. Xu), haowang@uestc.edu.cn (H. Wang)

changing left-hand sides and right-hand sides of the form

$$A^{(i)}x^{(i)}=b^{(i)},\quad i=1,2,\ldots, \tag{1.1}$$

where, associated with the i-th family, the left-hand side $A^{(i)}\in\mathbb{C}^{n\times n}$ is a large sparse and nonsingular coefficient matrix, the right-hand vector $b^{(i)}\in\mathbb{C}^n$ is already given, and $x^{(i)}\in\mathbb{C}^n$ is the solution to be computed. Such systems arise in many scientific and industrial simulation analyses, including modeling fatigue and fracture of engineering components [20], computational fluid dynamics [14], radiative hydrodynamic problem [39], multi-objective optimization [17], and parametric model order reduction [8,42], etc.

Throughout the context, we consider the circumstance where both $A^{(i)}$ and $b^{(i)}$ change slowly from one family to the next, and the families are not available simultaneously. After solving the current family, one can retain partial spectral information or approximation search subspace to be re-used to speed up solutions of the subsequent families. Such technique is called "subspace recycling" [22,44]. By taking into account the matrix properties and relevant application background, the choice of recycling spaces is diverse and can be applied to various algorithms or preconditioning techniques [11,24,44].

In the past thirty years, recycling variants of state-of-the-art Krylov subspace methods, including full orthogonalization-based and short-term recurrence-based methods, have been developed. To name a few, Morgan [28] proposes the GMRES with deflated restarting (referred to as GMRES-DR), which overcomes the convergence-drag effect to the restarted GMRES [41] caused by smallest eigenvalues in magnitude by using the thick restarting technique given by Wu and Simon [48] for the Lanczos eigenvalue method. By employing harmonic Ritz vectors at the beginning of each restart, GMRES-DR makes up lost search space information accumulated during the previous cycle and improves the convergence behaviour of the restarted GMRES to be more robust and faster. See [18] for discussions on three different deflation strategies. Baker, Jessup and Manteuffel [7] propose LGMRES, which adds difference vectors of approximate solutions into the newly formed Krylov subspace within each cycle of the restarted GMRES. This approach increases the skip angle over that of restarted GMRES's residuals, and effectively prevents the alternating behavior that causes the algorithm to stagnate [6], thereby accelerating convergence rates. Inspired by the recycling idea of GMRES-DR, Parks *et al.* [38] develop the GCRO-DR algorithm under the framework of GCRO [13]. GCRO-DR additionally extends to recycle spectral information (approximate eigenvectors) between adjacent families of linear systems. Furthermore, Giraud, Jing and Xiang [19] combine IB-BGMRES-DR [2] and GCRO-DR to obtain IB-GCRO-DR for solving sequences of linear systems, where each family contains a fixing left-hand side and multiple right-hand sides.

Another promising camp involves short-term recursive algorithms. Wang, de Sturler and Paulino [47] have applied the principles of GCRO-DR to obtain RMINRES , aiming at solving a series of symmetric systems. RMINRES retains the short-term recursive scheme while substantially reducing subspace selection costs through symmetry exploitation, thereby judiciously selecting a suitable subspace for recycling. Ahuja *et al.* [1,4]

modify the BiCG [15] algorithm to obtain RBiCG, which is an attractive choice in solving the linear systems containing a pair of primary and dual systems, one with $A^{(i)}$ and the other with $A^{(i)}$ transpose conjugate. Moreover, Ahuja *et al.* [1, 3] deduce RCGS and RBiCGSTAB. These algorithms avoid the appearance of $A^{(i)}$ transpose conjugate in RBiCG and obtain accelerating convergence effect. In addition, several short-recurrence recycling algorithms [30–34] based on the theory of IDR(s) [43] have been developed for solving families of linear systems with a fixed left-hand side. Compared to full orthogonalization-based algorithms, short-term recursive algorithms require less storage, making them memory-friendly for solving large sparse linear systems.

By defining a three-term recurrence relation based on the residual polynomial of BiCG, Zhang [50] derives the generalized product-type variant of BiCG method (termed as GPBiCG), which fits algorithms such as CGS, BiCGSTAB, and BiCGSTAB2 into a more general framework. In comparison with BiCGSTAB, GPBiCG exhibits greater competitiveness when solving nonsymmetric linear systems with complex spectrum. The contribution of this paper has two folds. Firstly, a generalized product-type method (referred to as RGPBiCG) based on RBiCG is developed with the residual-minimum technique employed in GPBiCG. RGPBiCG can also be considered as a recycling variant of GPBiCG. In order to further reduce computational costs brought by the construction of recycling spaces in RGPBiCG (as the same situation in RBiCGSTAB), which invokes other algorithms (like RBiCG) to compute approximate eigenspaces, we alternatively form the recycling spaces in RGPBiCG with difference vectors of approximate solutions, as employed for LGMRES [7]. As a result, a more promising algorithm termed as LR-GPBiCG is developed.

The rest of the paper is organized as follows. The RBiCG algorithm is recalled in Section 2 as the framework for the derivation of RGPBiCG, which is presented in Section 3.1 by expressing in polynomial form the residual and direction vectors generated in RBiCG. The involved parameters are precisely given in Section 3.2. In Section 4 the construction of recycling space is discussed, and a more economical alternative is offered for our algorithm. Section 4.1 gives a brief introduction to the construction of a recycling space in RBiCG. In Section 4.2, the recycling space consisting of the difference vectors of approximate solutions is exploited to derive LR-GPBiCG. In Section 5, numerical experiments are carried out to illustrate the effectiveness of our algorithms in the solution of families of linear systems in comparison with the state-of-the-art methods as mentioned above. Finally, concluding remarks are summarized in Section 6.

Throughout the paper, the symbols $\|\cdot\|$ denotes the 2-norm of a vector, and $(\cdot,\cdot)$ denotes the Euclidean inner product of two vectors in $\mathbb{C}^n$. The superscript ^{H} denotes the transpose conjugate, the overbar $\bar{\cdot}$ indicates complex conjugation. Vectors are described by lowercase letters, matrices are described by uppercase letters, and Greek letters indicate parameters. In RBiCG (BiCG), all matrices and vectors with $\tilde{\cdot}$ are associated with the dual system with A^H. For simplicity and notational convenience, we drop in the rest of this paper the superscript j in $A^{(j)}$ and $b^{(j)}$ when considering to solve the current j-th family in the entire sequence of systems.

2 A brief view on the augmented bi-Lanczos process and recycling BiCG

In this section, we revisit the augmented bi-Lanczos process and the RBiCG algorithm [1,4]. Consider a primary system $Ax=b$ and its dual system $A^H\tilde{x}=\tilde{b}$, where $\tilde{b}$ is a random vector. Let x_0 and $\tilde{x}_0$ be respectively the initial guesses for the primary and dual linear system, with the corresponding residuals $r_0=b-Ax_0$ and $\tilde{r}_0=\tilde{b}-A^H\tilde{x}_0$, satisfying $\tilde{r}_0^H r_0\neq 0$. In the classical BiCG algorithm, the bi-Lanczos process is applied to both the primary and dual systems, resulting in the following bi-Lanczos relations

$$\begin{cases} AV_i = V_{i+1}\underline{T}_i, \\ A^H\tilde{V}_i = \tilde{V}_{i+1}\underline{\tilde{T}}_i, \end{cases} \tag{2.1}$$

where the columns of $V_i=[v_1, v_2,\ldots, v_i]$ and $\tilde{V}_i=[\tilde{v}_1, \tilde{v}_2,\ldots, \tilde{v}_i]$ are referred to as Lanczos vectors, which are respectively the basis vectors of the Krylov subspaces

$$\begin{aligned} &\mathcal{K}_i(A, r_0) \equiv \text{span}\{r_0, Ar_0,\ldots, A^{i-1}r_0\}, \\ &\tilde{\mathcal{K}}_i(A^H,\tilde{r}_0) \equiv \text{span}\{\tilde{r}_0, A^H\tilde{r}_0,\ldots, (A^H)^{i-1}\tilde{r}_0\}. \end{aligned}$$

These Lanczos vectors satisfy the bi-orthogonal relation $(v_i, \tilde{v}_j)=0$ for $i\neq j$, which implies that $V_i \perp_b \tilde{V}_i$, where $\perp_b$ denotes bi-orthogonality. Specifically, $\tilde{V}_i^H V_i$ is an identity matrix. The upper square part of $\underline{T}_i$ (and, $\underline{\tilde{T}}_i$) $\in \mathbb{C}^{(i+1)\times i}$ is a tridiagonal matrix, and its last row is $(0,\ldots,0, t_{i+1,i}(\tilde{t}_{i+1,i}))$. A pair of Lanczos vectors computed by the bi-Lanczos process satisfies the following three-term recurrence:

$$\begin{aligned} &\|\varrho_{i+1}\|_2 v_{i+1} = \varrho_{i+1} = Av_i - V_i t_i \perp_b \tilde{V}_i, \\ &\|\tilde{\varrho}_{i+1}\|_2 \tilde{v}_{i+1} = \tilde{\varrho}_{i+1} = A^H\tilde{v}_i - \tilde{V}_i\tilde{t}_i \perp_b V_i, \end{aligned} \tag{2.2}$$

where, $t_i=(0,0,\ldots,t_{i-1,i}, t_{ii})^\top$ and $\tilde{t}_i=(0,0,\ldots,\tilde{t}_{i-1,i},\tilde{t}_{ii})^\top$ are determined by the bi-orthogonality condition $V_i \perp_b \tilde{V}_i$. It should be noted that the bi-Lanczos process may encounter a situation where $\tilde{v}_i^H v_i=0$ with $v_i, \tilde{v}_i\neq 0$ at any step i, which results in a breakdown of the BiCG method. This issue can be effectively addressed by the look-ahead strategies [16].

For a sequence of a pair of primary and dual linear systems, RBiCG (as referred in Algorithm 1) is derived as an augmented bi-Lanczos algorithm [4], by incorporating the Krylov subspace recycling technology into the bi-Lanczos algorithm. To solve the j-th pair of systems, let $U\in\mathbb{C}^{n\times k}$ and $\tilde{U}\in\mathbb{C}^{n\times k}$ represent the matrices associated with the recycling spaces for the primary and the dual systems, respectively, generated during the solution process of the $(j-1)$-th pair of systems. Then calculate $C=AU, \tilde{C}=A^H\tilde{U}$, ensuring that C and $\tilde{C}$ satisfy the bi-orthogonality, and it is emphasized that $\tilde{C}^H C$ is a diagonal but not identity matrix (refer to [1, p. 35]). Utilizing these recycling spaces, the bi-orthogonality condition $V_i \perp_b \tilde{V}_i$ in the bi-Lanczos process is modified as

$$\begin{bmatrix} C & V_i \end{bmatrix} \perp_b \begin{bmatrix} \tilde{C} & \tilde{V}_i \end{bmatrix}. \tag{2.3}$$

Similar to (2.2), the $(i+1)$-th Lanczos vectors here are computed by

$$\begin{aligned} \|\varrho_{i+1}\|_2 v_{i+1} &= \varrho_{i+1} = Av_i - V_i y_i - C\rho_i, \\ \|\tilde{\varrho}_{i+1}\|_2 \tilde{v}_{i+1} &= \tilde{\varrho}_{i+1} = A^H \tilde{v}_i - \tilde{V}_i \tilde{y}_i - \tilde{C}\tilde{\rho}_i, \end{aligned} \tag{2.4}$$

where, y_i, $\tilde{y}_i$, ρ_i, and $\tilde{\rho}_i$ are determined by the bi-orthogonality condition (2.3) as follows:

$$\begin{cases} \begin{bmatrix} \tilde{C} & \tilde{V}_i \end{bmatrix}^H \varrho_{i+1} = \begin{bmatrix} \tilde{C} & \tilde{V}_i \end{bmatrix}^H (Av_i - V_i y_i - C\rho_i) = 0, \\ \begin{bmatrix} C & V_i \end{bmatrix}^H \tilde{\varrho}_{i+1} = \begin{bmatrix} C & V_i \end{bmatrix} (A^H \tilde{v}_i - \tilde{V}_i \tilde{y}_i - \tilde{C}\tilde{\rho}_i) = 0 \end{cases} \tag{2.5}$$

$$\Longrightarrow \begin{cases} y_i = \tilde{V}^H A v_i \\ \rho_i = (\tilde{C}^H C)^{-1} \tilde{C}^H A v_i, \end{cases} \quad \begin{cases} \tilde{y}_i = V^H A^H \tilde{v}_i, \\ \tilde{\rho}_i = (C^H \tilde{C})^{-1} C^H A^H \tilde{v}_i. \end{cases} \tag{2.6}$$

Then combining (2.4) with (2.6), we have the following relations in matrix form:

$$\begin{cases} (I - C\hat{C}^H) A V_i = V_{i+1} \underline{Y}_i, \\ (I - \tilde{C}\check{C}^H) A^H \tilde{V}_i = \tilde{V}_{i+1} \underline{\tilde{Y}}_i \end{cases} \tag{2.7}$$

with

$$\hat{C} = \left[\frac{\tilde{c}_1}{c_1^H \tilde{c}_1} \; \frac{\tilde{c}_2}{c_2^H \tilde{c}_2} \cdots \frac{\tilde{c}_k}{c_k^H \tilde{c}_k} \right], \quad \check{C} = \left[\frac{c_1}{\tilde{c}_1^H c_1} \; \frac{c_2}{\tilde{c}_2^H c_2} \cdots \frac{c_k}{\tilde{c}_k^H c_k} \right]. \tag{2.8}$$

Let $A_1 = (I - C\hat{C}^H)A$ and $\tilde{A}_1 = (I - \tilde{C}\check{C}^H)A^H$. Although $A_1^H \neq \tilde{A}_1$, a weak condition is provided to demonstrate that (2.7) still satisfies the three-term recurrence (refer to [4, Theorem 3.2]). Consequently, the top square part of $\underline{Y}_i$ (and $\underline{\tilde{Y}}_i$)$\in \mathbb{C}^{(i+1)\times i}$ is a tridiagonal matrix, and its last row is $(0,\ldots,\, 0,\, y_{i+1,i}\; (\tilde{y}_{i+1,i}))$. Eqs. (2.7) are referred to as the augmented bi-Lanczos relations, which correspond to a bi-Lanczos recurrence with the operators A_1 and $\tilde{A}_1$.

3 A generalized product-type variant of RBiCG

The generalized product-type variant of BiCG (GPBiCG) is a transpose-free and smoother variant of BiCG, which is suitable for solving problems involving only the primary system. For solving the families of linear systems given in (1.1), based on RBiCG, we present the generalized product-type method RGPBiCG, which can also be considered as a recycling variant of GPBiCG.

3.1 Polynomial recurrences with a recycling space

As for RBiCG, we give the polynomial representations of the residual vector and direction vector associated with the primary system as

$$r_n^{RBiCG} = R_n(A_1) r_0, \quad p_n^{RBiCG} = P_n(A_1) r_0,$$

where, $A_1=(I-C\hat{C}^H)A$, R_n and P_n denote the polynomials of degree n respectively associated with the residual r_n and the direction p_n. According to Algorithm 1, we have

$$\begin{aligned}&R_{n+1}(\lambda)=R_n(\lambda)-\alpha_n\lambda P_n(\lambda),\\&P_{n+1}(\lambda)=R_{n+1}(\lambda)+\beta_n P_n(\lambda),\quad n=0,1,2,\ldots.\end{aligned}$$

Substituting the expression of P_n in R_{n+1}, we can obtain the three-term recurrence relations for R_{n+1} as follows:

$$R_{n+1}(\lambda)=\left(1+\frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n-\alpha_n\lambda\right)R_n(\lambda)-\frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n R_{n-1}(\lambda),\quad n=1,2,\ldots,$$

where $R_0(\lambda)=1, R_1(\lambda)=(1-\alpha_0\lambda)R_0(\lambda)$.

Borrowing the construction idea of GPBiCG, the residual vector of RGPBiCG is expressed as $r_n=H_n(A_1)r_n^{RBiCG}$ by combining r_n^{RBiCG} with an auxiliary polynomial $H_n(\lambda)$ of degree n. It is remarked that $H_n(\lambda)$ plays the role of an accelerating polynomial, sharing a form similar to that of $R_n(\lambda)$ as follows:

$$H_{n+1}(\lambda)=(1+\eta_n-\zeta_n\lambda)H_n(\lambda)-\eta_n H_{n-1}(\lambda),\quad \zeta_n\neq 0,\quad n=1,2,\ldots, \tag{3.1}$$

Algorithm 1. RBiCG.

1: Given $U(C)$ and $\tilde{U}(\tilde{C})$, compute $\check{C}$ and $\hat{C}$ using (2.8). If U and $\tilde{U}$ are not available, then initialize $U,\tilde{U},\check{C}$ and $\hat{C}$ to be empty matrices.
2: The convergence tolerance and maximum iterations are respectively prescribed as *tol* and *max_it*.
3: x_{-1} and $\tilde{x}_{-1}$ are initial guesses, $r_{-1}=b-Ax_{-1}$ and $\tilde{r}_{-1}=\tilde{b}-A^H\tilde{x}_{-1}$.
4: $x_0=x_{-1}+U\hat{C}^H r_{-1}$, $\tilde{x}_0=\tilde{x}_{-1}+\tilde{U}\check{C}^H\tilde{r}_{-1}$.
5: $r_0=(I-C\hat{C}^H)r_{-1}$, $\tilde{r}_0=(I-\tilde{C}\check{C}^H)\tilde{r}_{-1}$.
6: **if** $(r_0,\tilde{r}_0)=0$, **then** re-initialize $\tilde{x}_{-1}$ to be a random vector.
7: Set $\beta_{-1}=0$, $\tilde{\beta}_{-1}=0$, $p_{-1}=0$, and $\tilde{p}_{-1}=0$.
8: **for** $n=0,1,\ldots,max\_it$
9: $\quad p_n=r_n+\beta_{n-1}p_{n-1}$, $\tilde{p}_n=\tilde{r}_n+\tilde{\beta}_{n-1}\tilde{p}_{n-1}$.
10: $\quad q_n=(I-C\hat{C}^H)Ap_n$, $\tilde{q}_n=(I-\tilde{C}\check{C}^H)A^H\tilde{p}_n$.
11: $\quad \alpha_n=\dfrac{(\tilde{r}_n,\,r_n)}{(\tilde{p}_n,\,q_n)}$, $\tilde{\alpha}_n=\bar{\alpha}_n$.
12: $\quad x_{n+1}=x_n+\alpha_n(I-U\hat{C}^HA)p_n$, $\tilde{x}_{n+1}=\tilde{x}_n+\tilde{\alpha}_n(I-\tilde{U}\check{C}^HA^H)\tilde{p}_n$.
13: $\quad r_{n+1}=r_n-\alpha_n q_n$, $\tilde{r}_{n+1}=\tilde{r}_n-\tilde{\alpha}_n\tilde{q}_n$.
14: $\quad$ **if** $\|r_{n+1}\|/\|r_0\|<tol$ and $\|\tilde{r}_{n+1}\|/\|\tilde{r}_0\|<tol$, **then** break.
15: $\quad \beta_n=\dfrac{(\tilde{r}_{n+1},\,r_{n+1})}{(\tilde{r}_n,\,r_n)}$, $\tilde{\beta}_n=\bar{\beta}_n$.
16: **end**

where, $H_0(\lambda)=1, H_1(\lambda)=(1-\zeta_0\lambda)H_0(\lambda), \zeta_n$ and η_n are parameters to be given in Theorem 3.1. To facilitate derivation, another auxiliary polynomial

$$G_n(\lambda):=\frac{[H_n(\lambda)-H_{n+1}(\lambda)]}{\zeta_n}\lambda$$

is introduced analogously to that in [50, p. 542], and the recurrence relations of H_n in (3.1) are equal to

$$\begin{aligned}&H_n(\lambda)-H_{n+1}(\lambda)=\zeta_n\lambda H_n(\lambda)+\eta_n\big(H_{n-1}(\lambda)-H_n(\lambda)\big),\\ \Rightarrow\ &\zeta_n\lambda G_n(\lambda)=\zeta_n\lambda H_n(\lambda)+\eta_n\zeta_{n-1}\lambda G_{n-1}(\lambda).\end{aligned}$$

Hence, the following recursion relations for $H_n(\lambda)$ and $G_n(\lambda)$ can be constructed as

$$\begin{aligned}&H_{n+1}(\lambda)=H_n(\lambda)-\zeta_n\lambda G_n(\lambda),\\ &G_{n+1}(\lambda)=H_{n+1}(\lambda)+\zeta_n\frac{\eta_{n+1}}{\zeta_{n+1}}G_n(\lambda),\quad n=0,1,2,\ldots,\end{aligned}$$

where $H_0(\lambda)=1, G_0(\lambda)=1$.

We follow the derivation technique in GPBiCG to obtain the following recursion relations:

$$\begin{aligned}H_{n+1}R_{n+1}&=H_nR_{n+1}-\eta_n\zeta_{n-1}\lambda G_{n-1}R_{n+1}-\zeta_n\lambda H_nR_{n+1}\\ &=H_nR_n-\alpha_n\lambda H_nP_n-\zeta_n\lambda G_nR_{n+1},\\ \zeta_{n-1}\lambda G_{n-1}R_{n+1}&=H_{n-1}R_n-H_nR_n-\alpha_n\lambda H_{n-1}P_n+\alpha_n\lambda H_nP_n,\\ H_nR_{n+1}&=H_nR_n-\alpha_n\lambda H_nP_n,\\ H_{n+1}P_{n+1}&=H_{n+1}R_{n+1}+\beta_nH_nP_n-\beta_n\zeta_n\lambda G_nP_n,\\ \zeta_n\lambda G_nP_n&=\zeta_n\lambda H_nP_n+\eta_n(H_{n-1}R_n-H_nR_n+\beta_{n-1}\zeta_{n-1}\lambda G_{n-1}P_{n-1}),\\ \zeta_nG_nR_{n+1}&=\zeta_nH_nR_n+\eta_n\zeta_{n-1}G_{n-1}R_n-\alpha_n\zeta_n\lambda G_nP_n,\\ \lambda H_nP_{n+1}&=\lambda H_nR_{n+1}+\beta_n\lambda H_nP_n.\end{aligned}\tag{3.2}$$

Then one define six auxiliary iteration vectors

$$\begin{aligned}&t_n=H_n(A_1)r_{n+1}^{RBiCG}, && y_n=\big(H_{n-1}(A_1)-H_n(A_1)\big)r_{n+1}^{RBiCG},\\ &p_n=H_n(A_1)p_n^{RBiCG}, && z_n=\zeta_nG_n(A_1)r_{n+1}^{RBiCG},\\ &w_n=A_1H_n(A_1)p_{n+1}^{RBiCG}, && u_n=\zeta_nA_1G_n(A_1)p_n^{RBiCG}.\end{aligned}\tag{3.3}$$

According to (3.2), (3.3) and $r_n=H_n(A_1)r_n^{RBiCG}=H_n(A_1)R_n(A_1)r_0^{RBiCG}$, we can obtain the recursion relations of RGPBiCG as follows:

$$r_{n+1}=t_n-\eta_ny_n-\zeta_nA_1t_n \tag{3.4}$$

$$=r_n-\alpha_nA_1p_n-A_1z_n, \tag{3.5}$$

$$y_{n+1}=t_n-r_{n+1}-\alpha_{n+1}w_n+\alpha_{n+1}A_1p_{n+1}, \tag{3.6}$$
$$t_n=r_n-\alpha_nA_1p_n, \tag{3.7}$$
$$p_{n+1}=r_{n+1}+\beta_n(p_n-u_n), \tag{3.8}$$
$$u_n=\zeta_nA_1p_n+\eta_n(t_{n-1}-r_n+\beta_{n-1}u_{n-1}), \tag{3.9}$$
$$z_n=\zeta_nr_n+\eta_nz_{n-1}-\alpha_nu_n, \tag{3.10}$$
$$w_n=A_1t_n+\beta_nA_1p_n. \tag{3.11}$$

Next, to derive the recurrence for the approximate solution update, we use the relation $A^{-1}(I-CC^H)A=I-UC^HA$. That is,

$$\begin{aligned} & r_{n+1}=b-Ax_{n+1}=r_n-\alpha_nA_1p_n-A_1z_n \\ \Rightarrow\ & Ax_{n+1}=Ax_n+\alpha_nA_1p_n+A_1z_n \\ \Rightarrow\ & x_{n+1}=x_n+\alpha_n(I-U\hat{C}^HA)p_n+(I-U\hat{C}^HA)z_n. \end{aligned}$$

3.2 Computation for iterative parameters

It is noticed that in the RBiCG, the residual vector $\tilde{r}_n^{RBiCG}$ and the direction vector $\tilde{p}_n^{RBiCG}$ in the dual system can be written as

$$\begin{cases} \tilde{r}_n^{RBiCG}=\tilde{R}_n(A_1^H)\tilde{r}_0=\left((-1)^n\prod_{i=0}^{n-1}\tilde{\alpha}_i\right)(A_1^H)^n\tilde{r}_0+\vartheta_1, \\ \tilde{p}_n^{RBiCG}=\tilde{P}_n(A_1^H)\tilde{r}_0=\left((-1)^n\prod_{i=0}^{n-1}\tilde{\alpha}_i\right)(A_1^H)^n\tilde{r}_0+\vartheta_2, \end{cases}$$

where, $\vartheta_1,\vartheta_2\in\tilde{K}_n(A_1^H,\tilde{r}_0)$. According to the orthogonalities of RBiCG, i.e. $r_n^{RBiCG}\perp\tilde{K}_n(A_1^H,\tilde{r}_0)$ and $A_1p_n^{RBiCG}\perp\tilde{K}_n(A_1^H,\tilde{r}_0)$. Then the auxiliary formulae for computing α_n in step 11 and β_n in step 15 in Algorithm 1 can be presented as

$$\begin{aligned} \alpha_n&=\frac{(\tilde{r}_n^{RBiCG},r_n^{RBiCG})}{(\tilde{p}_n^{RBiCG},A_1p_n^{RBiCG})}=\frac{((A_1^H)^n\tilde{r}_0,r_n^{RBiCG})}{((A_1^H)^n\tilde{r}_0,A_1p_n^{RBiCG})}, \\ \beta_n&=\frac{(\tilde{r}_{n+1}^{RBiCG},r_{n+1}^{RBiCG})}{(\tilde{r}_n^{RBiCG},\,r_n^{RBiCG})}=-\alpha_n\frac{((A_1^H)^{n+1}\tilde{r}_0,r_{n+1}^{RBiCG})}{((A_1^H)^n\tilde{r}_0,\,r_n^{RBiCG})}. \end{aligned} \tag{3.12}$$

Hence, the computation of these parameters does not depend on $\tilde{r}_n^{RBiCG}$ and $\tilde{p}_n^{RBiCG}$ in the iterative process. Since the coefficient of the highest-order term of $H_n(\lambda)$ is $(-1)^n\prod_{i=0}^{n-1}\zeta_i$ from (3.1), we have

$$\begin{aligned} (\tilde{r}_0,r_n)&=(\tilde{r}_0,H_n(A_1)r_n^{RBiCG})=\left((-1)^n\prod_{i=0}^{n-1}\zeta_i\right)((A_1^H)^n\tilde{r}_0,r_n^{RBiCG}), \\ (\tilde{r}_0,A_1p_n)&=(\tilde{r}_0,H_n(A_1)A_1p_n^{RBiCG})=\left((-1)^n\prod_{i=0}^{n-1}\zeta_i\right)((A_1^H)^n\tilde{r}_0,A_1p_n^{RBiCG}). \end{aligned}$$

Then from (3.12), α_n and β_n can be expressed by r_{n+1}, r_n, and p_n as follows:

$$\alpha_n = \frac{(\tilde{r}_0,\, r_n)}{(\tilde{r}_0,\, A_1 p_n)}, \quad \beta_n = \frac{\alpha_n}{\zeta_n}\frac{(\tilde{r}_0,\, r_{n+1})}{(\tilde{r}_0,\, r_n)}.$$

Theorem 3.1. *To satisfy the local minimum condition of the residual vectors at each iteration, the parameters ζ_n and η_n are computed by the following recursion relations:*

$$\begin{cases} \eta_n = 0, \\ \zeta_n = \dfrac{(A_1 t_n,\, t_n)}{(A_1 t_n,\, A_1 t_n)}, \end{cases} \quad n = 0;$$

$$\begin{cases} \eta_n = \dfrac{(A_1 t_n, A_1 t_n)(y_n, t_n) - (y_n, A_1 t_n)(A_1 t_n, t_n)}{(A_1 t_n, A_1 t_n)(y_n, y_n) - (y_n, A_1 t_n)(A_1 t_n, y_n)}, \\ \zeta_n = \dfrac{(y_n, y_n)(A_1 t_n, t_n) - (y_n, t_n)(A_1 t_n,\, y_n)}{(A_1 t_n, A_1 t_n)(y_n, y_n) - (y_n,\, A_1 t_n)(A_1 t_n, y_n)}, \end{cases} \quad n = 1,2,\ldots.$$

Proof. In order to minimize the residual 2-norm, which is the function of ζ_n and η_n as shown in (3.4), we solve the least-squares problem

$$\min \|r_{n+1}\| = \min f(\zeta_n, \eta_n) = \min_{\eta_n, \zeta_n \in \mathbb{C}} \|t_n - \eta_n y_n - \zeta_n A_1 t_n\|. \tag{3.13}$$

For $n = 0$, setting $\beta_{-1} = 0, t_{-1} = w_{-1} = u_{-1} = z_{-1} = 0$, from (3.6)-(3.11) we have

$$\begin{aligned} p_0 &= r_0, \\ y_0 &= -r_0 + \alpha_0 A_1 p_0, \\ t_0 &= r_0 - \alpha_0 A_1 p_0, \\ u_0 &= \zeta_0 A_1 p_0 - \eta_0 r_0, \\ z_0 &= \zeta_0 r_0 - \alpha_0 u_0. \end{aligned}$$

From the formulas (3.5) and (3.7), we have

$$\begin{aligned} & t_0 - \eta_0 y_0 - \zeta_0 A_1 t_0 = r_0 - \alpha_0 A_1 p_0 - A_1 z_0 \\ \Rightarrow \quad & \eta_0 y_0 + \zeta_0 A_1 t_0 = A_1 z_0 \\ \Rightarrow \quad & -\eta_0 r_0 + \eta_0 \alpha_0 A_1 p_0 + \zeta_0 A_1 r_0 - \zeta_0 \alpha_0 A_1 A_1 p_0 = \zeta_0 A_1 r_0 - \alpha_0 A_1 u_0 \\ \Rightarrow \quad & -\eta_0 r_0 + \eta_0 \alpha_0 A_1 p_0 - \zeta_0 \alpha_0 A_1 A_1 p_0 = -\zeta_0 \alpha_0 A_1 A_1 p_0 + \eta_0 \alpha_0 A_1 r_0 \\ \Rightarrow \quad & -\eta_0 r_0 = 0. \end{aligned}$$

Since $r_0 \neq 0, \eta_0 = 0$, therefore (3.13) becomes

$$\min_{\zeta_0 \in \mathbb{C}} \|t_0 - \zeta_0 A_1 t_0\|,$$

which is solved equivalently by the orthogonal condition that $t_0-\zeta_0 A_1 t_0$ be orthogonal to the subspace spanned by $A_1 t_0$, that is $(A_1 t_0)^H A_1 t_0 \zeta_0 = (A_1 t_0)^H t_0$, yielding

$$\zeta_0 = \frac{(A_1 t_0, t_0)}{(A_1 t_0, A_1 t_0)}.$$

For solving (3.13) with $n=1,2,\ldots$, analogously, based on orthogonal conditions that $(t_n-\zeta_n A_1 t_n)-\eta_n y_n$ and $(t_n-\eta_n y_n)-\zeta_n A_1 t_n$ be respectively orthogonal to the subspaces spanned by y_n and $A_1 t_n$, we have

$$\begin{aligned}
&\begin{cases}(y_n)^H y_n \eta_n = (y_n)^H (t_n-\zeta_n A_1 t_n),\\ (A_1 t_n)^H A_1 t_n \zeta_n = (A_1 t_n)^H (t_n-\eta_n y_n),\end{cases}\\
\Rightarrow\ &\begin{cases}\eta_n = \dfrac{(A_1 t_n, A_1 t_n)(y_n, t_n)-(y_n, A_1 t_n)(A_1 t_n, t_n)}{(A_1 t_n, A_1 t_n)(y_n, y_n)-(y_n, A_1 t_n)(A_1 t_n, y_n)},\\ \zeta_n = \dfrac{(y_n, y_n)(A_1 t_n, t_n)-(y_n, t_n)(A_1 t_n, y_n)}{(A_1 t_n, A_1 t_n)(y_n, y_n)-(y_n, A_1 t_n)(A_1 t_n, y_n)}.\end{cases}
\end{aligned}$$

The proof is complete. □

The pseudocode for RGPBiCG is summarized in Algorithm 2. It should be emphasized that RGPBiCG, RCGS, and RBiCGSTAB are transpose-free variants derived form RBiCG, which belong to a unified framework. Setting $\zeta_n=\alpha_n, \eta_n=(\beta_{n-1}/\alpha_{n-1})\alpha_n$ leads to $H_n=R_n$, where $r_n=H_n(A_1)r_n^{RBiCG}=R_n^2(A_1)r_0$, which precisely represents the residual when deriving RCGS from RBiCG. On the other hand, setting $\eta_n=0$ results in the residual $r_n=H_n(A_1)r_n^{RBiCG}=H_n(A_1)R_n(A_1)r_0$ with $H_{n+1}(\lambda)=(1-\zeta_n\lambda)H_n(\lambda)$, $H_0=1, \zeta_n\neq 0$, $n=1,2,\ldots$, which is the residual representation during the derivation of RBiCGSTAB from RBiCG. For detailed derivations of RCGS and RBiCGSTAB, refer to [1]. Furthermore, the selection of the recycling spaces is flexible and variable in algorithms of such kind, as will be discussed in detail in Section 4.

4 On the recycling spaces

In this section, we discuss the recycling spaces for the transpose-free variant of RBiCG. We take RGPBiCG as an example, i.e. how to construct $U(C)$ and $\tilde{U}(\tilde{C})$ as outlined in Algorithm 2. We first analyze the construction of the recycling spaces in RBiCG in Section 4.1, which is closely related to our algorithm. Subsequently, in Section 4.2, we propose a more cost-effective approach for obtaining the recycling spaces.

4.1 Analysis of the recycling space used in RBiCG

In certain problems, the presence of small eigenvalues in magnitude hampers challenge for convergence of classical Krylov subspace methods. GMRES-DR [28] recycles the har-

Algorithm 2. RGPBiCG.

Given $U(C)$ and $\tilde{U}(\tilde{C})$, compute $\hat{C}$ using (2.8).
The convergence tolerance and maximum iterations are respectively prescribed as *tol* and *max_it*.
x_{-1} is an initial guess, $r_{-1}=b-Ax_{-1}$.
$x_0=x_{-1}+U\hat{C}^H r_{-1}$, $r_0=(I-C\hat{C}^H)r_{-1}$.
$\tilde{r}_0$ is an arbitrary vector, such that $(r_0,\tilde{r}_0)\neq 0$.
Set $\beta_{-1}=0$, $t_{-1}=0$, $w_{-1}=0$, $p_{-1}=0$, $z_{-1}=0$, and $u_{-1}=0$.
for $n=0,1,\ldots,max\_it$
  $p_n=r_n+\beta_{n-1}(p_{n-1}-u_{n-1})$.
  $q_n=(I-C\hat{C}^H)Ap_n$.
  $\alpha_n=\dfrac{(\tilde{r}_0,\, r_n)}{(\tilde{r}_0,\, q_n)}$.
  $y_n=t_{n-1}-r_n-\alpha_n w_{n-1}+\alpha_n q_n$.
  $t_n=r_n-\alpha_n q_n$.
  $at_n=(I-C\hat{C}^H)At_n$.
  $\zeta_n=\dfrac{(y_n,y_n)(at_n,t_n)-(y_n,t_n)(at_n,y_n)}{(at_n,at_n)(y_n,y_n)-(y_n,at_n)(at_n,y_n)}$.
  $\eta_n=\dfrac{(at_n,at_n)(y_n,t_n)-(y_n,at_n)(at_n,t_n)}{(at_n,at_n)(y_n,y_n)-(y_n,at_n)(at_n,y_n)}$.
  (**if** $n=0$, **then** $\zeta_n=(at_n,\, t_n)/(at_n,\, at_n)$, $\eta_n=0$).
  $u_n=\zeta_n q_n+\eta_n(t_{n-1}-r_n+\beta_{n-1}u_{n-1})$.
  $z_n=\zeta_n r_n+\eta_n z_{n-1}-\alpha_n u_n$.
  $x_{n+1}=x_n+\alpha_n(I-U\hat{C}^H A)p_n+(I-U\hat{C}^H A)z_n$.
  $r_{n+1}=t_n-\eta_n y_n-\zeta_n at_n$.
  if $\|r_{n+1}\|/\|r_0\|<tol$, **then** break.
  $\beta_n=\dfrac{\alpha_n}{\zeta_n}\dfrac{(\tilde{r}_0,r_{n+1})}{(\tilde{r}_0,\, r_n)}$.
  $w_n=at_n+\beta_n q_n$.
end

monic Ritz vectors to deflate the eigenvalues of smallest magnitude, markedly enhancing both convergence and stability, and is well-known as a deflation-based method. Although GMRES-DR is designed to solve a single linear system, Parks *et al.* [38] extend GMRES-DR into GCRO-DR under the framework of GCRO, making it an effective algorithm for solving sequences of linear systems. Similarly, it has been shown in GCRO-DR [38] that constructing a recycling space from an approximate invariant subspace of A, which consists of harmonic Ritz vectors, is an effective strategy. Additionally, a class of deflation strategies based on singular value decomposition (SVD) has been employed to construct recycling spaces [12, 45].

RBiCG [1] also generates the recycling spaces with harmonic Ritz vectors to approximate invariant subspaces corresponding to small eigenvalues in magnitude. However, compared with GCRO-DR, the implementation of RBiCG has the following differences:

- RBiCG involves the simultaneous solution of a primary system and its dual system. Consequently, the recycling spaces are introduced in pairs (as shown in Section 2). This necessitates the computation of the approximate eigenvectors of A and A^H (left and right approximate eigenvectors of A).
- As a short-term recursive algorithm, restart mechanism is not suited for RBiCG. The harmonic Ritz vectors must be computed from $\underline{Y}_i$ and $\underline{\tilde{Y}}_i$ in (2.7), which requires to explicitly store all Lanczos vectors for the calculation of the harmonic Ritz vectors.

To avoid such storage issues, RBiCG organizes the algorithm's execution into cycles, where each cycle consists of s iteration steps, indexed by $\epsilon = 1,2,\ldots$. The matrices associated with recycling spaces are updated at the end of each cycle and denoted as U^ϵ and $\tilde{U}^\epsilon$. It is important to note that the recycling matrices U and $\tilde{U}$ used in RBiCG are derived from the solution process of the previous pair of systems. The updated U^ϵ and $\tilde{U}^\epsilon$ during the iteration are not directly applied to the current pair of systems. The last update of U^ϵ and $\tilde{U}^\epsilon$ will be used as the recycling matrices U and $\tilde{U}$ for the next pair of systems. Suppose that the bi-orthogonal bases matrices $V_s^m = [v_{(m-1)s+1}, v_{(m-1)s+2}, \ldots, v_{ms}]$ and $\tilde{V}_s^m = [\tilde{v}_{(m-1)s+1}, \tilde{v}_{(m-1)s+2}, \ldots, \tilde{v}_{ms}]$ are generated at the m-th cycle. Then U^m and $\tilde{U}^m$ are obtained from V_s^m ($\tilde{V}_s^m$), U^{m-1} ($\tilde{U}^{m-1}$), and U ($\tilde{U}$), according to the definition of the harmonic Ritz pair [37]. The columns of the recycling matrices U^m and $\tilde{U}^m$ in RBiCG satisfy the condition

$$\begin{cases} (Au^m - \lambda u^m) \perp \text{range}\big(A^H\big[\tilde{U}^{m-1}, \tilde{V}_s^m\big]\big), \\ (A^H\tilde{u}^m - \mu\tilde{u}^m) \perp \text{range}\big(A\big[U^{m-1}, V_s^m\big]\big), \end{cases} \tag{4.1}$$

where, $u^m \in \text{span}\{U^{m-1}, V_s^m\}$, $\tilde{u}^m \in \text{span}\{\tilde{U}^{m-1}, \tilde{V}_s^m\}$. For more detailed derivation process, refer to [1, Chapter 3]. In the derivation of transpose-free variants of RBiCG, the recycling spaces remain unchanged. In fact, RCGS, RBiCGSTAB, and RGPBiCG still utilize the approximate eigensubspace of the coefficient matrix to construct the recycling spaces, which cannot be directly obtained from their own iterations. However, this problem can be addressed by incorporating other algorithms. Ahuja adopts a strategy that the recycling space used in RBiCGSTAB is generated by RBiCG. When the coefficient matrix A changes, RBiCG is invoked again to update the recycling space. Additionally, Amritkar *et al.* [5] employs RGCROT to generate only one recycling space for RBiCGSTAB, which shows that RBiCGSTAB does not necessarily require to recycle two spaces.

Repeatedly invoking other algorithms to obtain approximate eigenvectors naturally incurs additional computational cost and storage. Alternatively, we use information from approximate solutions to construct the recycling spaces. This idea has been utilized in LGMRES [7], which at each restart adds difference vectors of approximate solutions,

thereby preventing stagnation and accelerating convergence. Moreover, Kilmer *et al.* [24] use the GCRO algorithm to explore and analyze two subspace recycling strategies: One based on recycling approximate invariant subspaces, and another on recycling subspaces from previous solutions. For more related work on recycling information from approximate solutions, please refer to [23,30,36,44,49].

Furthermore, as a short-term recursive algorithm, the transpose-free variants of BiCG can explicitly generate approximate solutions, residuals, and direction vectors at each iteration. This allows us to obtain the approximate solutions without much effort. In the next section, we describe how to construct recycling spaces from approximate solutions.

4.2 An alternative construction of the recycling space

In GPBiCG [50], the approximate solution is updated as in the form of $x_{n+1}=x_n+\triangle x_n$ at each iteration. Assuming that GPBiCG converges after n_{final} iterations, we have $X=[x_0,x_1,x_2,\ldots,x_{n_{final}}]$. We denote the difference vector of the approximate solutions at interval δ_1 as $\mu_l^{\delta_1}$, that is

$$\mu_l^{\delta_1}=x_{l\delta_2}-x_{l\delta_2-\delta_1},\quad l\delta_2\leq n_{final}. \tag{4.2}$$

Next, by storing a $\mu_l^{\delta_1}$ after every δ_2 iterations, we construct a k-column matrix as follows:

$$\begin{aligned}&\mathbb{U}=\left[\mu_1^{\delta_1},\mu_2^{\delta_1},\ldots,\mu_k^{\delta_1}\right]=[u_1,u_2,\ldots,u_k],\\&u_i=\mu_i^{\delta_1}=x_{i\delta_2}-x_{i\delta_2-\delta_1},\quad i=1,2,\ldots,k,\end{aligned} \tag{4.3}$$

where $\delta_1\leq\delta_2$, $k\delta_2\leq n_{final}$. Here, we can control the column number k of $\mathbb{U}$ by changing δ_2. In the numerical experiments, we do not need to explicitly store each approximate solution; it is sufficient to store the corresponding difference vectors at the preset intervals. Moreover, δ_1 and δ_2 are empirical parameters. In this paper, unless otherwise specified, we set $\delta_1=1$. It is worth noting that if the algorithm converges as the iterations proceed, the approximate solution gradually approaches the true solution, causing $\mu_l^{\delta_1}$ to possibly approach the zero vector. One can increase δ_1 to avoid singularities. The choice of δ_2 will be discussed in Case 2 of Example 3 of Section 5. Therefore, we can achieve subspace recycling by carrying over $\mathbb{U}$ from the j-th system to the $(j+1)$-th system. To satisfy the relations

$$AU=C,\quad C^HC=I,$$

we implement

$$\begin{aligned}&[Q,R]=A^{(j+1)}\mathbb{U}\quad\text{by reduced } QR\text{-factorization},\\&C=Q,\\&U=\mathbb{U}R^{-1},\end{aligned} \tag{4.4}$$

where $Q\in\mathbb{C}^{n\times k}$ and $R\in\mathbb{C}^{k\times k}$. It is noticed that the collected $\mathbb{U}$ may become singular, leading to the matrices U and C in (4.4) not to satisfy $AU=C$. To address this issue,

we could employ QR-decomposition to extract the linearly independent columns of $\mathbb{U}$ before computing U and C. Here we associate the underlying Krylov subspace methods with the prefix "LR" to indicate that the resulting variant employs the recycling technique with the corresponding U as described herein. And the prefix "R" indicates that the recycling space [1,3] is calculated by RBiCG [3], which is related to approximate eigenvectors corresponding to small eigenvalues in magnitude.

For u_i, we have

$$\begin{aligned} &u_i = x_{i\delta_2} - x_{i\delta_2-1}, \\ &Au_i = r_{i\delta_2-1} - r_{i\delta_2} = (H_{i\delta_2-1}R_{i\delta_2-1} - H_{i\delta_2}R_{i\delta_2})(A)r_0 = \mathcal{U}_{2i\delta_2}(A)r_0, \quad i = 1,2,\ldots,k, \end{aligned}$$

where $\mathcal{U}_{2i\delta_2}(A)$ denotes a polynomial of degree $2i\delta_2$ for the operator A. Therefore, the matrix U associated with recycling space LR contains partial information of the approximation search subspace generated during the solution process of the current system. When the coefficient matrix A and the right-hand side b change slowly, the generated Krylov subspace also changes slowly. Thus, maintaining orthogonality within U during subsequent iterations for solving the next system can accelerate the convergence rate. The pseudocode of the LR-GPBiCG is summarized as in Algorithm 3. Furthermore, as the first update of the residual in each iteration (in Algorithms 2 and 3), t_n can serve as a basis for convergence judgment [46], which can save one matrix-vector multiplication. However, our numerical experiments employ a full update of the residuals at each iteration to stabilize the convergence and obtain the complete $\mu_l^{\delta_1}$.

Similar to the presentation of LR-GPBiCG in Algorithm 3, the recycling space introduced in this section can also be applied to RBiCGSTAB to obtain another variant referred to as LR-BiCGSTAB. We will omit the detailed presentation of LR-BiCGSTAB here, but will use it directly in the numerical experiments in the next section.

5 Numerical experiments

In this section, we numerically investigate the algorithmic performance on both academic and engineering simulation problems. The first three examples fix a single linear system as the ideal case for verifying subspace recycling techniques. We adopt the strategy in GCRO-DR [38] to solve the same system twice and obtain the recycling space in the process of the first solution. The last three examples are from engineering simulation to illustrate the effectiveness of our methods. Example 4 is on a linear system with multiple right-hand sides discretized by the time-harmonic electromagnetic scattering problem, and the last two are respectively from the lattice quantum chromodynamics (QCD) problem and the fatigue and fracture of engineering components (FFEC) model problem. In the following examples, for GCRO-DR(m,k), m is the restart parameter, and k is the dimension of the recycling space. And for RBiCGSTAB(k), RGPBiCG(k), LR-BiCGSTAB(k) and LR-GPBiCG(k), k is the dimension of the recycling space.

Algorithm 3. LR-GPBiCG.

Given $\mathbb{U}$, if $\mathbb{U}$ is not available, then turn to use GPBiCG to solve the current system and calculate $\mathbb{U}$.
Let $[Q, R]$ be obtained by the reduced QR-factorization of $A\mathbb{U}$.
The convergence tolerance and maximum iterations are respectively prescribed as *tol* and *max_it*.
$C = Q$.
$U = \mathbb{U}R^{-1}$.
x_{-1} is an initial guess, $r_{-1} = b - Ax_{-1}$.
$x_0 = x_{-1} + UC^H r_{-1}$.
$r_0 = r_{-1} - CC^H r_{-1}$.
$\tilde{r}_0$ is an arbitrary vector such that $(\tilde{r}_0, r_0) \neq 0$, e.g. $\tilde{r}_0 = r_0$.
Set $\beta_{-1} = 0$, $t_{-1} = 0$, $w_{-1} = 0$, $p_{-1} = 0$, $z_{-1} = 0$, and $u_{-1} = 0$.
$rho = (\tilde{r}_0, r_0)$, $\chi = 0$.
for $n = 0,1,\ldots,max\_it$
    $p_n = r_n + \beta_{n-1}(p_{n-1} - u_{n-1})$.
    $q_n = (I - CC^H)Ap_n$.
    $\alpha_n = \dfrac{rho}{(\tilde{r}_0, q_n)}$.
    $y_n = t_{n-1} - r_n - \alpha_n w_{n-1} + \alpha_n q_n$.
    $t_n = r_n - \alpha_n q_n$.
    $at_n = (I - CC^H)At_n$.
    $\zeta_n = \dfrac{(y_n, y_n)(at_n, t_n) - (y_n, t_n)(at_n, y_n)}{(at_n, at_n)(y_n, y_n) - (y_n, at_n)(at_n, y_n)}$.
    $\eta_n = \dfrac{(at_n, at_n)(y_n, t_n) - (y_n, at_n)(at_n, t_n)}{(at_n, at_n)(y_n, y_n) - (y_n, at_n)(at_n, y_n)}$.
    (**if** $n = 0$, **then** $\zeta_n = (at_n, t_n)/(at_n, at_n)$, $\eta_n = 0$).
    $u_n = \zeta_n q_n + \eta_n(t_{n-1} - r_n + \beta_{n-1}u_{n-1})$.
    $z_n = \zeta_n r_n + \eta_n z_{n-1} - \alpha_n u_n$.
    $\chi = \chi + \alpha_n p_n + z_n$.
    $r_{n+1} = t_n - \eta_n y_n - \zeta_n at_n$.
    if $\|r_{n+1}\|/\|r_0\| < tol$, **then** break.
    $\beta_n = \dfrac{\alpha_n}{\zeta_n}\dfrac{(\tilde{r}_0, r_{n+1})}{rho}$.
    $rho = (\tilde{r}_0, r_{n+1})$.
    $w_n = at_n + \beta_n q_n$.
end
$x_{n+1} = x_0 + (I - UC^H A)\chi$.

All the numerical experiments are performed in double precision floating point arithmetic in MATLAB of version 2019A on a personal computer with an Intel(R) Core(TM) i7-10700 CPU 2.90GHz, 16.0GB of RAM. The stopping criterion is that the relative residual norm is smaller than a prescribed convergence tolerance (i.e. $\|r_n\|/\|r_0\| < tol$), or the iterations exceed a prescribed maximum number (max_it). In our experiments, all solvers take a zero vector as the initial guess, except for Case 1 in Example 3 with a vector of all ones; the default settings are $tol = 10^{-6}$ and $max\_it = 2000$ for all examples, except for Example 2 with $tol = 10^{-8}$ and Example 5 with $tol = 10^{-10}$; no preconditioner is used except for Case 2 in Example 3, and Examples 4 and 5. Numerical results are reported in terms of iterations (referred to as "iters") and CPU computing time in seconds (referred to as CPU). It is remarked that, at each iteration, two matrix-vector products (mvps) are eamied out in all solvers except for GCRO-DR with one mvps at each iteration.

5.1 Example 1 with bidiagonal matrices

The coefficient matrices in this example are bidiagonal matrices, and all entries of the right-hand side are set to be one. We consider two academic cases.

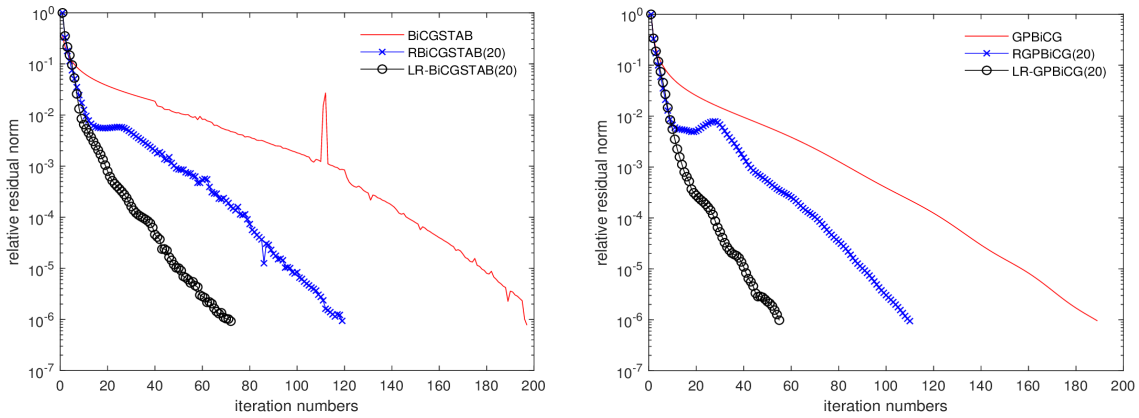
5.1.1 Case 1

We choose a bidiagonal matrix of order 4000 [27]. The elements on each secondary diagonal are 0.1, and the elements on the main diagonal are $1,2,\ldots,3999,4000$. The first experiment is to compare the effect of the recycling spaces associated with R and LR variants as stated in Section 4.2. It is remarked that the construction of the recycling space with LR is directly from the difference vectors of approximate solutions generated during GPBiCG (no additional calculation is required) while that with R is calculated by RBiCG. The second experiment is to compare LR-GPBiCG and LR-BiCGSTAB. We set the dimension of the recycling space in R and LR both as $k = 20$.

The convergence histories are displayed in Fig. 1 and the numerical results are presented in Table 1. It is observed that the LR recycling technique accelerates both BiCGSTAB and GPBiCG, and LR accelerates better than R in this example. LR-GPBiCG(20) converges slightly faster than LR-BiCGSTAB(20) in terms of iterations.

Table 1: Numerical results of recycling GPBiCG and BiCGSTAB associated with LR and R for Case 1 in Example 1.

Algorithm	Iters	Acceleration ratio
BiCGSTAB	197	
RBiCGSTAB(20)	119	39.59%
LR-BiCGSTAB(20)	72	63.45%
GPBiCG	189	
RGPBiCG(20)	110	41.80%
LR-GPBiCG(20)	55	70.90%



(a) (b)

Figure 1: (a): Convergence curves of BiCGSTAB, RBiCGSTAB(20), and LR-BiCGSTAB(20); (b): Convergence curves of GPBiCG, RGPBiCG(20), and LR-GPBiCG(20) for Case 1 of Example 1.

5.1.2 Case 2

We consider a matrix with smaller eigenvalues. The matrix is still bidiagonal but with entries $0.01, 0.02, 0.1, 0.2, 1, 2, \ldots, 1995, 1996$ on the main diagonal and 1's on the superdiagonal [28]. In this case, we test basic algorithms BiCGSTAB, GPBiCG and their recycling variants RBiCGSTAB, RGPBiCG, LR-BiCGSTAB, LR-GPBiCG. The size of recycling space (LR or R) is set to be $k=4$. The convergence curves are shown in Fig. 2. In cases where the system has small eigenvalues in magnitude, the recycling space R offers substantial benefits, not only by effectively accelerating convergence but also by enhancing the stability of GPBiCG and BiCGSTAB, as evidenced by the performance curves of RBiCGSTAB(4) and RGPBiCG(4). While our recycling space LR also contributes to convergence acceleration, it is less competitive compared to recycling space R in such cases.

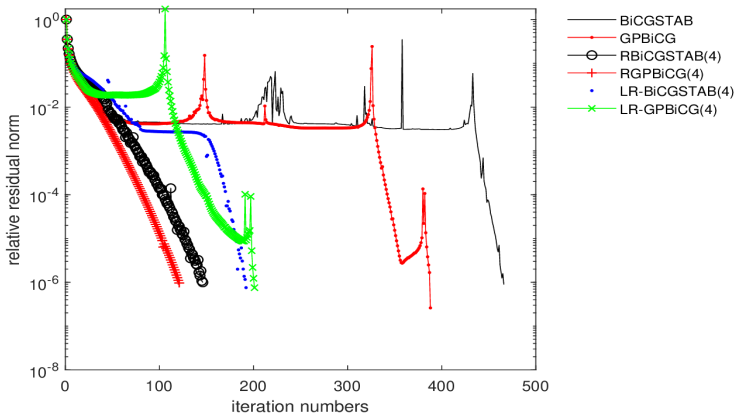


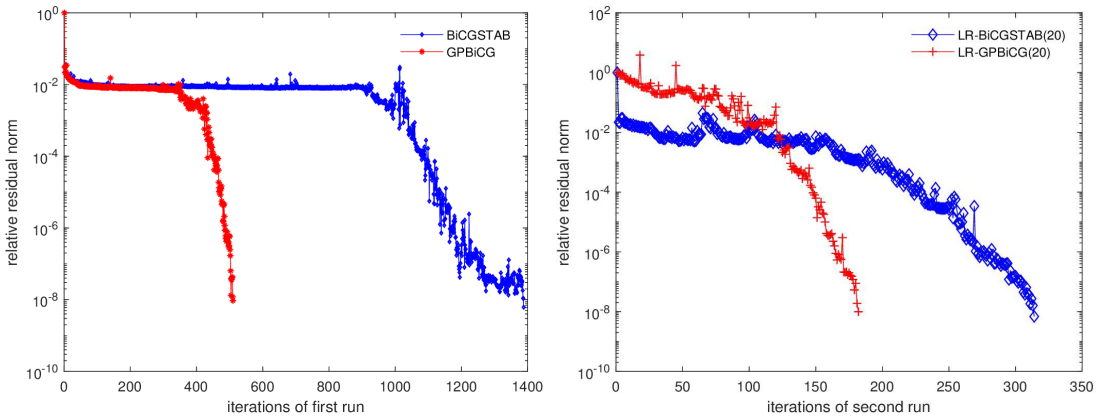
Figure 2: Convergence curves for Case 2 of Example 1.

5.2 Example 2 with a Toeplitz matrix

We consider a complex Toeplitz matrix as used in [50] of order 200 with a parameter γ

$$A=\begin{pmatrix} 4 & 0 & 1 & 0.7 & & \\ \gamma i & 4 & 0 & 1 & 0.7 & \\ & \gamma i & 4 & 0 & 1 & \ddots \\ & & \gamma i & 4 & 0 & \ddots \\ & & & \gamma i & 4 & \ddots \\ & & & & \ddots & \ddots \end{pmatrix}.$$

The parameter γ is set to be $\gamma=3.79$, and the right-hand side has all entries $1i$. The dimension of the recycling space with LR is set to be $k=20$. It is observed from Fig. 3 that GPBiCG converges faster than BiCGSTAB for this nonsymmetric linear system with complex spectrum, and so does LR-GPBiCG in comparison with LR-BiCGSTAB.



(a) (b)

Figure 3: (a): Comparison of convergence curves for BiCGSTAB and GPBiCG; (b): Comparison of convergence curves for LR-BiCGSTAB(20) and LR-GPBiCG(20) in Example 2.

5.3 Example 3 with a matrix from a convection-diffusion problem

In this example, we solve two linear systems derived from the discretization of convection-diffusion problems.

5.3.1 Case 1

We consider the vertex centered finite volume discretization of the partial differential equation (PDE) [3] as

$$-(u_x)_x-(u_y)_y+10u_x-10u_y=0.$$

Using the boundary conditions

$$u_{south} = u_{west} = 1,$$
$$u_{north} = u_{east} = 0,$$

we solve the problem on a 82×82 grid unit square, resulting in a 6400×6400 linear system. We compare BiCGSTAB, GPBiCG, RBiCGSTAB, RGPBiCG, LR-BiCGSTAB, and LR-GPBiCG. By invoking RBiCG, we generate ten approximate left eigenvectors and ten approximate right eigenvectors of the coefficient matrix, which are used to construct the recycling space R for RBiCGSTAB and RGPBiCG. For LR-GPBiCG and LRBiCGSTAB, we set up a recycling space LR of dimension 20. Table 2 presents the number of matrix-vector products required by each algorithm, and the convergence curves are shown in Fig. 4.

From Fig. 4, it is observed that both of the recycling spaces R and LR significantly help to improve the convergence behaviour of BiCGSTAB and GPBiCG, and the recycling space LR leads to a relatively smoother convergence compared to R. Moreover, LR-BiCGSTAB and LR-GPBiCG converge faster than RBiCGSTAB and RGPBiCG respectively, and LR-GPBiCG performs the best to converge after 50 iterations. This indicates that the recycling space LR offers greater advantages in accelerating the solution of convection-diffusion problems.

Table 2: The number of matrix-vector products required by each algorithm for Case 1 in Example 3.

Algorithm	Matrix-vector products
BiCGSTAB	286
RBiCGSTAB(20)	174
LR-BiCGSTAB(20)	126
GPBiCG	274
RGPBiCG(20)	178
LR-GPBiCG(20)	100

5.3.2 Case 2

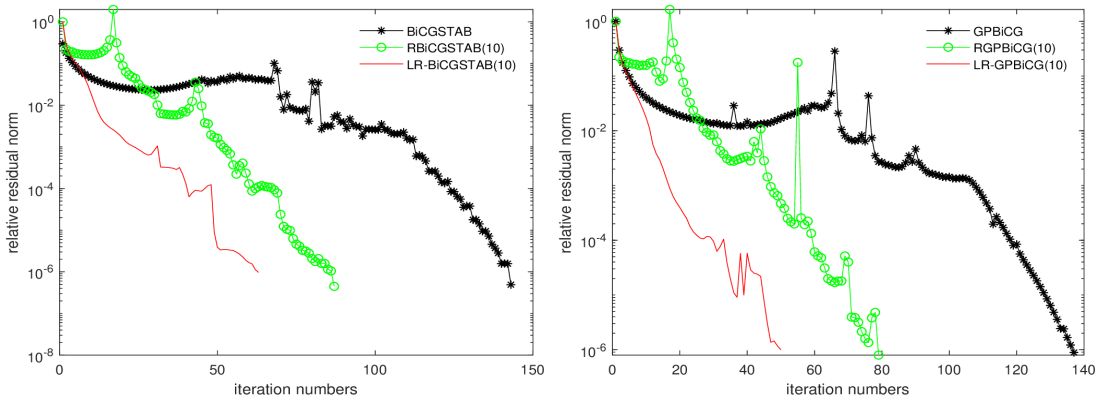
Here we consider the finite difference discretization of the PDE [46] as

$$-(au_x)_x - (au_y)_y + c(x,y)u_x = F$$

on a square $[0,1] \times [0,1]$ with Dirichlet boundary conditions

$$u(x,0) = u(0,y) = u(1,y) = 1,$$
$$u(x,1) = 0,$$

where $c(x,y) = 2e^{2(x^2+y^2)}$. The function a is defined as in Fig. 5, and $F = 0$ everywhere, except for the small subsquare with a side length 16/128 (from 56/128 to 72/128) in the centre, $F = 100$. We generate a $127^2 \times 127^2$ matrix using the central difference scheme with



(a) (b)

Figure 4: (a): Convergence curves of BiCGSTAB, RBiCGSTAB(10) and LR-BiCGSTAB(10); (b): Convergence curves of GPBiCG, RGPBiCG(10) and LR-GPBiCG(10) for Case 1 in Example 3.

the grid width being $h=1/128$ in two directions. We choose ILU-factorization [40] as the right preconditioner with a drop tolerance being 0.1.

The convergence curves are shown in Fig. 6. The standard GPBiCG method, which does not utilize recycling spaces, required 347 iterations and 45.87 seconds of CPU time to converge. In contrast, the LR-GPBiCG method, with recycling space dimensions of 5,10,20,30, and 40, demonstrated significant improvements in both convergence rate and computational efficiency. Specifically, as the dimension of the recycling space increased from 5 to 40, the number of iterations decreased from 194 to 79, and the corresponding CPU time reduced from 25.63 seconds to 10.44 seconds. Moreover, when the dimension of the recycling space exceeds 10, there is indeed a slight reduction in both the iters count and CPU, but the improvement is not as significant as the reduction observed when increasing the dimension from 5 to 10. This indicates that, although a larger recycling space can improve performance, the benefit may weaken after exceeding a certain threshold. Moreover, a larger recycling space occupies more memory, hence in this situation, the results of LRGPBiCG(10) are more advantageous. The choice of the dimension of the recycling space is problem-dependent and deserves to be studied further.

Additionally, in all the experiments conducted in this paper, the LR-type recycling spaces used are obtained from the entire approximate solution space (i.e. derived from $X=[x_0,x_1,x_2,\ldots,x_{n_{final}}]$). Here, we fix the dimension of the recycling space at 10 and reduce the extracted range as $X=[x_0,x_1,x_2,\ldots,x_{n_{final}/2}]$. The corresponding results are presented as LR-GPBiCG*(10), as shown in Fig. 7. Clearly, since LR-GPBiCG(10) covers a broader region of approximate solution information, it converges faster. However, LR-GPBiCG*(10) also exhibits favorable convergence behavior, which provides an insight: We can construct the recycling space within a smaller approximate solution space rather than requiring the previous system to converge completely. Although this sacrifices some of the additional accelerated convergence, it requires less storage and computational effort. The numerical results are shown in Table 3.

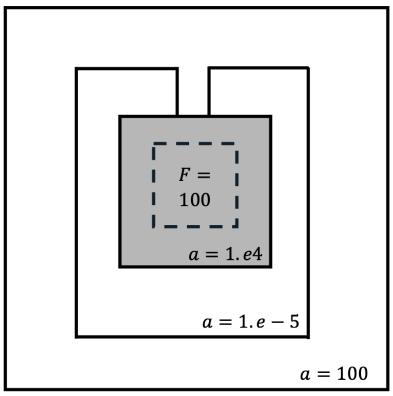


Figure 5: Coefficients of the PDE for Case 2 in Example 3.

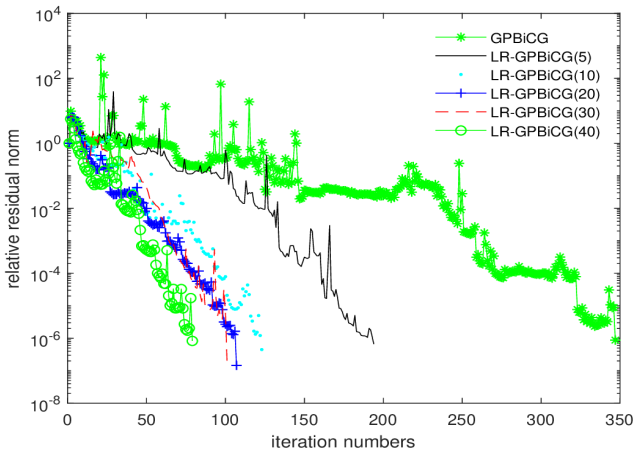


Figure 6: Convergence curves of LR-GPBiCG with recycling spaces of varying dimensions for Case 2 in Example 3.

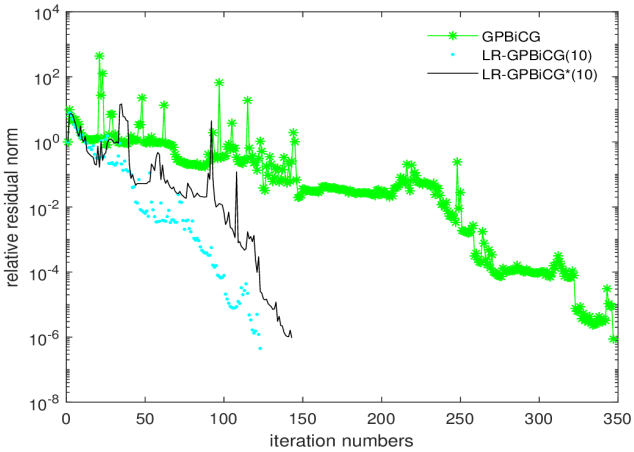


Figure 7: Convergence curves of LR-GPBiCG(10) and LR-GPBiCG*(10) for Case 2 in Example 3.

Table 3: Numerical results on LR-GPBiCG with recycling spaces of different dimensions for Case 2 in Example 3.

Algorithm	Dimension (LR)	Iters	CPU(s)
GPBiCG		347	45.87
LR-GPBiCG	5	194	25.63
	10	123	16.29
	20	107	14.06
	30	101	13.37
	40	79	10.44
LR-GPBiCG*	10	143	18.79

5.4 Example 4 from time-harmonic electromagnetic scattering simulation

In this section, we consider the solution of a sequence of linear systems arising from radar cross section (RCS) calculations [35] in time-harmonic electromagnetic scattering problems [25] by handling the weak form [51] of electromagnetic field vector wave equation with the finite element method. For monostatic RCS, the radar transmitter and receiver are at the same location. In order to compute the monostatic RCS from different directions, a series of incident plane wave excitations different from the monostatic RCS directions should be set up, so that there are a number of right-hand sides associated with the linear system. Individually solving such linear systems would result in inefficiency. We investigate our LR-GPBiCG in comparison with relevant single solvers to address such linear systems with multiple right-hand sides available in sequence as in (1.1). Refer to [2, 10, 19, 21, 29] and the references therein on block Krylov subspace methods for simultaneously solving these systems if all the right-hand sides are available at the same time.

We consider the electromagnetic scattering analysis of X-51 hypersonic aircraft [25] taking the frequency of the plane wave excitations as 0.2 GHz, whose model is shown in Fig. 8. The directions of the plane wave are 0 degree in the ϕ direction and $0\sim360$ degrees in the θ direction with an 3 degree interval. The polarization direction of the plane wave electric field is ϕ direction and has a magnitude of 1. This leads to a sequence of linear systems of order 120450 with slowly changing right-hand sides. Six linear systems corresponding to 0 degree to 15 degree with a 3 degree interval are chosen for numerical verification.

We apply ILU-factorization with a drop tolerance 3×10^{-4} for right preconditioning after matrix reordering (symmetric approximate minimum degree permutation). After solving the first system with GPBiCG, we obtain a recycling space U of dimension $k=14$ (the dimension can be flexibly tuned by the difference vectors of the approximate solutions), then LR-GPBiCG(14) is used to solve the remaining five systems. Numerical results with counterpart solvers are reported in Table 4. It is observed that for solving the subsequent five systems with LR-GPBiCG(14), the introduction of the recycling space achieves about 52.64% reduction of matrix-vector products and about 69.06% reduction

Table 4: Numerical results on LR-GPBiCG with recycling spaces of different dimensions for Case 2 in Example 3.

Method		System					
		1	2	3	4	5	6
GPBiCG	Iters	435	470	483	512	524	490
	CPU	204.31	217.35	224.79	238.55	243.56	226.97
LR-GPBiCG(14)	Iters	435	237	256	233	221	227
	CPU	204.31	71.86	77.86	70.59	67.16	68.69
GCRO-DR(40,14)	Iters	1097	1403	1237	987	1586	1205
	CPU	207.98	264.58	234.16	187.67	300.11	228.03
RBiCG & RBiCGSTAB(14)	Iters	255	–	–	–	–	–
	CPU	152.88					

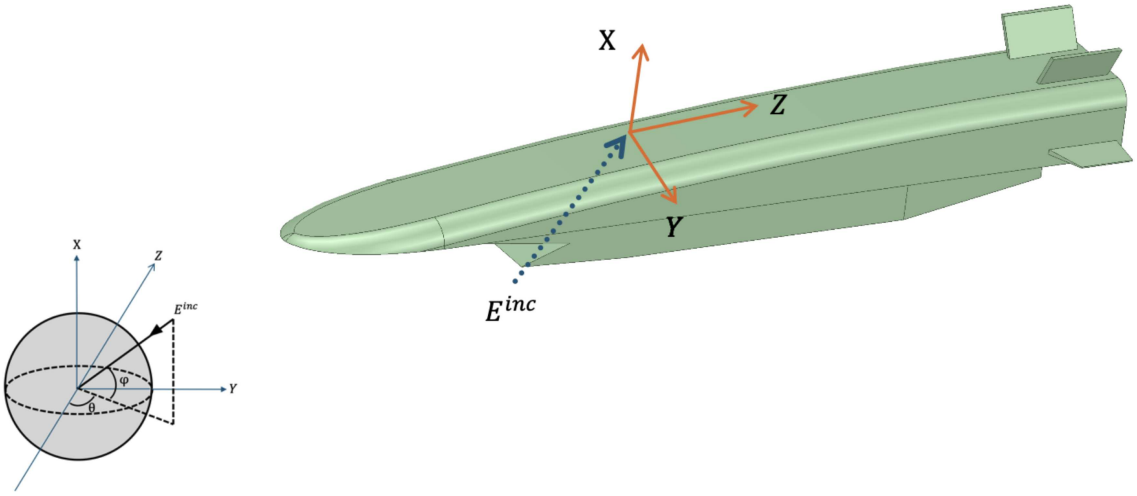


Figure 8: The model of X-51 class hypersonic vehicle.

of CPU computing time, as compared with GPBiCG for solving each system separately. "RBiCG & RBiCGSTAB(14)" indicates that RBiCG is used to solve the first system, and then RBiCGSTAB(14) is performed to solve subsequent systems. The size of recycling spaces in all algorithms are set as k=14. RBiCGSTAB(14) did not converge to the targeted accuracy within 2000 iterations for the rest systems, as indicated by the symbol "–". In this example, LR-GPBiCG outperforms both RBiCGSTAB [3] and GCRO-DR [38] in both terms of CPU and mvps.

5.5 Example with matrices from QCD

This example tests a sequence of linear systems from quantum chromodynamics problem in physics. For this model problem, we borrow the matrix conf5.0 00l4x4.1000.mtx of size

3072×3072 as D from the Matrix Market website at NIST [9], which was submitted by Medeke [26]. The families of linear systems are constructed as $(I-\kappa D)x=b$ with $0\leq\kappa<\kappa_c$. We set $\kappa=0.202$, $\kappa_c=0.205$ and use the MATLAB function `linspace`(0.202,0.205,10) to generate the parameters $\kappa^{(i)}$ $(i=1,2,\ldots,10)$ for a sequence of matrices. It is noticed that those matrices have the same eigenvectors associated with shifted eigenvalues. We set 10 consecutive right-hand sides as the first 10 Cartesian basis vectors. We employ GPBiCG to solve the first linear system and obtain a recycling space U with dimension $k=20$, and then use LR-GPBiCG(20) to solve the rest nine linear systems. We apply ILU-factorization with a drop tolerance 0.08 for right preconditioning. Accelerating effect of LR-GPBiCG(20) in comparison with GPBiCG can be again observed from Fig. 9.

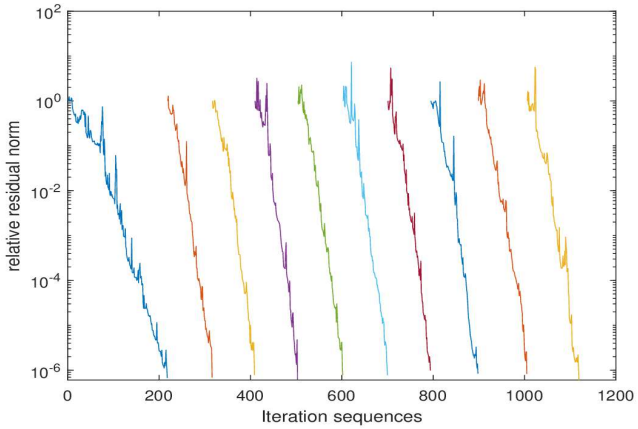


Figure 9: Convergence curves for 10 consecutive left-hand sides and right-hand sides for a model quantum chromodynamics problem with LR-GPBiCG(20).

5.6 Example with matrices from FFEC

In this section, we compare our LR-GPBiCG with GCRO-DR [38] for solving a sequence of linear systems discretized from a finite element fracture mechanics problem in the field of fatigue and fracture of engineering components (denoted as FFEC collection). In the process of capturing the fracture progress, there are more than 2000 linear systems of order 3988 to be solved, of which the 151 linear systems from 400 to 550 are typical subsets of the fracture procession of many cohesive elements break. We borrow the first ten linear systems numbered 400-409 for numerical experiments.

The first system is solved by GPBiCG (no recycling space is available), and the recycling space generated is applied to accelerate the solution of the subsequent systems through LR-GPBiCG. Convergence curves are shown in Fig. 10 to compare the number of iterations required by LR-GPBiCG(25) and GCRO-DR(50,25) for solving all the ten systems.

With the slowly change of linear systems, the acceleration ratio of LR-GPBiCG(25) is reduced from 59.08% to 58.53%, which is within an acceptable range. At each iteration,

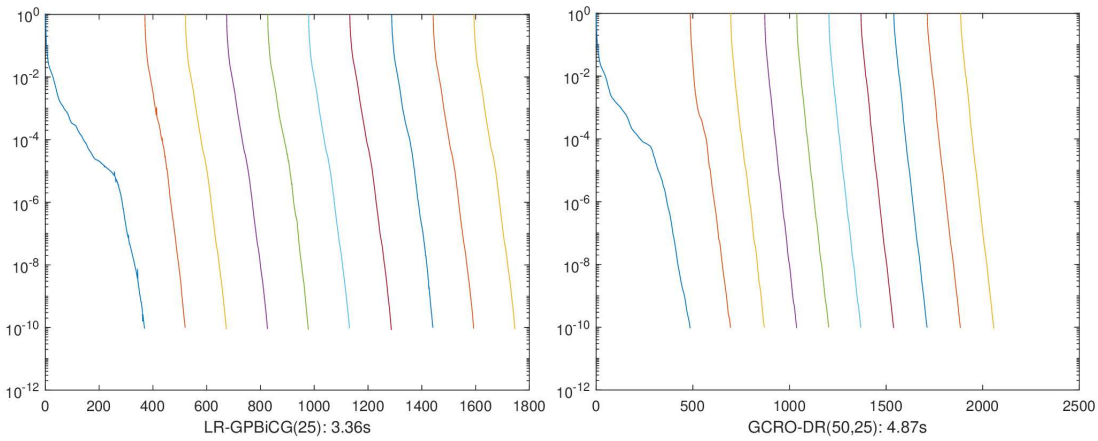


Figure 10: Comparison of convergence curves for LR-GPBICG(25) and GCRO-DR(50,25) for FFEC Problems, where x-axis represents the number of iterations, and y-axis represents the relative residual norm.

GPBiCG requires two matrix-vector products, while GCRO-DR requires only one, so LR-GPBiCG is more expensive than GCRO-DR in terms of matrix-vector products at each iteration. However, in terms of CPU computing time, GCRO-DR(50,25) takes 4.87 seconds, while LR-GPBiCG(25) only consumes 3.36 seconds. In addition, LR-GPBiCG has lower storage requirements due to short-term recursion.

It is remarked that, as the coefficient matrices and the right-hand sides change accumulately, the improvement of convergence efficiency will gradually become worse. In practice, it is difficult to monitor the acceleration quality of the recycling space. If the improvement effect of convergence is no longer obvious after solving a certain number of systems with LR-GPBiCG, we would suggest to re-apply GPBiCG to solve the system to update LR. By the way, adaptive updating of the recycling space needs to be studied.

6 Conclusions

LR-GPBiCG as a promising product-type variant of RBiCG, inherits the stable convergence property of GPBiCG while efficiently recycles partial information of approximation search subspaces for solving subsequent systems of linear equations. That is, for the solution of slowly changing sequential linear equations, it speeds up convergence by keeping part of the subspace information from one family to the next. The recycling space constructed by the difference vector of approximate solutions, which saves the cost of storage and calculation, provides an economic way for producing product-type variants of RBiCG. Numerical experiments show that the solution process of the subsequent systems can be accelerated obviously when the recycling space is selected with appropriate dimensions. Compared to GCRO-DR, LR-GPBiCG is competitive in both memory and CPU computing time in our experiments.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable suggestions to improve the quality of our paper.

The research is supported by the Science Fund for Distinguished Young Scholars of Sichuan Province (Grant No. 2023NSFSC1920), by the National Natural Science Foundation of China (Grant No. 12071062), by the Science Challenge Project (Grant No. TZ2016002).

References

[1] K. Ahuja, *Recycling bi-Lanczos algorithms: BiCG, CGS, and BiCGSTAB*, Master's Thesis, Virginia Polytechnic Institute and State University, 2009.
[2] E. Agullo, L. Giraud, and Y.-F. Jing, *Block GMRES method with inexact breakdowns and deflated restarting*, SIAM J. Matrix Anal. Appl., 35(4):1625–1651, 2014.
[3] K. Ahuja, E. de Sturler, L.-H. Feng, and P. Benner, *Recycling BiCGSTAB with an application to parametric model order reduction*, SIAM J. Sci. Comput., 37(5):S429–S446, 2015.
[4] K. Ahuja, E. de Sturler, S. Gugercin, and E. R. Chang, *Recycling BiCG with an application to model reduction*, SIAM J. Sci. Comput., 34(4):A1925–A1949, 2012.
[5] A. Amritkar, E. de Sturler, K. Świrydowicz, D. Tafti, and K. Ahuja, *Recycling Krylov subspaces for CFD applications and a new hybrid recycling solver*, J. Comput. Phys., 303:222–237, 2015.
[6] A. H. Baker, E. R. Jessup, and Tz. V. Kolev, *A simple strategy for varying the restart parameter in GMRES(m)*, J. Comput. Appl. Math., 230:751–761, 2009.
[7] A. H. Baker, E. R. Jessup, and T. Manteuffel, *A technique for accelerating the convergence of restarted GMRES*, SIAM J. Sci. Comput., 26(4):962–984, 2005.
[8] U. Baur, C. Beattie, P. Benner, and S. Gugercin, *Interpolatory projection methods for parameterized model reduction*, SIAM J. Sci. Comput., 33(5):2489–2518, 2011.
[9] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. J. Dongarra, *Matrix Market: A web resource for test matrix collections*, in: Quality of Numerical Software, Springer, 125–137, 1997.
[10] B. Carpentieri, M. Tavelli, D.-L. Sun, T.-Z. Huang, and Y.-F. Jing, *Fast iterative solution of multiple right-hand sides MoM linear systems on CPUs and GPUs computers*, IEEE Trans. Microw. Theory Tech., 72(8):4431–4444, 2024.
[11] A. Carr, E. de Sturler, and S. Gugercin, *Preconditioning parametrized linear systems*, SIAM J. Sci. Comput., 43(3):A2242–A2267, 2021.
[12] H. A. Daas, L. Grigori, P. Hénon, and P. Ricoux, *Recycling Krylov subspaces and truncating deflation subspaces for solving sequence of linear systems*, ACM Trans. Math. Softw., 47:1–30, 2021.
[13] E. de Sturler, *Nested Krylov methods based on GCR*, J. Comput. Appl. Math., 67(1):15–41, 1996.
[14] K. Du, J.-J. Fan, X.-H. Sun, F. Wang, and Y.-L. Zhang, *On Krylov subspace methods for skew-symmetric and shifted skew-symmetric linear systems*, Adv. Comput. Math., 50(4):78, 2024.
[15] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in: Numerical Analysis. Lecture Notes in Mathematics, Vol. 506, Springer, 73–89, 1976.
[16] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14(1):137–158, 1993.

[17] A. Garcia, S.-H. Xie, and A. Carr, *Implementing recycling methods for linear systems in Python with an application to multiple objective optimization*, in: 2023 International Conference on Machine Learning and Applications (ICMLA), IEEE, 1759–1764, 2023.
[18] L. Giraud, S. Gratton, X. Pinel, and X. Vasseur, *Flexible GMRES with deflated restarting*, SIAM J. Sci. Comput., 32:1858–1878, 2010.
[19] L. Giraud, Y.-F. Jing, and Y.-F. Xiang, *A block minimum residual norm subspace solver with partial convergence management for sequences of linear systems*, SIAM J. Matrix Anal. Appl., 43(2):710–739, 2022.
[20] A. S. Gullerud and R. H. Dodds Jr., *MPI-based implementation of a PCG solver using an EBE architecture and preconditioner for implicit, 3-D finite element analysis*, Comput. Struct., 79(5):553–575, 2001.
[21] M. H. Gutknecht, *Block Krylov space methods for linear systems with multiple right-hand sides: An introduction*, in: Modern Mathematical Models, Methods and Algorithms for Real World Systems, Anamaya Publishers, 420–447, 2006.
[22] M. H. Gutknecht, *Deflated and augmented Krylov subspace methods: A framework for deflated BiCG and related solvers*, SIAM J. Matrix Anal. Appl., 35(4):1444–1466, 2014.
[23] T. Iwashita, K. Ikehara, T. Fukaya, and T. Mifune, *Convergence acceleration of preconditioned conjugate gradient solver based on error vector sampling for a sequence of linear systems*, Numer. Linear Algebra Appl., 30(6):e2512, 2023.
[24] M. Kilmer and E. de Sturler, *Recycling subspace information for diffuse optical tomography*, SIAM J. Sci. Comput., 27(6):2140–2166, 2006.
[25] B. Liu, L. Xu, H. Wang, S.-Y. Yang, H.-X. Liu, S.-C. Huang, and B. Li, *A high efficiency three-dimensional finite-element electromagnetic radiation simulation tool*, Int. J. RF Microw. Comput.-Aided Eng., 31(10):e22789, 2021.
[26] B. Medeke, *Set QCD: Quantum Chromodynamics, description of matrix set on NIST Matrix Market*, 2004, `http://math.nist.gov/MatrixMarket`.
[27] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Sci. Comput., 16(4):1154–1171, 1995.
[28] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24(1):20–37, 2002.
[29] R. B. Morgan, *Restarted block GMRES with deflation of eigenvalues*, Appl. Numer. Math., 54(2):222–236, 2005.
[30] M. P. Neuenhofen, *Short-recurrence and -storage recycling of large Krylov subspaces for sequences of linear systems with changing right-hand sides*, arXiv:1512.05101, 2015.
[31] M. P. Neuenhofen, *M(s)STAB(ℓ): A generalization of IDR(s)STAB(ℓ) for sequences of linear systems*, arXiv:1604.06043, 2016.
[32] M. P. Neuenhofen, *A restarted GMRES-based implementation of IDR(s)stab(ℓ) to yield higher robustness*, Master's Thesis, RWTH Aachen University, Aachen Institute for Advanced Study in Computational Engineering Science, 2017.
[33] M. P. Neuenhofen and C. Greif, *MSTAB: Stabilized induced dimension reduction for Krylov subspace recycling*, SIAM J. Sci. Comput., 40(2):B554–B571, 2018.
[34] M. P. Neuenhofen and S. Groß, *Memory-efficient recycling of large Krylov-subspaces for sequences of Hermitian linear systems*, arXiv:1604.04052, 2017.
[35] M. Nilsson, *Different methods that reduce cost in monostatic RCS computations for MoM accelerated by MLFMA*, Department of Information Technology, Uppsala University, 2004.
[36] Q. Niu, L.-Z. Lu, and G. Liu, *Accelerated GCRO-DR method for solving sequences of systems of linear equations*, J. Comput. Appl. Math., 253:131–141, 2013.
[37] C. C. Paige, B. N. Parlett, and H. A. Van der Vorst, *Approximate solutions and eigenvalue bounds*

from Krylov subspaces, Numer. Linear Algebra Appl., 2(2):115–133, 1995.
[38] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput., 28(5):1651–1674, 2006.
[39] X.-Q. Yue, S.-L. Huang, and X.-W. Xu, *αSetup-PCTL: An adaptive setup-based two-level preconditioner for sequence of linear systems of three-temperature energy equations*, Commun. Comput. Phys., 32(5):1287–1309, 2023.
[40] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.
[41] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7:856–869, 1986.
[42] Y. Shao, Z. Peng, and J. F. Lee, *Full-wave real-life 3-D package signal integrity analysis using nonconformal domain decomposition method*, IEEE Trans. Microw. Theory Tech., 59(2):230–241, 2010.
[43] P. Sonneveld and M. B. Van Gijzen, *IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations*, SIAM J. Sci. Comput., 31(2):1035–1062, 2009.
[44] K. M. Soodhalter, E. de Sturler, and M. E. Kilmer, *A survey of subspace recycling iterative methods*, GAMM-Mitteilungen, 43(4):e202000016, 2020.
[45] H. Tamori, T. Fukaya, and T. Iwashita, *Subspace correction preconditioning for solving a sequence of asymmetric linear systems using the Bi-CGSTAB method*, J. Inf. Process., 31:875–884, 2023.
[46] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Comput., 13(2):631–644, 1992.
[47] S. Wang, E. de Sturler, and G. H. Paulino, *Large-scale topology optimization using preconditioned Krylov subspace methods with recycling*, Internat. J. Numer. Methods Engrg., 69(12):2441–2468, 2007.
[48] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22(2):602–616, 2000.
[49] Z. Yang, Y.-F. Jing, and Q. Niu, *Restarted simpler GMRES augmented with harmonic Ritz vectors and approximate errors*, J. Comput. Appl. Math., 368:112565, 2020.
[50] S.-L. Zhang, *GPBi-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems*, SIAM J. Sci. Comput., 18(2):537–551, 1997.
[51] Y. Zhu and A. C. Cangellaris, *Multigrid Finite Element Methods for Electromagnetic Field Modeling*, John Wiley & Sons, 2006.