Preconditioned IDRStab Algorithms for Solving Nonsymmetric Linear Systems

Kensuke Aihara, Kuniyoshi Abe, and Emiko Ishiwata

***Abstract*—The IDRStab method, which combines the Induced Dimension Reduction (IDR) (s) method with higher-order stabilizing polynomials, is an effective method for solving large nonsymmetric linear systems. IDRStab can be implemented using different algorithms which are mathematically equivalent. In this paper, we illustrate preconditioned algorithms for three variants of IDRStab and describe their advantages. Numerical experiments show the differences in the convergence of the variants of IDRStab with preconditioning.**

***Index Terms*—linear systems, induced dimension reduction, IDRStab method, preconditioning.**

I. Introduction

We consider Krylov subspace methods for solving a large nonsymmetric linear system

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$. The Bi-Conjugate Gradient (Bi-CG) method [4], [7] and the hybrid Bi-CG methods, such as the Bi-CG STABilized (Bi-CGSTAB) method [17] and the BiCGstab(ℓ) method [11], are well-known methods for solving linear systems.

Preconditioning strategies are useful for enhancing the convergence of Krylov subspace methods (see, e.g., [6], [16]). In general, when using a preconditioner K such that $K \approx A$, the system (1) is transformed into a well-conditioned system. Multiplying the system (1) by K^{-1} from the right and left sides gives, respectively, the right preconditioned system

$$\tilde{A}\tilde{\boldsymbol{x}} = \boldsymbol{b}, \quad \tilde{A} = AK^{-1}, \quad \tilde{\boldsymbol{x}} = K\boldsymbol{x} \tag{2}$$

and the left preconditioned system

$$\tilde{A}\boldsymbol{x} = \tilde{\boldsymbol{b}}, \quad \tilde{A} = K^{-1}A, \quad \tilde{\boldsymbol{b}} = K^{-1}\boldsymbol{b}. \tag{3}$$

Incomplete LU (ILU) factorization [8], [10] is often used to construct the preconditioner K.

The IDR(s) method [14], which is based on the Induced Dimension Reduction (IDR) theorem, has been proposed in 2008. It has been reported that IDR(s) is often more effective than the hybrid Bi-CG methods. IDR(s) with $s > 1$ can be considered to be Bi-CGSTAB with an s-dimensional initial shadow residual [12], [15]. The GBi-CGSTAB(s, ℓ) method [15] and the IDR(s)stab(ℓ) method [13], which combine IDR(s) with higher-order stabilizing polynomials, such as are used in BiCGstab(ℓ), have independently been developed in 2010. GBi-CGSTAB(s, ℓ) and IDR(s)stab(ℓ) are mathematically equivalent, but the methods differ in their implementation; for instance, the computation of the recurrence coefficients in GBi-CGSTAB(s, ℓ) is different from that in IDR(s)stab(ℓ). The convention to call all of these $IDRStab$ has been introduced in [9]. A variant of IDR(s)stab(ℓ), which we will refer to as the $AAI\ variant$, has recently been proposed for improving the accuracy of the approximate solutions [1]. The AAI variant is also mathematically equivalent to IDR(s)stab(ℓ), but uses an alternative recurrence formula for updating the residuals, and the AAI variant requires fewer vector updates than does IDR(s)stab(ℓ).

The right preconditioned algorithms of GBi-CGSTAB(s, ℓ) and the AAI variant have been derived in [15] and [1], respectively. It has been reported in [5] that the accuracy of the approximate solutions obtained by GBi-CGSTAB(s, ℓ) and IDR(s)stab(ℓ) is improved by preconditioning. However, the convergence of IDR(s)stab(ℓ), GBi-CGSTAB(s, ℓ), and the AAI variant with right and left preconditioning has not previously been compared. In this paper, we therefore derive the right and left preconditioned algorithms for the variants of IDRStab, and discuss their advantages. Numerical experiments on model problems with nonsymmetric matrices show the differences in the convergence of IDR(s)stab(ℓ), GBi-CGSTAB(s, ℓ), and the AAI variant with right and left preconditioning.

This paper is organized as follows. In the next section, the preconditioned algorithms for the variants of IDRStab are derived and the computational costs are compared. In section III, through numerical experiments, we compare the convergence of IDR(s)stab(ℓ), GBi-CGSTAB(s, ℓ), and the AAI variant with the ILU preconditioner. Concluding remarks are presented in section IV.

II. IDRStab with preconditioning

In this section, we illustrate the right and left preconditioned algorithms for IDRStab. Let $\boldsymbol{x}_0$ and $\boldsymbol{r}_0 \equiv \boldsymbol{b} - A\boldsymbol{x}_0$ be an initial guess and the corresponding initial residual, respectively. The residual $\boldsymbol{r}_k$ generated by IDRStab satisfies

$$\boldsymbol{r}_k \in \mathcal{S}(P_k, A, \tilde{R}_0) \equiv \{P_k(A)\boldsymbol{v} | \boldsymbol{v} \perp \mathcal{K}_k(A^\top, \tilde{R}_0)\}.$$

Here, the integer k is a multiple of ℓ, and the polynomial $P_k(\lambda)$ of degree k is defined by a product of stabilizing polynomials of degree ℓ, i.e., $P_k(\lambda) = (1 - \sum_{j=1}^{\ell} \gamma_{j,k-\ell}\lambda^j)P_{k-\ell}(\lambda)$. $\mathcal{K}_k(A^\top, \tilde{R}_0)$ is a so-called kth-block Krylov subspace generated by the transpose $A^\top$ of A and a fixed matrix $\tilde{R}_0 \in \mathbb{R}^{n\times s}$:

$$\mathcal{K}_k(A^\top, \tilde{R}_0) \equiv \left\{ \sum_{j=0}^{k-1} (A^\top)^j \tilde{R}_0 \vec{\eta}_j \mid \vec{\eta}_j \in \mathbb{R}^s \right\}.$$

Manuscript received December 30, 2013; revised July 6, 2014.

K. Aihara is with Department of Mathematical Information Science, Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162-8601, Japan. (e-mail: kaihara@rs.tus.ac.jp)

K. Abe is with Faculty of Economics and Information, Gifu Shotoku University, 1-38 Nakauzura, Gifu 500-8288, Japan.

E. Ishiwata is with Department of Mathematical Information Science, Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162-8601, Japan.

The residual $\boldsymbol{r}_k$ is updated to $\boldsymbol{r}_{k+\ell}$ by performing the following two steps[1].

- **The IDR step.** This step consists of ℓ repetitions. Suppose that a residual $\boldsymbol{r}$ in the subspace $\{P_k(A)\boldsymbol{v}|\boldsymbol{v} \perp \mathcal{K}_{k+j-1}(A^\top, \tilde{R}_0)\}$ is generated at the $(j-1)$st $(j \leq \ell)$ repetition, where $\boldsymbol{r} = \boldsymbol{r}_k$ for $j = 1$. At the jth repetition, $\boldsymbol{r}$ is updated to a residual $\boldsymbol{r}'$ which belongs to $\{P_k(A)\boldsymbol{v}|\boldsymbol{v} \perp \mathcal{K}_{k+j}(A^\top, \tilde{R}_0)\}$. If $j < \ell$, $\boldsymbol{r}'$ is renamed to $\boldsymbol{r}$ at the next repetition.
- **The polynomial step.** After the ℓ repetitions of the IDR step, multiplying $\boldsymbol{r}' \in \{P_k(A)\boldsymbol{v}|\boldsymbol{v} \perp \mathcal{K}_{k+\ell}(A^\top, \tilde{R}_0)\}$ by a stabilizing polynomial of degree ℓ gives $\boldsymbol{r}_{k+\ell} = (I - \sum_{j=1}^{\ell} \gamma_{j,k} A^j)\boldsymbol{r}' \in \mathcal{S}(P_{k+\ell}, A, \tilde{R}_0)$, where the scalars $\gamma_{j,k}$ for $j = 1, 2, \ldots, \ell$ are determined by minimizing $\|\boldsymbol{r}_{k+\ell}\|_2$.

The steps to update $\boldsymbol{r}_k$ to $\boldsymbol{r}_{k+\ell}$ are referred to as one cycle of IDRStab. IDR(s)stab(ℓ), GBi-CGSTAB(s, ℓ), and the AAI variant generate the same residual at each cycle in exact arithmetic, but they may converge differently in finite precision arithmetic because the methods differ in their implementation (e.g., the computation of recurrence coefficients, such as $\vec{\beta}$, below). We refer to [1], [13], [15] for the implementation details.

The notation in the algorithms follows MATLAB conventions: the matrix $W = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_q]$ and the vector $\boldsymbol{w}_q$ for $q \leq s$ are denoted by $W_{(:,1:q)}$ and $W_{(:,q)}$, respectively, and $[W_0; W_1; \cdots; W_j] \equiv [W_0^\top, W_1^\top, \ldots, W_j^\top]^\top$. $\mathbf{U}_i$, $\mathbf{V}_i$, $\mathbf{r}_i$, and $\mathbf{u}_i$ for $i = 0, 1, \ldots, j$ are related to $\mathbf{U}$, $\mathbf{V}$, $\mathbf{r}$, and $\mathbf{u}$ according as $\mathbf{U} = [\mathbf{U}_0; \mathbf{U}_1; \cdots; \mathbf{U}_j]$, $\mathbf{V} = [\mathbf{V}_0; \mathbf{V}_1; \cdots; \mathbf{V}_j]$, $\mathbf{r} = [\mathbf{r}_0; \mathbf{r}_1; \cdots; \mathbf{r}_j]$, and $\mathbf{u} = [\mathbf{u}_0; \mathbf{u}_1; \cdots; \mathbf{u}_j]$, respectively.

A. Original IDR(s)stab(ℓ) with preconditioning

We now derive the preconditioned algorithms for IDR(s)stab(ℓ).

We first apply the IDR(s)stab(ℓ) algorithm [13, Algorithm 5.3] to the right preconditioned system (2). By replacing $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{r}}$, and an auxiliary matrix $\tilde{U} \in \mathbb{R}^{n \times s}$ by $K\boldsymbol{x}$, $\boldsymbol{r}$, and U, respectively, the approximation and the corresponding residual are updated by

$$\boldsymbol{x}' = \boldsymbol{x} + K^{-1}U\vec{\alpha}, \quad \boldsymbol{r}' = \boldsymbol{r} - AK^{-1}U\vec{\alpha} \tag{4}$$

at the jth repetition of the IDR step, where $\vec{\alpha} = \sigma^{-1}(\tilde{R}_0^\top (AK^{-1})^{j-1}\boldsymbol{r})$ and $\sigma = \tilde{R}_0^\top (AK^{-1})^j U$. Introducing $\hat{U} \equiv K^{-1}U$ transforms the first expression in (4) to

$$\boldsymbol{x}' = \boldsymbol{x} + \hat{U}\vec{\alpha}. \tag{5}$$

Therefore, U is not needed for updating the approximation and the residual. With $\hat{\boldsymbol{r}} \equiv K^{-1}\boldsymbol{r}$ and $\hat{\boldsymbol{r}}' \equiv K^{-1}\boldsymbol{r}'$, we also compute the vectors

$$\begin{aligned}(K^{-1}A)^i\hat{\boldsymbol{r}}' &= (K^{-1}A)^i\hat{\boldsymbol{r}} - (K^{-1}A)^{i+1}\hat{U}\vec{\alpha}, \\ &i = 0, 1, \ldots, j-2,\end{aligned} \tag{6}$$

$$\begin{aligned}(AK^{-1})^i\boldsymbol{r}' &= (AK^{-1})^i\boldsymbol{r} - (AK^{-1})^{i+1}U\vec{\alpha}, \\ &i = 1, 2, \ldots, j-1\end{aligned} \tag{7}$$

at the jth repetition. Then, multiplying $(AK^{-1})^{j-1}\boldsymbol{r}'$ by K^{-1} and A gives $(K^{-1}A)^{j-1}\hat{\boldsymbol{r}}'$ and $(AK^{-1})^j\boldsymbol{r}'$. Note that $(K^{-1}A)^i\hat{\boldsymbol{r}}'$ and $(AK^{-1})^i\boldsymbol{r}'$ are used for updating the approximation and the corresponding residual, respectively, at the polynomial step. The auxiliary vectors $\hat{\boldsymbol{u}}_{i+1}$ and $\boldsymbol{u}_{i+1}$ for $i = 0, 1, \ldots, j-1$ are set to $\hat{\boldsymbol{u}}_{i+1} = (K^{-1}A)^i\hat{\boldsymbol{r}}'$ and $\boldsymbol{u}_{i+1} = (AK^{-1})^{i+1}\boldsymbol{r}'$, respectively. To update $(K^{-1}A)^i\hat{U}$ and $(AK^{-1})^iU$ in (6) and (7), the following procedures are repeated for $q = 1, 2, \ldots, s$:

$$\vec{\beta} = \sigma^{-1}(\tilde{R}_0^\top \boldsymbol{u}_j), \tag{8}$$

$$\begin{cases}\hat{\boldsymbol{u}}_i = \hat{\boldsymbol{u}}_{i+1} - (K^{-1}A)^i\hat{U}\vec{\beta}, \\ \boldsymbol{u}_i = \boldsymbol{u}_{i+1} - (AK^{-1})^{i+1}U\vec{\beta}, \\ i = 0, 1, \ldots, j-1,\end{cases} \tag{9}$$

$$\hat{\boldsymbol{u}}_j = K^{-1}\boldsymbol{u}_{j-1}, \quad \boldsymbol{u}_j = A\hat{\boldsymbol{u}}_j, \tag{10}$$

$$\begin{cases}(K^{-1}A)^i\hat{V}\boldsymbol{e}_q = \hat{\boldsymbol{u}}_i, \\ (AK^{-1})^{i+1}V\boldsymbol{e}_q = \boldsymbol{u}_i, \\ i = 0, 1, \ldots, j.\end{cases} \tag{11}$$

If $j < \ell$, $(K^{-1}A)^i\hat{V}$ and $(AK^{-1})^{i+1}V$ for $i = 0, 1, \ldots, j$ are renamed to $(K^{-1}A)^i\hat{U}$ and $(AK^{-1})^{i+1}U$, respectively.

Algorithm 1. Original IDR(s)stab(ℓ) with right preconditioning.

1. Select an initial guess $\mathbf{x}$ and an $(n \times s)$ matrix $\tilde{\mathbf{R}}_0$.
2. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}$, $\mathbf{r} = [\mathbf{r}_0]$
3. Generate an initial $\hat{\mathbf{U}} = [\hat{\mathbf{U}}_0] = [\mathbf{K}^{-1}\mathbf{U}_0]$, $\mathbf{U} = [\mathbf{A}\hat{\mathbf{U}}_0]$
4. While $\|\mathbf{r}_0\|_2 > tol$
% *The IDR step*
5. For $j = 1, 2, \ldots, \ell$
6. $\sigma = \tilde{\mathbf{R}}_0^\top \mathbf{U}_{j-1}$, $\vec{\alpha} = \sigma^{-1}(\tilde{\mathbf{R}}_0^\top \mathbf{r}_{j-1})$
7. $\mathbf{x} = \mathbf{x} + \hat{\mathbf{U}}_0\vec{\alpha}$, $\mathbf{r} = \mathbf{r} - [\mathbf{U}_0; \mathbf{U}_1; \cdots; \mathbf{U}_{j-1}]\vec{\alpha}$
8. if $j = 1$ then
9. $\hat{\mathbf{r}} = [\mathbf{K}^{-1}\mathbf{r}_0]$
10. else
11. $\hat{\mathbf{r}} = \hat{\mathbf{r}} - [\hat{\mathbf{U}}_1; \hat{\mathbf{U}}_2; \cdots; \hat{\mathbf{U}}_{j-1}]\vec{\alpha}$, $\hat{\mathbf{r}} = [\hat{\mathbf{r}}; \mathbf{K}^{-1}\mathbf{r}_{j-1}]$
12. end if
13. $\mathbf{r} = [\mathbf{r}; \mathbf{A}\hat{\mathbf{r}}_{j-1}]$
14. For $q = 1, 2, \ldots, s$
15. if $q = 1$ then
16. $\hat{\mathbf{u}} = \hat{\mathbf{r}}$, $\mathbf{u} = [\mathbf{r}_1; \mathbf{r}_2; \cdots; \mathbf{r}_j]$
17. else
18. $\hat{\mathbf{u}} = [\hat{\mathbf{u}}_1; \hat{\mathbf{u}}_2; \cdots; \hat{\mathbf{u}}_j]$, $\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \cdots; \mathbf{u}_j]$
19. end if
20. $\vec{\beta} = \sigma^{-1}(\tilde{\mathbf{R}}_0^\top \mathbf{u}_{j-1})$, $\hat{\mathbf{u}} = \hat{\mathbf{u}} - \hat{\mathbf{U}}\vec{\beta}$, $\mathbf{u} = \mathbf{u} - \mathbf{U}\vec{\beta}$,
21. $\hat{\mathbf{u}} = [\hat{\mathbf{u}}; \mathbf{K}^{-1}\mathbf{u}_{j-1}]$
22. Orthonormalize $\hat{\mathbf{u}}_j$ to the columns of $\hat{\mathbf{V}}_{j(:,1:q-1)}$
23. $\mathbf{u} = [\mathbf{u}; \mathbf{A}\hat{\mathbf{u}}_j]$, $\hat{\mathbf{V}}_{(:,q)} = \hat{\mathbf{u}}$, $\mathbf{V}_{(:,q)} = \mathbf{u}$
24. End for
25. if $j < \ell$ then
26. $\hat{\mathbf{U}} = \hat{\mathbf{V}}$, $\mathbf{U} = \mathbf{V}$
27. end if
28. End for
% *The polynomial step*
29. $\vec{\gamma} = [\gamma_1; \gamma_2; \cdots; \gamma_\ell] = \arg\min_{\vec{\gamma}} \|\mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_\ell]\vec{\gamma}\|_2$
30. $\mathbf{x} = \mathbf{x} + [\hat{\mathbf{r}}_0, \hat{\mathbf{r}}_1, \ldots, \hat{\mathbf{r}}_{\ell-1}]\vec{\gamma}$, $\mathbf{r}_0 = \mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_\ell]\vec{\gamma}$
31. $\hat{\mathbf{U}} = [\hat{\mathbf{V}}_0 - \sum_{j=1}^{\ell} \gamma_j \hat{\mathbf{V}}_j]$, $\mathbf{U} = [\mathbf{V}_0 - \sum_{j=1}^{\ell} \gamma_j \mathbf{V}_j]$
32. End while

[1] In GBi-CGSTAB(s, ℓ), the IDR step and the polynomial step are referred to as the *GBi-CG(s) part* and the *MR part*, respectively [15].

Algorithm 2. Original IDR(s)stab(ℓ) with left preconditioning.

1. Select an initial guess $\mathbf{x}$ and an $(n \times s)$ matrix $\tilde{\mathbf{R}}_0$.
2. Compute $\mathbf{r}_0 = \mathbf{K}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}),\ \mathbf{r} = [\mathbf{r}_0]$
3. Generate an initial $\mathbf{U} = [\mathbf{U}_0; \mathbf{U}_1] = [\mathbf{U}_0; \mathbf{K}^{-1}\mathbf{A}\mathbf{U}_0]$
4. While $\|\mathbf{r}_0\|_2 > tol$
5. % *The IDR step*
6. For $j = 1, 2, \ldots, \ell$
7. $\sigma = \tilde{\mathbf{R}}_0^\top \mathbf{U}_j,\ \vec{\alpha} = \sigma^{-1}(\tilde{\mathbf{R}}_0^\top \mathbf{r}_{j-1})$
8. $\mathbf{x} = \mathbf{x} + \mathbf{U}_0\vec{\alpha},\ \mathbf{r} = \mathbf{r} - [\mathbf{U}_1; \mathbf{U}_2; \cdots; \mathbf{U}_j]\vec{\alpha}$
9. $\mathbf{r} = [\mathbf{r}; \mathbf{K}^{-1}\mathbf{A}\mathbf{r}_{j-1}]$
10. For $q = 1, 2, \ldots, s$
11. if $q = 1$ then
12. $\mathbf{u} = \mathbf{r}$
13. else
14. $\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \cdots; \mathbf{u}_{j+1}]$
15. end if
16. $\vec{\beta} = \sigma^{-1}(\tilde{\mathbf{R}}_0^\top \mathbf{u}_j),\ \mathbf{u} = \mathbf{u} - \mathbf{U}\vec{\beta}$
17. Orthonormalize $\mathbf{u}_j$ to the columns of $\mathbf{V}_{j(:,1:q-1)}$
18. $\mathbf{u} = [\mathbf{u}; \mathbf{K}^{-1}\mathbf{A}\mathbf{u}_j],\ \mathbf{V}_{(:,q)} = \mathbf{u}$
19. End for
20. if $j < \ell$ then
21. $\mathbf{U} = \mathbf{V}$
22. end if
23. End for
24. % *The polynomial step*
25. $\vec{\gamma} = [\gamma_1; \gamma_2; \cdots; \gamma_\ell] = \arg\min_{\vec{\gamma}} \|\mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_\ell]\vec{\gamma}\|_2$
26. $\mathbf{x} = \mathbf{x} + [\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{\ell-1}]\vec{\gamma},\ \mathbf{r}_0 = \mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_\ell]\vec{\gamma}$
27. $\mathbf{U} = [\mathbf{V}_0 - \sum_{j=1}^{\ell} \gamma_j \mathbf{V}_j; \mathbf{V}_1 - \sum_{j=1}^{\ell} \gamma_j \mathbf{V}_{j+1}]$
28. End while

At the polynomial step, the approximation and the corresponding residual are updated by

$$\boldsymbol{x}_{k+\ell} = \boldsymbol{x}' + \sum_{j=1}^{\ell} \gamma_{j,k} K^{-1}(AK^{-1})^{j-1}\boldsymbol{r}', \qquad (12)$$

$$\boldsymbol{r}_{k+\ell} = \boldsymbol{r}' - \sum_{j=1}^{\ell} \gamma_{j,k}(AK^{-1})^{j}\boldsymbol{r}'.$$

Using $\hat{\boldsymbol{r}} = K^{-1}\boldsymbol{r}$, (12) is reformulated as

$$\boldsymbol{x}_{k+\ell} = \boldsymbol{x}' + \sum_{j=1}^{\ell} \gamma_{j,k}(K^{-1}A)^{j-1}\hat{\boldsymbol{r}}'. \qquad (13)$$

Finally, the matrices $\hat{U}$ and $AK^{-1}U$, which are used at the next cycle, are computed by

$$\hat{U} = \hat{V} - \sum_{j=1}^{\ell} \gamma_{j,k}(K^{-1}A)^{j}\hat{V},$$

$$AK^{-1}U = AK^{-1}V - \sum_{j=1}^{\ell} \gamma_{j,k}(AK^{-1})^{j+1}V.$$

Algorithm 1 displays IDR(s)stab(ℓ) with right preconditioning. Note that the residual $\boldsymbol{r}$ and the auxiliary matrix $\hat{U}$ are stored in $\mathbf{r}_0$ and $\hat{\mathbf{U}}_0$, respectively, and $\mathbf{r}_j,\ \hat{\mathbf{r}}_j,\ \hat{\mathbf{U}}_j$, and $\mathbf{U}_j$ represent $(\mathbf{A}\mathbf{K}^{-1})^j\mathbf{r}_0,\ \mathbf{K}^{-1}\mathbf{r}_j,\ (\mathbf{K}^{-1}\mathbf{A})^j\hat{\mathbf{U}}_0$, and $\mathbf{A}\hat{\mathbf{U}}_j$, respectively.

IDR(s)stab(ℓ) with left preconditioning is displayed in Algorithm 2, which can be derived by applying the IDR(s)stab(ℓ) algorithm to the left preconditioned system (3). In other words, we obtain the left preconditioned algorithm by setting an initial residual $\boldsymbol{r}_0 = K^{-1}(\boldsymbol{b} - A\boldsymbol{x}_0)$ and replacing $\tilde{A}$, $\tilde{\boldsymbol{r}}$, and $\tilde{U}$ by $K^{-1}A$, $\boldsymbol{r}$, and U, respectively. In Algorithm 2, $\mathbf{r}_j$ and $\mathbf{U}_j$ represent $(\mathbf{K}^{-1}\mathbf{A})^j\mathbf{r}_0$ and $(\mathbf{K}^{-1}\mathbf{A})^j\mathbf{U}_0$, respectively.

B. GBi-CGSTAB(s, ℓ) with preconditioning

We now describe the preconditioned algorithms of GBi-CGSTAB(s, ℓ).

Algorithm 3 displays GBi-CGSTAB(s, ℓ) with the right preconditioning proposed in [15], which can be derived by applying the GBi-CGSTAB(s, ℓ) algorithm [15, Algorithm 4] to the right preconditioned system (2). Here $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{r}}$, and $\tilde{U}$ are replaced by $K\boldsymbol{x}$, $\boldsymbol{r}$, and U, respectively. As are used in the right preconditioned IDR(s)stab(ℓ), $\hat{U} \equiv K^{-1}U$ and $\hat{\boldsymbol{r}} \equiv K^{-1}\boldsymbol{r}$ are introduced to design the recurrence formulas for updating the approximations $\boldsymbol{x}$ (see (5) and (13)). In Algorithm 3, $\mathbf{r}_j,\ \hat{\mathbf{r}}_j,\ \mathbf{U}_j$, and $\hat{\mathbf{U}}_j$ represent $(\mathbf{A}\mathbf{K}^{-1})^j\mathbf{r}_0,\ \mathbf{K}^{-1}\mathbf{r}_j,\ (\mathbf{A}\mathbf{K}^{-1})^j\mathbf{U}_0$, and $\mathbf{K}^{-1}\mathbf{U}_j$, respectively.

GBi-CGSTAB(s, ℓ) with left preconditioning is displayed in Algorithm 4, which can be derived by applying the GBi-CGSTAB(s, ℓ) algorithm to the left preconditioned system (3). Here, $\boldsymbol{r}_0$ is set to $K^{-1}(\boldsymbol{b} - A\boldsymbol{x}_0)$, and $\tilde{A}$, $\tilde{\boldsymbol{r}}$, and $\tilde{U}$ are replaced by $K^{-1}A$, $\boldsymbol{r}$, and U, respectively. As in Algorithm 2, $\mathbf{r}_j$ and $\mathbf{U}_j$ in Algorithm 4 represent $(\mathbf{K}^{-1}\mathbf{A})^j\mathbf{r}_0$ and $(\mathbf{K}^{-1}\mathbf{A})^j\mathbf{U}_0$, respectively.

C. AAI variant of IDR(s)stab(ℓ) with preconditioning

We next derive the preconditioned algorithms of the AAI variant. Note that the AAI variant uses an alternative recurrence formula for updating the residuals in order to improve the accuracy of the approximate solutions [1].

We apply the algorithm of the AAI variant [1, Algorithm 1] to the right preconditioned system (2). By replacing $\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{r}}$, and $\tilde{U}$ by $K\boldsymbol{x}$, $\boldsymbol{r}$, and U, respectively, the approximation and the corresponding residual are expressed by

$$\boldsymbol{x}' = \boldsymbol{x} + K^{-1}\boldsymbol{p}, \quad \boldsymbol{r}' = \boldsymbol{r} - AK^{-1}\boldsymbol{p}, \quad \boldsymbol{p} = U\vec{\alpha} \qquad (14)$$

at the jth repetition of the IDR step. By introducing $\hat{U} \equiv K^{-1}U$, (14) can be reformulated as

$$\boldsymbol{x}' = \boldsymbol{x} + \hat{\boldsymbol{p}}, \quad \boldsymbol{r}' = \boldsymbol{r} - A\hat{\boldsymbol{p}}, \quad \hat{\boldsymbol{p}} = \hat{U}\vec{\alpha},$$

where $A\hat{\boldsymbol{p}}$ is obtained by explicitly multiplying $\hat{\boldsymbol{p}}$ by A. Therefore U is not needed for updating the approximation and the residual. Unlike the situation with the right preconditioned IDR(s)stab(ℓ), the matrix $AK^{-1}U$ is also not needed. By using the recurrence formulas (6) and (7) with $\hat{\boldsymbol{r}} \equiv K^{-1}\boldsymbol{r}$ and $\hat{\boldsymbol{r}}' \equiv K^{-1}\boldsymbol{r}'$, the vectors $(K^{-1}A)^i\hat{\boldsymbol{r}}'$ for $i = 0, 1, \ldots, j-2$ and $(AK^{-1})^i\boldsymbol{r}'$ for $i = 1, 2, \ldots, j-2$ are computed at the jth repetition. Multiplying $(K^{-1}A)^{j-2}\hat{\boldsymbol{r}}'$ by A and K^{-1} gives $(AK^{-1})^{j-1}\boldsymbol{r}'$ and $(K^{-1}A)^{j-1}\hat{\boldsymbol{r}}'$. For $j = \ell$, we also compute $(AK^{-1})^\ell\boldsymbol{r}'$ by multiplying $(K^{-1}A)^{\ell-1}\hat{\boldsymbol{r}}'$ by A. These vectors are required for updating the approximation and the corresponding residual at the polynomial step. In particular, $(AK^{-1})^j\boldsymbol{r}'$ for $j = 1, 2, \ldots, \ell$ are used only to determine the scalars

Algorithm 3. GBi-CGSTAB(s, ℓ) with right preconditioning [15].

1. Select an initial guess $\mathbf{x}$ and an $(n \times s)$ matrix $\tilde{\mathbf{R}}_0$.
2. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x},\ \mathbf{r} = [\mathbf{r}_0]$
3. Generate an initial $\mathbf{U} = [\mathbf{U}_0],\ \hat{\mathbf{U}} = [\mathbf{K}^{-1}\mathbf{U}_0],\ \mathbf{U} = [\mathbf{U}; \mathbf{A}\hat{\mathbf{U}}_0]$
4. $\sigma = \tilde{\mathbf{R}}_0^\top \mathbf{U}_1,\ \mathbf{m} = \tilde{\mathbf{R}}_0^\top \mathbf{r}_0,\ \vec{\alpha} = \sigma^{-1}\mathbf{m}$
5. $\mathbf{x} = \mathbf{x} + \hat{\mathbf{U}}_0\vec{\alpha},\ \mathbf{r} = \mathbf{r} - \mathbf{U}_1\vec{\alpha}$
6. $\hat{\mathbf{r}} = [\mathbf{K}^{-1}\mathbf{r}_0],\ \mathbf{r} = [\mathbf{r}; \mathbf{A}\hat{\mathbf{r}}_0],\ \omega = -1$
7. While $\|\mathbf{r}_0\|_2 > tol$
8. $\sigma = -\omega\sigma$
% *The GBi-CG(s) part*
9. For $j = 1, 2, \ldots, \ell$
10. if $j = 1$ at the first cycle, then set $j = 2$
11. $\mathbf{m} = \tilde{\mathbf{R}}_0^\top \mathbf{r}_{j-1}$
12. For $q = 1, 2, \ldots, s$
13. if $q = 1$ then
14. $\vec{\beta} = \sigma^{-1}\mathbf{m},\ \mathbf{U}_{(:,1)} = \mathbf{r} - \mathbf{U}\vec{\beta},\ \hat{\mathbf{U}}_{(:,1)} = \hat{\mathbf{r}} - \hat{\mathbf{U}}\vec{\beta}$
15. else
16. $\vec{\beta} = [\mathbf{m}, \sigma_{(:,1:q-2)}, \sigma_{(:,q:s)}]^{-1}\sigma_{(:,q-1)}$
17. For $i = 0, 1, \ldots, j-1$
18. $\mathbf{U}_{i(:,q)} = \mathbf{U}_{i+1(:,q-1)} - [\mathbf{r}_i, \mathbf{U}_{i+1(:,1:q-2)}, \mathbf{U}_{i(:,q:s)}]\vec{\beta}$
19. End for,
20. For $i = 0, 1, \ldots, j-2$
21. $\hat{\mathbf{U}}_{i(:,q)} = \hat{\mathbf{U}}_{i+1(:,q-1)} - [\hat{\mathbf{r}}_i, \hat{\mathbf{U}}_{i+1(:,1:q-2)}, \hat{\mathbf{U}}_{i(:,q:s)}]\vec{\beta}$
22. End for
23. end if
24. $\hat{\mathbf{U}}_{j-1(:,q)} = \mathbf{K}^{-1}\mathbf{U}_{j-1(:,q)},\ \mathbf{U}_{j(:,q)} = \mathbf{A}\hat{\mathbf{U}}_{j-1(:,q)}$
25. $\sigma_{(:,q)} = \tilde{\mathbf{R}}_0^\top \mathbf{U}_{j(:,q)}$
26. End for
27. $\vec{\alpha} = \sigma^{-1}\mathbf{m}$
28. $\mathbf{x} = \mathbf{x} + \hat{\mathbf{U}}_0\vec{\alpha},\ \mathbf{r} = \mathbf{r} - [\mathbf{U}_1; \mathbf{U}_2; \cdots; \mathbf{U}_j]\vec{\alpha}$
29. $\hat{\mathbf{r}} = \hat{\mathbf{r}} - [\hat{\mathbf{U}}_1; \hat{\mathbf{U}}_2; \cdots; \hat{\mathbf{U}}_{j-1}]\vec{\alpha}$
30. $\hat{\mathbf{r}} = [\hat{\mathbf{r}}; \mathbf{K}^{-1}\mathbf{r}_{j-1}],\ \mathbf{r} = [\mathbf{r}; \mathbf{A}\hat{\mathbf{r}}_{j-1}]$
31. End for
% *The MR part*
32. For $j = 1, 2, \ldots, \ell$
33. For $i = 1, 2, \ldots, j-1$
34. $\tau_{ij} = \mathbf{r}_i^\top \mathbf{r}_j / \rho_i,\ \mathbf{r}_j = \mathbf{r}_j - \tau_{ij}\mathbf{r}_i$
35. End for
36. $\rho_j = \mathbf{r}_j^\top \mathbf{r}_j,\ \gamma'_j = \mathbf{r}_j^\top \mathbf{r}_0 / \rho_j$
37. End for
38. $\gamma_\ell = \gamma'_\ell,\ \omega = \gamma_\ell$
39. For $j = \ell - 1, \ell - 2, \ldots, 1$
40. $\gamma_j = \gamma'_j - \sum_{i=j+1}^{\ell} \tau_{ji}\gamma_i$
41. End for
42. For $j = 1, 2, \ldots, \ell$
43. $\mathbf{x} = \mathbf{x} + \gamma_j\hat{\mathbf{r}}_{j-1},\ \mathbf{r}_0 = \mathbf{r}_0 - \gamma'_j\mathbf{r}_j,\ \mathbf{U}_0 = \mathbf{U}_0 - \gamma_j\mathbf{U}_j$
44. End for
45. End while

Algorithm 4. GBi-CGSTAB(s, ℓ) with left preconditioning.

1. Select an initial guess $\mathbf{x}$ and an $(n \times s)$ matrix $\tilde{\mathbf{R}}_0$.
2. Compute $\mathbf{r}_0 = \mathbf{K}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}),\ \mathbf{r} = [\mathbf{r}_0]$
3. Generate an initial $\mathbf{U} = [\mathbf{U}_0; \mathbf{U}_1] = [\mathbf{U}_0; \mathbf{K}^{-1}\mathbf{A}\mathbf{U}_0]$
4. $\sigma = \tilde{\mathbf{R}}_0^\top \mathbf{U}_1,\ \mathbf{m} = \tilde{\mathbf{R}}_0^\top \mathbf{r}_0,\ \vec{\alpha} = \sigma^{-1}\mathbf{m}$
5. $\mathbf{x} = \mathbf{x} + \mathbf{U}_0\vec{\alpha},\ \mathbf{r} = \mathbf{r} - \mathbf{U}_1\vec{\alpha}$
6. $\mathbf{r} = [\mathbf{r}; \mathbf{K}^{-1}\mathbf{A}\mathbf{r}_0],\ \omega = -1$
7. While $\|\mathbf{r}_0\|_2 > tol$
8. $\sigma = -\omega\sigma$
% *The GBi-CG(s) part*
9. For $j = 1, 2, \ldots, \ell$
10. if $j = 1$ at the first cycle, then set $j = 2$
11. $\mathbf{m} = \tilde{\mathbf{R}}_0^\top \mathbf{r}_{j-1}$
12. For $q = 1, 2, \ldots, s$
13. if $q = 1$ then
14. $\vec{\beta} = \sigma^{-1}\mathbf{m},\ \mathbf{U}_{(:,1)} = \mathbf{r} - \mathbf{U}\vec{\beta}$
15. else
16. $\vec{\beta} = [\mathbf{m}, \sigma_{(:,1:q-2)}, \sigma_{(:,q:s)}]^{-1}\sigma_{(:,q-1)}$
17. For $i = 0, 1, \ldots, j-1$
18. $\mathbf{U}_{i(:,q)} = \mathbf{U}_{i+1(:,q-1)} - [\mathbf{r}_i, \mathbf{U}_{i+1(:,1:q-2)}, \mathbf{U}_{i(:,q:s)}]\vec{\beta}$
19. End for
20. end if
21. $\mathbf{U}_{j(:,q)} = \mathbf{K}^{-1}\mathbf{A}\mathbf{U}_{j-1(:,q)},\ \sigma_{(:,q)} = \tilde{\mathbf{R}}_0^\top \mathbf{U}_{j(:,q)}$
22. End for
23. $\vec{\alpha} = \sigma^{-1}\mathbf{m}$
24. $\mathbf{x} = \mathbf{x} + \mathbf{U}_0\vec{\alpha},\ \mathbf{r} = \mathbf{r} - [\mathbf{U}_1; \mathbf{U}_2; \cdots; \mathbf{U}_j]\vec{\alpha}$
25. $\mathbf{r} = [\mathbf{r}; \mathbf{K}^{-1}\mathbf{A}\mathbf{r}_{j-1}]$
26. End for
% *The MR part*
27. For $j = 1, 2, \ldots, \ell$
28. For $i = 1, 2, \ldots, j-1$
29. $\tau_{ij} = \mathbf{r}_i^\top \mathbf{r}_j / \rho_i,\ \mathbf{r}_j = \mathbf{r}_j - \tau_{ij}\mathbf{r}_i$
30. End for
31. $\rho_j = \mathbf{r}_j^\top \mathbf{r}_j,\ \gamma'_j = \mathbf{r}_j^\top \mathbf{r}_0 / \rho_j$
32. End for
33. $\gamma_\ell = \gamma'_\ell,\ \omega = \gamma_\ell$
34. For $j = \ell - 1, \ell - 2, \ldots, 1$
35. $\gamma_j = \gamma'_j - \sum_{i=j+1}^{\ell} \tau_{ji}\gamma_i$
36. End for
37. For $j = 1, 2, \ldots, \ell - 1$
38. $\gamma''_j = \gamma_{j+1} + \sum_{i=j+1}^{\ell-1} \tau_{ji}\gamma_{i+1}$
39. End for
40. $\mathbf{x} = \mathbf{x} + \gamma_1\mathbf{r}_0,\ \mathbf{r}_0 = \mathbf{r}_0 - \gamma'_\ell\mathbf{r}_\ell,\ \mathbf{U}_0 = \mathbf{U}_0 - \gamma_\ell\mathbf{U}_\ell$
41. For $j = 1, 2, \ldots, \ell - 1$
42. $\mathbf{x} = \mathbf{x} + \gamma''_j\mathbf{r}_j,\ \mathbf{r}_0 = \mathbf{r}_0 - \gamma'_j\mathbf{r}_j,\ \mathbf{U}_0 = \mathbf{U}_0 - \gamma_j\mathbf{U}_j$
43. End for
44. End while

$\gamma_{j,k}$. We use procedures (8)–(11) to obtain $(K^{-1}A)^i\hat{V}$ for $i = 0, 1, \ldots, j$ and $(AK^{-1})^i V$ for $i = 2, 3, \ldots, j$; however, we note that the matrices $(AK^{-1})^i V$ are required only for $1 < j < \ell$. Following [1], [2], σ, $\vec{\alpha}$, and $\vec{\beta}$ are expressed by $\sigma = T^\top (K^{-1}A)^{j-1}\hat{U}$, $\vec{\alpha} = \sigma^{-1}(T^\top (K^{-1}A)^{j-2}\hat{\boldsymbol{r}})$, and $\vec{\beta} = \sigma^{-1}(T^\top \hat{\boldsymbol{u}}_j)$, respectively, where $T \equiv A^\top \tilde{R}_0$ is computed once and stored during the initialization.

At the polynomial step, the approximation and the corresponding residual are expressed by

$$\begin{aligned} \boldsymbol{x}_{k+\ell} &= \boldsymbol{x}' + K^{-1}\boldsymbol{p}', \quad \boldsymbol{r}_{k+\ell} = \boldsymbol{r}' - AK^{-1}\boldsymbol{p}', \\ \boldsymbol{p}' &= \sum_{j=1}^{\ell} \gamma_{j,k}(AK^{-1})^{j-1}\boldsymbol{r}'. \end{aligned} \tag{15}$$

Algorithm 5. AAI variant of IDR(s)stab(ℓ) with right preconditioning [1].

Select an initial guess $\mathbf{x}$ and an $(n \times s)$ matrix $\tilde{\mathbf{R}}_0$.
Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x},\ \mathbf{r} = [\mathbf{r}_0],\ \mathbf{T} = \mathbf{A}^\top \tilde{\mathbf{R}}_0$
Generate an initial $\hat{\mathbf{U}} = [\hat{\mathbf{U}}_0] = [\mathbf{K}^{-1}\mathbf{U}_0]$
While $\|\mathbf{r}_0\|_2 > tol$
% *The IDR step*
  For $j = 1, 2, \ldots, \ell$
    $\sigma = \mathbf{T}^\top \hat{\mathbf{U}}_{j-1}$
    if $j = 1$ then
      $\vec{\alpha} = \sigma^{-1}(\tilde{\mathbf{R}}_0^\top \mathbf{r}_0)$
    else
      $\vec{\alpha} = \sigma^{-1}(\mathbf{T}^\top \hat{\mathbf{r}}_{j-2})$
    end if
    $\hat{\mathbf{p}} = \hat{\mathbf{U}}_0 \vec{\alpha},\ \mathbf{x} = \mathbf{x} + \hat{\mathbf{p}},\ \mathbf{r}_0 = \mathbf{r}_0 - \mathbf{A}\hat{\mathbf{p}}$
    For $i = 1, 2, \ldots, j-2$
      $\mathbf{r}_i = \mathbf{r}_i - \mathbf{U}_{i-1}\vec{\alpha}$
    End for
    if $j = 1$ then
      $\hat{\mathbf{r}} = [\mathbf{K}^{-1}\mathbf{r}_0]$
    else
      $\hat{\mathbf{r}} = \hat{\mathbf{r}} - [\hat{\mathbf{U}}_1; \hat{\mathbf{U}}_2; \cdots; \hat{\mathbf{U}}_{j-1}]\vec{\alpha}$
      $\mathbf{r} = [\mathbf{r}; \mathbf{A}\hat{\mathbf{r}}_{j-2}],\ \hat{\mathbf{r}} = [\hat{\mathbf{r}}; \mathbf{K}^{-1}\mathbf{r}_{j-1}]$
    end if
    For $q = 1, 2, \ldots, s$
      if $q = 1$ then
        $\hat{\mathbf{u}} = \hat{\mathbf{r}},\ \mathbf{u} = [\mathbf{r}_2; \mathbf{r}_3; \cdots; \mathbf{r}_{j-2}]$
      else
        $\hat{\mathbf{u}} = [\hat{\mathbf{u}}_1; \hat{\mathbf{u}}_2; \cdots; \hat{\mathbf{u}}_j],\ \mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \cdots; \mathbf{u}_{j-2}]$
      end if
      $\vec{\beta} = \sigma^{-1}(\mathbf{T}^\top \hat{\mathbf{u}}_{j-1}),\ \hat{\mathbf{u}} = \hat{\mathbf{u}} - \hat{\mathbf{U}}\vec{\beta}$
      if $2 < j < \ell$ then
        $\mathbf{u} = \mathbf{u} - \mathbf{U}\vec{\beta}$
      end if
      $\hat{\mathbf{u}} = [\hat{\mathbf{u}}; \mathbf{K}^{-1}\mathbf{A}\hat{\mathbf{u}}_{j-1}]$
      if $j = 2$ then
        $\mathbf{u} = [\mathbf{A}\hat{\mathbf{u}}_1]$
        else if $2 < j < \ell$ then
          $\mathbf{u} = [\mathbf{u}; \mathbf{A}\hat{\mathbf{u}}_{j-1}]$
      end if
      Orthonormalize $\hat{\mathbf{u}}_j$ to the columns of $\hat{\mathbf{V}}_{j(:,1:q-1)}$
      $\hat{\mathbf{V}}_{(:,q)} = \hat{\mathbf{u}}$
      if $1 < j < \ell$ then
        $\mathbf{V}_{(:,q)} = \mathbf{u}$
      end if
    End for
    if $j < \ell$ then
      $\hat{\mathbf{U}} = \hat{\mathbf{V}}$
      and if $j > 1$ then
        $\mathbf{U} = \mathbf{V}$
    end if
  End for
  $\mathbf{r} = [\mathbf{r}; \mathbf{A}\hat{\mathbf{r}}_{\ell-1}]$
% *The polynomial step*
  $\vec{\gamma} = [\gamma_1; \gamma_2; \cdots; \gamma_\ell] = \arg\min_{\vec{\gamma}} \|\mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_\ell]\vec{\gamma}\|_2$
  $\hat{\mathbf{p}} = [\hat{\mathbf{r}}_0, \hat{\mathbf{r}}_1, \ldots, \hat{\mathbf{r}}_{\ell-1}]\vec{\gamma},\ \mathbf{x} = \mathbf{x} + \hat{\mathbf{p}},\ \mathbf{r}_0 = \mathbf{r}_0 - \mathbf{A}\hat{\mathbf{p}}$
  $\hat{\mathbf{U}} = [\hat{\mathbf{V}}_0 - \sum_{j=1}^{\ell} \gamma_j \hat{\mathbf{V}}_j]$
End while

Algorithm 6. AAI variant of IDR(s)stab(ℓ) with left preconditioning.

Select an initial guess $\mathbf{x}$ and an $(n \times s)$ matrix $\tilde{\mathbf{R}}_0$.
Compute $\mathbf{r}_0 = \mathbf{K}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}),\ \mathbf{r} = [\mathbf{r}_0],\ \mathbf{T} = \mathbf{A}^\top \mathbf{K}^{-\top} \tilde{\mathbf{R}}_0$
Generate an initial $\mathbf{U} = [\mathbf{U}_0]$
While $\|\mathbf{r}_0\|_2 > tol$
% *The IDR step*
  For $j = 1, 2, \ldots, \ell$
    $\sigma = \mathbf{T}^\top \mathbf{U}_{j-1}$
    if $j = 1$ then
      $\vec{\alpha} = \sigma^{-1}(\tilde{\mathbf{R}}_0^\top \mathbf{r}_0)$
    else
      $\vec{\alpha} = \sigma^{-1}(\mathbf{T}^\top \mathbf{r}_{j-2})$
    end if
    $\mathbf{p} = \mathbf{U}_0 \vec{\alpha},\ \mathbf{x} = \mathbf{x} + \mathbf{p},\ \mathbf{r}_0 = \mathbf{r}_0 - \mathbf{K}^{-1}\mathbf{A}\mathbf{p}$
    For $i = 1, 2, \ldots, j-2$
      $\mathbf{r}_i = \mathbf{r}_i - \mathbf{U}_{i+1}\vec{\alpha}$
    End for
    if $j > 1$ then
      $\mathbf{r} = [\mathbf{r}; \mathbf{K}^{-1}\mathbf{A}\mathbf{r}_{j-2}]$
    end if
    For $q = 1, 2, \ldots, s$
      if $q = 1$ then
        $\mathbf{u} = \mathbf{r}$
      else
        $\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \cdots; \mathbf{u}_j]$
      end if
      $\vec{\beta} = \sigma^{-1}(\mathbf{T}^\top \mathbf{u}_{j-1}),\ \mathbf{u} = \mathbf{u} - \mathbf{U}\vec{\beta},\ \mathbf{u} = [\mathbf{u}; \mathbf{K}^{-1}\mathbf{A}\mathbf{u}_{j-1}]$
      Orthonormalize $\mathbf{u}_j$ to the columns of $\mathbf{V}_{j(:,1:q-1)}$
      $\mathbf{V}_{(:,q)} = \mathbf{u}$
    End for
    if $j < \ell$ then
      $\mathbf{U} = \mathbf{V}$
    end if
  End for
  $\mathbf{r} = [\mathbf{r}; \mathbf{K}^{-1}\mathbf{A}\mathbf{r}_{\ell-1}]$
% *The polynomial step*
  $\vec{\gamma} = [\gamma_1; \gamma_2; \cdots; \gamma_\ell] = \arg\min_{\vec{\gamma}} \|\mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_\ell]\vec{\gamma}\|_2$
  $\mathbf{p} = [\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{\ell-1}]\vec{\gamma},\ \mathbf{x} = \mathbf{x} + \mathbf{p},\ \mathbf{r}_0 = \mathbf{r}_0 - \mathbf{K}^{-1}\mathbf{A}\mathbf{p}$
  $\mathbf{U} = [\mathbf{V}_0 - \sum_{j=1}^{\ell} \gamma_j \mathbf{V}_j]$
End while

Using $\hat{\boldsymbol{r}}' = K^{-1}\boldsymbol{r}'$, (15) is reformulated as

$$\boldsymbol{x}_{k+\ell} = \boldsymbol{x}' + \hat{\boldsymbol{p}}', \quad \boldsymbol{r}_{k+\ell} = \boldsymbol{r}' - A\hat{\boldsymbol{p}}',$$
$$\hat{\boldsymbol{p}}' = \sum_{j=1}^{\ell} \gamma_{j,k} (K^{-1}A)^{j-1} \hat{\boldsymbol{r}}',$$

where $A\hat{\boldsymbol{p}}'$ is obtained by explicitly multiplying $\hat{\boldsymbol{p}}'$ by A. Finally, the matrix $\hat{U}$ used at the next cycle is computed by

$$\hat{U} = \hat{V} - \sum_{j=1}^{\ell} \gamma_{j,k} (K^{-1}A)^j \hat{V}.$$

Algorithm 5 displays the AAI variant with the right preconditioning proposed in [1]. Note that $\mathbf{r}_j$, $\hat{\mathbf{r}}_j$, $\hat{\mathbf{U}}_j$, and $\mathbf{U}_j$ represent $(\mathbf{A}\mathbf{K}^{-1})^j \mathbf{r}_0$, $\mathbf{K}^{-1}\mathbf{r}_j$, $(\mathbf{K}^{-1}\mathbf{A})^j \hat{\mathbf{U}}_0$, and $\mathbf{A}\hat{\mathbf{U}}_{j+1}$, respectively.

The AAI variant with left preconditioning is displayed in Algorithm 6, which can be derived by applying the

TABLE I
COMPUTATIONAL COSTS FOR MVS, $K^{-1}\boldsymbol{v}$'S, AXPYS, AND DOTS OF IDR(s)STAB(ℓ), GBI-CGSTAB(s, ℓ), AND THE AAI VARIANT WITH RIGHT AND LEFT PRECONDITIONING PER CYCLE FOR $\ell > 1$.

Side	Solver	MVs	$K^{-1}\boldsymbol{v}$'s	AXPYs	DOTs
Right	IDR(s)stab(ℓ)	$\ell(s+1)$	$\ell(s+1)$	$\ell s(\ell+1)(s+1)+2\ell(s+1)$	$\ell s(2s+1)+\frac{1}{2}\ell(\ell+3)$
	GBi-CGSTAB(s, ℓ)	$\ell(s+1)$	$\ell(s+1)$	$\ell s(\ell+1)(s+1)-\ell(s^2-s-\frac{1}{2}\ell-\frac{3}{2})$	$\ell s(s+1)+\frac{1}{2}\ell(\ell+3)$
	AAI variant	$\ell(s+2)+1$	$\ell(s+1)$	$\ell s(\ell+1)(s+1)-s(\ell-1)(3s+1)+\frac{3}{2}\ell+\frac{1}{2}$	$\ell s(2s+1)+\frac{1}{2}\ell(\ell+3)$
Left	IDR(s)stab(ℓ)	$\ell(s+1)$	$\ell(s+1)$	$\frac{1}{2}\ell s(\ell+1)(s+1)+\ell(s^2+3s+2)$	$\ell s(2s+1)+\frac{1}{2}\ell(\ell+3)$
	GBi-CGSTAB(s, ℓ)	$\ell(s+1)$	$\ell(s+1)$	$\frac{1}{2}\ell s(\ell+1)(s+1)+\ell(2s+\frac{1}{2}\ell+\frac{3}{2})$	$\ell s(s+1)+\frac{1}{2}\ell(\ell+3)$
	AAI variant	$\ell(s+2)+1$	$\ell(s+2)+1$	$\frac{1}{2}\ell s(\ell+1)(s+1)+\frac{3}{2}\ell+s+\frac{1}{2}$	$\ell s(2s+1)+\frac{1}{2}\ell(\ell+3)$

algorithm of the AAI variant to the left preconditioned system (3). Here, $\boldsymbol{r}_0$ is set to $K^{-1}(\boldsymbol{b}-A\boldsymbol{x}_0)$, and $\tilde{A}$, $\tilde{\boldsymbol{r}}$, and $\tilde{U}$ are replaced by $K^{-1}A$, $\boldsymbol{r}$, and U, respectively. σ, $\vec{\alpha}$, and $\vec{\beta}$ are expressed by $\sigma = T^\top(K^{-1}A)^{j-1}U$, $\vec{\alpha} = \sigma^{-1}(T^\top(K^{-1}A)^{j-2}\boldsymbol{r})$, and $\vec{\beta} = \sigma^{-1}(T^\top \boldsymbol{u}_j)$, respectively, where $T \equiv A^\top K^{-\top}\tilde{R}_0$. In Algorithm 6, $\mathbf{r}_j$ and $\mathbf{U}_j$ represent $(\mathbf{K}^{-1}\mathbf{A})^j\mathbf{r}_0$ and $(\mathbf{K}^{-1}\mathbf{A})^j\mathbf{U}_0$, respectively.

D. Computational costs

The right preconditioned IDRStab requires additional vector updates to obtain the auxiliary matrices and vectors, such as $\hat{U} = K^{-1}U$ and $\hat{\boldsymbol{r}} = K^{-1}\boldsymbol{r}$, but the left preconditioned IDRStab requires no additional vector updates.

Table I summarizes the computational costs of IDR(s)stab(ℓ), GBi-CGSTAB(s, ℓ), and the AAI variant with right and left preconditioning per cycle for $\ell > 1$. A matrix–vector multiplication by A and a multiplication by K^{-1} are indicated by MV and $K^{-1}\boldsymbol{v}$, respectively. AXPY stands for a vector update of the form $a\boldsymbol{x}+\boldsymbol{y}$ with a scalar a and n-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. A computation of the form $a\boldsymbol{x}$ or $\boldsymbol{x}+\boldsymbol{y}$ is counted as $\frac{1}{2}$ AXPY. DOT denotes an inner product between two n-dimensional vectors. Table II shows the computational costs for (s, ℓ)=(4, 4).

Following [1], [13], the columns of $(K^{-1}A)^j\hat{U}$ of IDR(s)stab(ℓ) and the AAI variant with right preconditioning are orthonormalized for numerical stability (see lines 14 and 20 in Algorithms 1 and 5, respectively). Similarly, for IDR(s)stab(ℓ) and the AAI variant with left preconditioning, the columns of $(K^{-1}A)^jU$ are orthonormalized (see lines 12 and 16 in Algorithms 2 and 6, respectively). Table III summarizes the computational costs when using Gram–Schmidt orthonormalization.

From Tables I–III, we can observe the following.

- **Comparison between the variants of IDRStab.** The numbers of MVs and $K^{-1}\boldsymbol{v}$'s of the preconditioned GBi-CGSTAB(s, ℓ) are the same as those of the preconditioned IDR(s)stab(ℓ), but the numbers of AXPYs and DOTs of the preconditioned GBi-CGSTAB(s, ℓ) are much less than those of the preconditioned IDR(s)stab(ℓ). The AAI variant with preconditioning requires $\ell+1$ additional MVs per cycle, and in the case of left preconditioning, also requires $\ell+1$ additional $K^{-1}\boldsymbol{v}$'s. However, the number of AXPYs of the preconditioned AAI variant is much less than that of the preconditioned IDR(s)stab(ℓ).
- **Comparison between right and left preconditioning.** The computational costs of IDR(s)stab(ℓ) and GBi-CGSTAB(s, ℓ) with left preconditioning are much less than those with right preconditioning. It is therefore expected that the left preconditioned algorithm is more efficient than the right preconditioned algorithm in terms of computation time. On the other hand, the AAI variant with left preconditioning requires fewer AXPYs than it does with right preconditioning, but it requires additional $K^{-1}\boldsymbol{v}$'s.

TABLE II
COMPUTATIONAL COSTS FOR MVS, $K^{-1}\boldsymbol{v}$'S, AXPYS, AND DOTS OF IDR(s)STAB(ℓ), GBI-CGSTAB(s, ℓ), AND THE AAI VARIANT WITH RIGHT AND LEFT PRECONDITIONING PER CYCLE FOR (s, ℓ) = (4, 4).

Side	Solver	MVs	$K^{-1}\boldsymbol{v}$'s	AXPYs	DOTs
Right	IDR(4)stab(4)	20	20	440	158
	GBi-CGSTAB(4, 4)	20	20	366	94
	AAI variant	25	20	250.5	158
Left	IDR(4)stab(4)	20	20	320	158
	GBi-CGSTAB(4, 4)	20	20	246	94
	AAI variant	25	25	210.5	158

TABLE III
COMPUTATIONAL COSTS FOR AXPYS AND DOTS OF GRAM–SCHMIDT ORTHONORMALIZATION IN IDR(s)STAB(ℓ) AND THE AAI VARIANT WITH RIGHT AND LEFT PRECONDITIONING PER CYCLE FOR $\ell > 1$.

Side	Solver	AXPYs	DOTs
Right	IDR(s)stab(ℓ)	$\frac{1}{2}\ell s^2(\ell+2)$	$\frac{1}{2}\ell s(s+1)$
	AAI variant	$\frac{1}{2}s^2(\ell^2+1)$	$\frac{1}{2}\ell s(s+1)$
Left	IDR(s)stab(ℓ)	$\frac{1}{4}\ell s^2(\ell+3)$	$\frac{1}{2}\ell s(s+1)$
	AAI variant	$\frac{1}{4}\ell s^2(\ell+3)$	$\frac{1}{2}\ell s(s+1)$

III. NUMERICAL EXPERIMENTS

In this section, we present numerical experiments on model problems with sparse nonsymmetric matrices to show the difference in convergence between IDR(s)stab(ℓ), GBi-CGSTAB(s, ℓ), and the AAI variant with right and left preconditioning. We use the test matrices pde2961, airfoil_2d, sherman5, orsreg_1, memplus, and wang4 from the University of Florida Sparse Matrix Collection [3]. Table IV shows for each matrix the dimension n, the number of nonzero entries (abbreviated as nnz), the average nnz per row (abbreviated as $Ave.\ nnz$), the condition number (indicated by $cond(A)$), and the discipline of the application that produced that matrix.

Numerical calculations were carried out in double-precision floating-point arithmetic on a PC (Intel Core i7 2.67 GHz CPU) with a GNU C++ 4.5.2 compiler. The right-hand side vector $\boldsymbol{b}$ is obtained by substituting $\boldsymbol{x}^* \equiv (1, 1, \ldots, 1)^\top$ into the equation $\boldsymbol{b} = A\boldsymbol{x}^*$. The iterations were started

TABLE IV
CHARACTERISTIC OF TEST MATRICES.

Matrices	n	nnz	Ave. nnz	cond(A)	Application discipline
pde2961	2961	14585	4.93	6.4E+02	Partial differential equations
airfoil_2d	14214	259688	18.3	1.8E+06	Computational fluid dynamics
sherman5	3312	20793	6.28	1.9E+05	Computational fluid dynamics
orsreg_1	2205	14133	6.41	6.7E+04	Computational fluid dynamics
memplus	17758	99147	5.58	1.3E+05	Circuit simulation
wang4	26068	177196	6.80	4.0E+05	Semiconductor device

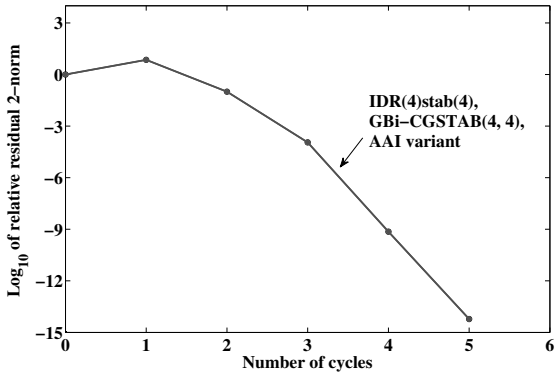

Fig. 1. Convergence histories plotted with the number of cycles of IDR(4)stab(4), GBi-CGSTAB(4, 4), and the AAI variant with right preconditioning for airfoil_2d.

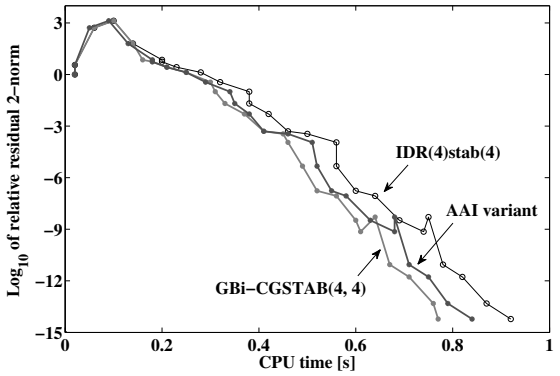

Fig. 3. Convergence histories plotted with the computation time of IDR(4)stab(4), GBi-CGSTAB(4, 4), and the AAI variant with right preconditioning for airfoil_2d.

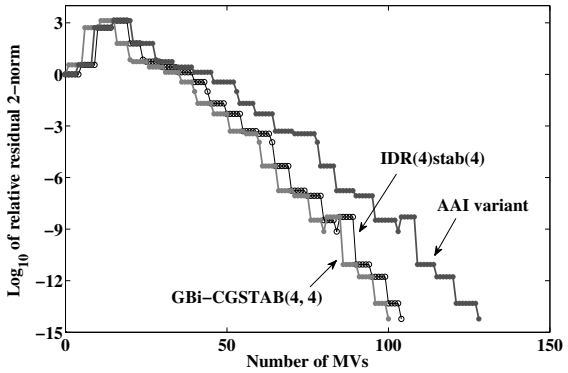

Fig. 2. Convergence histories plotted with the number of MVs of IDR(4)stab(4), GBi-CGSTAB(4, 4), and the AAI variant with right preconditioning for airfoil_2d.

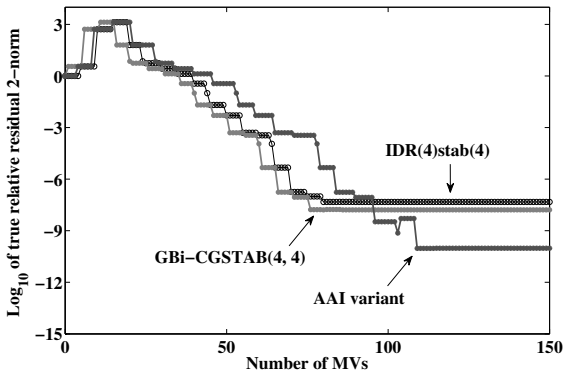

Fig. 4. Convergence histories of the true relative residual norms of IDR(4)stab(4), GBi-CGSTAB(4, 4), and the AAI variant with right preconditioning for airfoil_2d.

with $\mathbf{0}$, and were stopped when the relative residual 2-norms (i.e., $\|\boldsymbol{r}_k\|_2/\|\boldsymbol{r}_0\|_2$) become 10^{-12}. The columns of $\tilde{R}_0$ were given by the orthonormalization of s real random vectors in the interval (0, 1). The columns of the initial matrices for IDRStab with right, with left, and without preconditioning were given by the orthonormalization of the columns of $\hat{U} = [\hat{\boldsymbol{r}}_0, K^{-1}A\hat{\boldsymbol{r}}_0, \ldots, (K^{-1}A)^{s-1}\hat{\boldsymbol{r}}_0]$, $U = [\boldsymbol{r}_0, K^{-1}A\boldsymbol{r}_0, \ldots, (K^{-1}A)^{s-1}\boldsymbol{r}_0]$, and $U = [\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{s-1}\boldsymbol{r}_0]$, respectively. The parameters (s, ℓ) were set to (4, 4), (8, 2), and (2, 8). ILU(0) [10, Algorithm 10.4] was used as the preconditioner.

Figures 1–4 display the convergence histories of the right preconditioned IDRStab with (s, ℓ)=(4, 4) for airfoil_2d. The numbers of cycles and MVs, and the computation time are plotted on the horizontal axis in Figures 1, 2, and 3, respectively; the $\log_{10}$ of the relative residual 2-norm is plotted on the vertical axis in the figures. Figure 4 displays the number of MVs on the horizontal axis versus the $\log_{10}$ of the explicitly computed relative residual 2-norm ($\|\boldsymbol{b} - A\boldsymbol{x}_k\|_2/\|\boldsymbol{b}\|_2$) on the vertical axis, respectively. Tables V–VII show the numbers of cycles and MVs (the number of multiplications by K^{-1}), the computation times, and the explicitly computed relative residual 2-norms at termination of IDRStab without and with (right and left) preconditioning, which are abbreviated as "Cycles", "MVs ($K^{-1}\boldsymbol{v}$'s)", "Time [s]", and "True res.", respectively. We did not include the times for constructing the ILU(0) preconditioner.

From Figures 1–4 and Tables V–VII, we can observe the following.

- **Comparison between the variants of IDRStab.** The convergence for all variants of IDRStab is enhanced by preconditioning. The numbers of cycles required

TABLE V
NUMBERS OF CYCLES AND MVS (NUMBER OF MULTIPLICATIONS BY K^{-1}), COMPUTATION TIMES, AND TRUE RELATIVE RESIDUAL NORMS OF IDR(s)STAB(ℓ), GBI-CGSTAB(s, ℓ), AND THE AAI VARIANT WITHOUT AND WITH (RIGHT AND LEFT) PRECONDITIONING FOR (s, ℓ)=(4, 4).

Matrices	Preconditioning	Solver	Cycles	MVs ($K^{-1}\boldsymbol{v}$'s)	Time [s]	True res.
pde2961	-	IDR(4)stab(4)	17	345	0.36	1.26E−10
		GBi-CGSTAB(4, 4)	17	341	0.19	3.96E−11
		AAI variant	16	404	0.29	8.25E−13
	Right ILU(0)	IDR(4)stab(4)	4	85 (84)	0.12	9.92E−13
		GBi-CGSTAB(4, 4)	4	81 (80)	0.06	1.54E−12
		AAI variant	4	104 (84)	0.10	1.58E−13
	Left ILU(0)	IDR(4)stab(4)	4	85 (85)	0.11	2.41E−11
		GBi-CGSTAB(4, 4)	4	81 (81)	0.04	1.46E−11
		AAI variant	5	129 (129)	0.11	3.84E−14
airfoil_2d	-	IDR(4)stab(4)	220	4405	30.5	1.44E+00
		GBi-CGSTAB(4, 4)	205	4101	15.1	2.42E−08
		AAI variant	239	5979	31.6	7.72E−10
	Right ILU(0)	IDR(4)stab(4)	5	105 (104)	0.92	4.73E−08
		GBi-CGSTAB(4, 4)	5	101 (100)	0.77	1.64E−08
		AAI variant	5	129 (104)	0.84	9.56E−11
	Left ILU(0)	IDR(4)stab(4)	4	85 (85)	0.78	8.94E−08
		GBi-CGSTAB(4, 4)	4	81 (81)	0.55	9.74E−06
		AAI variant	4	104 (104)	0.66	8.19E−10
sherman5	-	IDR(4)stab(4)	114	2285	2.74	2.73E−10
		GBi-CGSTAB(4, 4)	116	2321	1.31	5.83E−13
		AAI variant	123	3079	2.66	1.83E−13
	Right ILU(0)	IDR(4)stab(4)	3	65 (64)	0.09	3.30E−13
		GBi-CGSTAB(4, 4)	3	61 (60)	0.06	3.44E−14
		AAI variant	3	79 (64)	0.07	5.94E−16
	Left ILU(0)	IDR(4)stab(4)	3	65 (65)	0.08	3.02E−12
		GBi-CGSTAB(4, 4)	3	61 (61)	0.05	1.46E−13
		AAI variant	3	79 (79)	0.08	7.04E−16
orsreg_1	-	IDR(4)stab(4)	25	505	0.38	1.38E−07
		GBi-CGSTAB(4, 4)	20	401	0.16	3.07E−11
		AAI variant	25	629	0.35	2.14E−12
	Right ILU(0)	IDR(4)stab(4)	5	105 (104)	0.12	3.56E−11
		GBi-CGSTAB(4, 4)	5	101 (100)	0.07	2.28E−11
		AAI variant	5	129 (104)	0.09	1.30E−12
	Left ILU(0)	IDR(4)stab(4)	5	105 (105)	0.09	4.51E−11
		GBi-CGSTAB(4, 4)	5	101 (101)	0.05	2.39E−11
		AAI variant	5	129 (129)	0.10	1.02E−12
memplus	-	IDR(4)stab(4)	97	1945	13.4	1.26E−08
		GBi-CGSTAB(4, 4)	94	1881	7.48	1.44E−11
		AAI variant	124	3104	15.3	8.06E−13
	Right ILU(0)	IDR(4)stab(4)	23	465 (464)	3.99	2.43E−10
		GBi-CGSTAB(4, 4)	24	481 (480)	2.47	8.07E−12
		AAI variant	25	629 (504)	3.64	9.35E−13
	Left ILU(0)	IDR(4)stab(4)	26	525 (525)	4.31	5.21E−12
		GBi-CGSTAB(4, 4)	24	481 (481)	2.03	1.18E−12
		AAI variant	28	704 (704)	4.40	1.13E−13
wang4	-	IDR(4)stab(4)	34	685	5.74	2.97E−09
		GBi-CGSTAB(4, 4)	33	661	3.82	1.69E−11
		AAI variant	34	854	5.12	7.49E−13
	Right ILU(0)	IDR(4)stab(4)	5	105 (104)	1.53	5.39E−12
		GBi-CGSTAB(4, 4)	5	101 (100)	0.90	1.72E−12
		AAI variant	5	129 (104)	1.06	4.09E−15
	Left ILU(0)	IDR(4)stab(4)	5	105 (105)	1.02	9.81E−12
		GBi-CGSTAB(4, 4)	5	101 (101)	0.62	8.34E−12
		AAI variant	5	129 (129)	0.94	5.14E−15

for successful convergence of IDR(s)stab(ℓ), GBi-CGSTAB(s, ℓ), and the AAI variant with preconditioning are almost the same. The computation times for GBi-CGSTAB(s, ℓ) with preconditioning are shorter than those for IDR(s)stab(ℓ) and the AAI variant with preconditioning. The computation times for the right preconditioned AAI variant are slightly shorter than those for the right preconditioned IDR(s)stab(ℓ); the computational costs for the additional MVs in the AAI variant with right preconditioning are compensated for by a reduction in the number of AXPYs. The approximate solutions obtained by IDR(s)stab(ℓ) and GBi-

TABLE VI
NUMBERS OF CYCLES AND MVS (NUMBER OF MULTIPLICATIONS BY K^{-1}), COMPUTATION TIMES, AND TRUE RELATIVE RESIDUAL NORMS OF IDR(s)STAB(ℓ), GBI-CGSTAB(s, ℓ), AND THE AAI VARIANT WITHOUT AND WITH (RIGHT AND LEFT) PRECONDITIONING FOR (s, ℓ)=(8, 2).

Matrices	Preconditioning	Solver	Cycles	MVs ($K^{-1}\boldsymbol{v}$'s)	Time [s]	True res.
pde2961	-	IDR(8)stab(2)	17	315	0.42	5.81E−11
		GBi-CGSTAB(8, 2)	16	289	0.19	6.76E−12
		AAI variant	16	344	0.34	7.41E−13
	Right ILU(0)	IDR(8)stab(2)	5	99 (98)	0.15	2.79E−12
		GBi-CGSTAB(8, 2)	5	91 (90)	0.08	8.17E−14
		AAI variant	5	113 (98)	0.11	4.10E−15
	Left ILU(0)	IDR(8)stab(2)	5	99 (99)	0.14	1.36E−11
		GBi-CGSTAB(8, 2)	5	91 (91)	0.07	6.05E−12
		AAI variant	5	113 (113)	0.11	4.91E−15
airfoil_2d	-	IDR(8)stab(2)	235	4239	39.3	4.92E−04
		GBi-CGSTAB(8, 2)	223	4015	19.9	1.55E−08
		AAI variant	271	5699	39.9	2.14E−08
	Right ILU(0)	IDR(8)stab(2)	6	117 (116)	1.49	1.21E−08
		GBi-CGSTAB(8, 2)	6	109 (108)	0.88	1.05E−08
		AAI variant	6	134 (116)	1.26	8.36E−11
	Left ILU(0)	IDR(8)stab(2)	5	99 (99)	1.09	2.48E−09
		GBi-CGSTAB(8, 2)	5	91 (91)	0.57	8.69E−09
		AAI variant	5	113 (113)	1.03	8.12E−11
sherman5	-	IDR(8)stab(2)	136	2457	4.02	2.97E−10
		GBi-CGSTAB(8, 2)	126	2269	1.78	8.69E−13
		AAI variant	134	2822	3.29	1.87E−13
	Right ILU(0)	IDR(8)stab(2)	3	63 (62)	0.12	6.35E−15
		GBi-CGSTAB(8, 2)	3	55 (54)	0.06	3.08E−15
		AAI variant	3	71 (62)	0.08	5.41E−16
	Left ILU(0)	IDR(8)stab(2)	3	63 (63)	0.09	3.42E−15
		GBi-CGSTAB(8, 2)	3	55 (55)	0.05	1.94E−15
		AAI variant	3	71 (71)	0.08	5.23E−16
orsreg_1	-	IDR(8)stab(2)	23	423	0.47	4.20E−11
		GBi-CGSTAB(8, 2)	23	415	0.21	1.93E−12
		AAI variant	25	533	0.34	1.83E−12
	Right ILU(0)	IDR(8)stab(2)	6	117 (116)	0.14	2.49E−12
		GBi-CGSTAB(8, 2)	6	109 (108)	0.08	2.84E−12
		AAI variant	6	134 (116)	0.12	8.15E−13
	Left ILU(0)	IDR(8)stab(2)	5	99 (99)	0.11	1.87E−11
		GBi-CGSTAB(8, 2)	5	91 (91)	0.05	1.23E−11
		AAI variant	5	113 (113)	0.10	1.22E−11
memplus	-	IDR(8)stab(2)	91	1647	16.6	5.99E−10
		GBi-CGSTAB(8, 2)	88	1585	7.29	1.79E−12
		AAI variant	106	2234	14.1	9.02E−13
	Right ILU(0)	IDR(8)stab(2)	25	459 (458)	6.18	1.55E−10
		GBi-CGSTAB(8, 2)	25	451 (450)	3.07	4.37E−13
		AAI variant	27	575 (494)	5.19	7.63E−13
	Left ILU(0)	IDR(8)stab(2)	25	459 (459)	5.14	4.99E−09
		GBi-CGSTAB(8, 2)	27	487 (487)	2.90	1.07E−12
		AAI variant	30	638 (638)	5.45	5.49E−13
wang4	-	IDR(8)stab(2)	36	657	10.3	1.38E−08
		GBi-CGSTAB(8, 2)	36	649	4.46	7.91E−13
		AAI variant	36	764	8.64	1.28E−13
	Right ILU(0)	IDR(8)stab(2)	5	99 (98)	1.70	5.02E−13
		GBi-CGSTAB(8, 2)	5	91 (90)	0.92	2.92E−13
		AAI variant	5	113 (98)	1.48	7.39E−15
	Left ILU(0)	IDR(8)stab(2)	6	117 (117)	1.90	3.92E−11
		GBi-CGSTAB(8, 2)	6	109 (109)	0.94	7.70E−13
		AAI variant	6	134 (134)	1.68	4.52E−15

CGSTAB(s, ℓ) with preconditioning often, especially for larger ℓ, do not attain the required accuracy, but those obtained by the preconditioned AAI variant are more accurate or have the same accuracy.

- **Comparison between right and left preconditioning.** The computation times for IDR(s)stab(ℓ) and GBi-CGSTAB(s, ℓ) with left preconditioning are shorter than those with right preconditioning. The computation times for the left preconditioned AAI variant are comparable to those for the right preconditioned one. We note that the total computational costs of the left preconditioned AAI variant may be almost the same as those of the

TABLE VII
NUMBERS OF CYCLES AND MVS (NUMBER OF MULTIPLICATIONS BY K^{-1}), COMPUTATION TIMES, AND TRUE RELATIVE RESIDUAL NORMS OF IDR(s)STAB(ℓ), GBI-CGSTAB(s, ℓ), AND THE AAI VARIANT WITHOUT AND WITH (RIGHT AND LEFT) PRECONDITIONING FOR (s, ℓ)=(2, 8).

Matrices	Preconditioning	Solver	Cycles	MVs ($K^{-1}\boldsymbol{v}$'s)	Time [s]	True res.
pde2961	-	IDR(2)stab(8)	17	411	0.42	2.82E−08
		GBi-CGSTAB(2, 8)	16	385	0.19	1.25E−08
		AAI variant	16	530	0.35	1.18E−13
	Right ILU(0)	IDR(2)stab(8)	4	99 (98)	0.14	9.28E−10
		GBi-CGSTAB(2, 8)	4	97 (96)	0.09	5.47E−11
		AAI variant	4	134 (98)	0.14	1.59E−14
	Left ILU(0)	IDR(2)stab(8)	4	99 (99)	0.11	9.76E−08
		GBi-CGSTAB(2, 8)	4	97 (97)	0.07	1.86E−09
		AAI variant	4	134 (134)	0.11	3.92E−13
airfoil_2d	-	IDR(2)stab(8)	194	4659	30.4	5.22E−01
		GBi-CGSTAB(2, 8)	197	4729	21.3	8.43E−07
		AAI variant	216	7130	35.3	2.43E−09
	Right ILU(0)	IDR(2)stab(8)	5	123 (122)	1.09	1.85E−02
		GBi-CGSTAB(2, 8)	5	121 (120)	0.98	8.50E−06
		AAI variant	5	167 (122)	1.07	1.56E−10
	Left ILU(0)	IDR(2)stab(8)	4	99 (99)	0.88	6.11E−03
		GBi-CGSTAB(2, 8)	4	97 (97)	0.56	1.28E−03
		AAI variant	4	134 (134)	0.98	1.43E−10
sherman5	-	IDR(2)stab(8)	100	2403	2.77	6.02E−09
		GBi-CGSTAB(2, 8)	108	2593	1.89	9.37E−12
		AAI variant	121	3995	3.19	8.20E−13
	Right ILU(0)	IDR(2)stab(8)	3	75 (74)	0.14	7.69E−12
		GBi-CGSTAB(2, 8)	3	73 (72)	0.07	1.31E−10
		AAI variant	3	101 (74)	0.11	6.77E−16
	Left ILU(0)	IDR(2)stab(8)	3	75 (75)	0.09	1.43E−10
		GBi-CGSTAB(2, 8)	3	73 (73)	0.05	1.27E−10
		AAI variant	3	101 (101)	0.10	1.49E−15
orsreg_1	-	IDR(2)stab(8)	25	603	0.49	1.55E−04
		GBi-CGSTAB(2, 8)	21	505	0.23	1.18E−06
		AAI variant	25	827	0.43	2.63E−12
	Right ILU(0)	IDR(2)stab(8)	4	99 (98)	0.11	4.39E−08
		GBi-CGSTAB(2, 8)	4	97 (96)	0.08	8.81E−09
		AAI variant	4	134 (98)	0.10	3.22E−12
	Left ILU(0)	IDR(2)stab(8)	4	99 (99)	0.08	6.36E−08
		GBi-CGSTAB(2, 8)	4	97 (97)	0.05	4.84E−09
		AAI variant	4	134 (134)	0.08	3.09E−12
memplus	-	IDR(2)stab(8)	126	3027	19.6	2.05E−04
		GBi-CGSTAB(2, 8)	105	2521	10.1	2.64E−07
		AAI variant	115	3797	17.1	2.37E−13
	Right ILU(0)	IDR(2)stab(8)	23	555 (554)	5.83	6.89E−06
		GBi-CGSTAB(2, 8)	23	553 (552)	3.74	4.21E−10
		AAI variant	23	761 (554)	4.37	2.62E−13
	Left ILU(0)	IDR(2)stab(8)	23	555 (555)	4.33	1.87E−07
		GBi-CGSTAB(2, 8)	25	601 (601)	3.21	1.16E−09
		AAI variant	25	827 (827)	4.82	5.98E−13
wang4	-	IDR(2)stab(8)	42	1011	7.89	5.51E−03
		GBi-CGSTAB(2, 8)	34	817	4.72	1.12E−09
		AAI variant	36	1190	6.40	5.76E−13
	Right ILU(0)	IDR(2)stab(8)	4	99 (98)	1.48	1.04E−09
		GBi-CGSTAB(2, 8)	4	97 (96)	0.94	1.00E−09
		AAI variant	4	134 (98)	1.35	4.89E−14
	Left ILU(0)	IDR(2)stab(8)	4	99 (99)	1.09	1.96E−07
		GBi-CGSTAB(2, 8)	4	97 (97)	0.74	6.86E−10
		AAI variant	4	134 (134)	1.11	3.46E−12

right preconditioned one, because the computational costs for $K^{-1}\boldsymbol{v}$ are not expensive when using the ILU(0) preconditioner for sparse matrices. The right and left preconditioned algorithms generate different residuals $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$ and $\boldsymbol{r}_k = K^{-1}(\boldsymbol{b} - A\boldsymbol{x}_k)$, respectively, which are used in the stopping criterion. However, for each variant of IDRStab, the attainable accuracy of the approximate solutions obtained by the right preconditioned algorithm is comparable to that obtained by the left preconditioned algorithm.

IV. Concluding remarks

In this paper, we have illustrated the right and left preconditioned algorithms of IDR(s)stab(ℓ), GBi-CGSTAB(s, ℓ), and the AAI variant. Numerical experiments show the differences in their convergence when using the ILU(0) preconditioner. The numbers of cycles required for successful convergence of the variants of IDRStab with preconditioning are almost the same, but GBi-CGSTAB(s, ℓ) and the AAI variant with preconditioning are superior to the others in terms of the computation time and the accuracy of the approximate solutions, respectively. For IDR(s)stab(ℓ) and GBi-CGSTAB(s, ℓ), the left preconditioned algorithm is often more efficient than the right one in terms of the computation time. The AAI variant with right preconditioning, when applied to a sparse linear system, is as efficient as that with left preconditioning.

Acknowledgment

The authors would like to thank Dr. Gerard L. G. Sleijpen (Utrecht University) and Dr. Kees Vuik (Delft University of Technology) for their helpful advice for this work. The authors would like to also thank the reviewers for their careful reading and constructive comments.

References

[1] K. Aihara, K. Abe and E. Ishiwata, "A variant of IDRstab with reliable update strategies for solving sparse linear systems," *J. Comput. Appl. Math.*, **259** (2014), pp. 244-258.

[2] K. Aihara, K. Abe and E. Ishiwata, "An alternative implementation of the IDRstab method saving vector updates," *JSIAM Lett.*, **3** (2011), pp. 69-72.

[3] T. Davis, The University of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices/

[4] R. Fletcher, "Conjugate gradient methods for indefinite systems," *Lecture Notes in Math.*, **506** (1976), pp. 73-89.

[5] S. Fujino, T. Sekimoto and K. Murakami, "A rich variety of role of preconditioning for hybrid-typed IDR(s) and BiCGStab(L) methods," *IPSJ SIG Technical Report*, 2012-HPC-133, pp. 1-6, 2012. (in Japanese)

[6] O. Kardani, A. V. Lyamin and K. Krabbenhøft, "A Comparative Study of Preconditioning Techniques for Large Sparse Systems Arising in Finite Element Limit Analysis," *IAENG Int. J. Appl. Math.*, **43** (2013), pp. 195-203.

[7] C. Lanczos, "Solution of systems of linear equations by minimized iterations," *J. Research Nat. Bur. Standards*, **49** (1952), pp. 33-53.

[8] J.A. Meijerink and H.A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," *Math. Comp.*, **31** (1977), pp. 148-162.

[9] O. Rendel, A. Rizvanolli and J.-P.M. Zemke, "IDR: A new generation of Krylov subspace methods?," *Linear Algebra Appl.*, **439** (2013), pp. 1040-1061.

[10] Y. Saad, "Iterative Methods for Sparse Linear Systems," 2nd edition, SIAM, Philadelphia, 2003.

[11] G.L.G. Sleijpen and D.R. Fokkema, "BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum," *Electron. Trans. Numer. Anal.*, **1** (1993), pp. 11-32.

[12] G.L.G. Sleijpen, P. Sonneveld and M.B. van Gijzen, "Bi-CGSTAB as an induced dimension reduction method," *Appl. Numer. Math.*, **60** (2010), pp. 1100-1114.

[13] G.L.G. Sleijpen and M.B. van Gijzen, "Exploiting BiCGstab(ℓ) strategies to induce dimension reduction," *SIAM J. Sci. Comput.*, **32** (2010), pp. 2687-2709.

[14] 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** (2008), pp. 1035-1062.

[15] M. Tanio and M. Sugihara, "GBi-CGSTAB(s, L): IDR(s) with higher-order stabilization polynomials," *J. Comput. Appl. Math.*, **235** (2010), pp. 765-784.

[16] M. ur Rehman, C. Vuik and G. Segal, "Preconditioners for the Steady Incompressible Navier-Stokes Problem," *IAENG Int. J. Appl. Math.*, **38** (2008), pp. 223-232.

[17] 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. Statist. Comput.*, **13** (1992), pp. 631-644.