Journal of Computational and Applied Mathematics 259 (2014) 244–258

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

# A variant of IDRstab with reliable update strategies for solving sparse linear systems

Kensuke Aihara [a,*], Kuniyoshi Abe [b], Emiko Ishiwata [c]

[a] Graduate School of Science, Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162-8601, Japan
[b] Faculty of Economics and Information, Gifu Shotoku University, 1-38 Nakauzura, Gifu 500-8288, Japan
[c] Department of Mathematical Information Science, Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo 162-8601, Japan

ARTICLE INFO

*Article history:*
Received 30 October 2012
Received in revised form 21 August 2013

*Keywords:*
Linear systems
Induced dimension reduction
IDRstab method
Residual gap

ABSTRACT

The IDRStab method is often more effective than the IDR($s$) method and the BiCGstab($\ell$) method for solving large nonsymmetric linear systems. IDRStab can have a large so-called residual gap: the convergence of recursively computed residual norms does not coincide with that of explicitly computed residual norms because of the influence of rounding errors. We therefore propose an alternative recursion formula for updating the residuals to narrow the residual gap. The formula requires extra matrix–vector multiplications, but we reduce total computational costs by giving an alternative implementation which reduces the number of vector updates. Numerical experiments show that the alternative recursion formula reliably reduces the residual gap, and that our proposed variant of IDRStab is effective for sparse linear systems.



## 1. Introduction

The IDR($s$) method [1], which is based on the Induced Dimension Reduction (IDR) principle, has been proposed for solving large nonsymmetric linear systems $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{C}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{C}^n$. IDR(1) is mathematically equivalent to the Bi-Conjugate Gradient STABilized (BiCGSTAB) method [2]; IDR($s$) with $s > 1$ can be considered as BiCGSTAB with an $s$-dimensional initial shadow residual [3,4].

IDR($s$) uses stabilizing polynomials of degree one as are used in BiCGSTAB. Real stabilizing polynomials of degree one often cause numerical instabilities if $A \in \mathbb{R}^{n\times n}$ has large non-real eigenvalues close to the imaginary axis. To overcome this problem, the GBi-CGSTAB($s$, $L$) method [4] and the IDRstab method [5] have independently been developed.[1] These algorithms combine IDR($s$) with higher order stabilizing polynomials. IDRStab with $s = 1$ is mathematically equivalent to the BiCGstab($\ell$) method [7], and in the case of $\ell = 1$, it simplifies to IDR($s$). Although GBi-CGSTAB($s$, $L$) and IDRstab differ in the implementation, they are mathematically equivalent. In this paper, we focus specifically on an IDRstab implementation.

For all Krylov subspace methods in finite precision arithmetic that do not compute the residuals as $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k$ but use some recursion formula, the *residual gap* is defined as the difference between the recursively computed residual $\boldsymbol{r}_k$ and the explicitly computed residual (referred to as *true residual*) $\boldsymbol{b} - A\boldsymbol{x}_k$. The residual gap will most probably be non-zero because of rounding errors. A large residual gap almost certainly affects the ultimately attainable accuracy in such a manner that it is not possible to compute an approximate solution $\boldsymbol{x}_k$ that has a high tolerance. The large intermediate residual and

* Corresponding author. Tel.: +81 3 3260 4271.
*E-mail address:* j1411701@ed.tus.ac.jp (K. Aihara).

[1] The convention to call all these methods "IDRStab" has been suggested in [6].

0377-0427/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cam.2013.08.028

approximation norms are known to affect the ultimately attainable accuracy [8,9,14]. For improving the attainable accuracy, it has been proposed in [8] that the approximations and residuals are updated by an alternative formula in which groups of updates for the approximations and residuals are cumulated strategically. This is referred to as a strategy of *group-wise* update, and we briefly explain it in the Appendix. The influence of rounding errors, which arise from the matrix–vector multiplications, on the residual gap has been analyzed in [10].

The number of matrix–vector multiplications (MVs) required for successful convergence of IDRstab is often less than those of IDR($s$) and BiCGstab($\ell$), but IDRstab sometimes has a large residual gap. In this paper, we introduce an alternative recursion formula for updating the residuals to reduce the residual gap. The formulation reliably reduces the residual gap, but requires some extra MVs. In order to reduce the total computational costs, we give an alternative implementation of IDRstab in which the number of vector updates is reduced.

Numerical experiments demonstrate that the residual gap of our proposed variant is small compared with that of the original IDRstab variant. To understand why the residual gap can be narrowed, the reliability of the alternative recursion formula for updating the residuals is investigated by numerical experiments. We reveal the difference between the original IDRstab and our proposed variant using the strategy of group-wise update described in the Appendix.

This paper is organized as follows. The original IDRstab method is described in the next section. In Section 3, we derive an alternative implementation of IDRstab. In Section 4, we discuss the reliability of the alternative recursion formula for avoiding a large residual gap. Numerical experiments demonstrate that our proposed variant is effective for sparse linear systems in Section 5. We also derive the algorithm of our variant with preconditioning and present some numerical results. Concluding remarks are given in Section 6. In the Appendix, we present some numerical experiments to examine the effects of the strategy of group-wise update.

## 2. IDRstab method

We here describe an outline of the original IDRstab method [5]. The $k$th residual $\boldsymbol{r}_k$ of an IDR-type method is generated in a subspace $\mathcal{G}_k$. The subspaces $\mathcal{G}_k$ are defined by $\mathcal{G}_0 \equiv \mathcal{K}_n(A, \boldsymbol{r}_0)$ and $\mathcal{G}_{k+1} \equiv (I - \omega_{k+1}A)(\mathcal{G}_k \cap \tilde{R}_0^{\perp})$ for $k = 0, 1, \ldots$, where $\mathcal{K}_n(A, \boldsymbol{r}_0)$ is the full Krylov subspace generated by $A$ and an initial residual $\boldsymbol{r}_0 \equiv \boldsymbol{b} - A\boldsymbol{x}_0$, $\tilde{R}_0^{\perp}$ is the orthogonal complement of the range of a fixed $n \times s$ matrix $\tilde{R}_0$, and $\omega_k$'s are nonzero scalars. The dimension of $\mathcal{G}_k$ shrinks with increasing $k$, by the IDR theorem [1,3,5].

The residual $\boldsymbol{r}_k \in \mathcal{G}_k$ of IDRstab is updated to the next residual $\boldsymbol{r}_{k+\ell} \in \mathcal{G}_{k+\ell}$, where the integer $k$ is a multiple of $\ell$. The process of updating $\boldsymbol{r}_k$ to $\boldsymbol{r}_{k+\ell}$ is referred to as one cycle of IDRstab. One cycle consists of $\ell + 1$ steps of two different types: $\ell$ *IDR steps* and one *polynomial step*.

### 2.1. The IDR step

Suppose that we have an approximation $\boldsymbol{x}_k$ and the corresponding residual $\boldsymbol{r}_k \in \mathcal{G}_k$ (referred to as the *primary* residual), plus the $n \times s$ matrices $U_k$ and $AU_k$ with columns also in $\mathcal{G}_k$. Before performing the polynomial step, the IDR step is repeated $\ell$ times by using the projections $\Pi_i^{(j)}$ for $i = 0, 1, \ldots, j$, $j = 1, 2, \ldots, \ell$ which are defined by $\Pi_i^{(j)} \equiv I - A^i U_k^{(j-1)} \sigma_j^{-1} \tilde{R}_0^* A^{j-i}$ with $\sigma_j \equiv \tilde{R}_0^* A^j U_k^{(j-1)}$. Here, $\tilde{R}_0^*$ denotes the conjugate transpose of $\tilde{R}_0$, and the superscript "$(j)$" denotes the $j$th repetition.

Suppose that an approximation $\boldsymbol{x}_k^{(j-1)}$ and the corresponding residual $\boldsymbol{r}_k^{(j-1)}$, plus the vectors $A^i\boldsymbol{r}_k^{(j-1)}$ for $i = 1, 2, \ldots, j-1$, and the $n \times s$ matrices $A^i U_k^{(j-1)}$ for $i = 0, 1, \ldots, j$, are generated at the $(j-1)$st ($j \le \ell$) repetition, where $\boldsymbol{x}_k^{(0)} \equiv \boldsymbol{x}_k$, $\boldsymbol{r}_k^{(0)} \equiv \boldsymbol{r}_k$ and $U_k^{(0)} \equiv U_k$. The $j$th repetition is performed as follows. The residual $\boldsymbol{r}_k^{(j)}$ (referred to as a *secondary* residual) is obtained by the vector update

$$\boldsymbol{r}_k^{(j)} \equiv \Pi_1^{(j)} \boldsymbol{r}_k^{(j-1)} = \boldsymbol{r}_k^{(j-1)} - AU_k^{(j-1)} \vec{\alpha}^{(j)} \tag{1}$$

with the computation of $\vec{\alpha}^{(j)} \equiv \sigma_j^{-1}(\tilde{R}_0^* A^{j-1} \boldsymbol{r}_k^{(j-1)})$, and the associated approximation $\boldsymbol{x}_k^{(j)}$ is expressed by $\boldsymbol{x}_k^{(j)} = \boldsymbol{x}_k^{(j-1)} + U_k^{(j-1)} \vec{\alpha}^{(j)}$. For $i = 1, 2, \ldots, j-1$, since $A^{i+1} U_k^{(j-1)}$ are available, the vectors $A^i \boldsymbol{r}_k^{(j)}$ are also obtained by vector updates

$$A^i \boldsymbol{r}_k^{(j)} \equiv \Pi_{i+1}^{(j)} A^i \boldsymbol{r}_k^{(j-1)} = A^i \boldsymbol{r}_k^{(j-1)} - A^{i+1} U_k^{(j-1)} \vec{\alpha}^{(j)}. \tag{2}$$

The vector $A^j \boldsymbol{r}_k^{(j)}$ is obtained by multiplying $A^{j-1}\boldsymbol{r}_k^{(j)}$ by $A$. The matrices $A^i U_k^{(j-1)}$ are updated to $A^i U_k^{(j)}$ for $i = 0, 1, \ldots, j$ such that the columns of $A^j U_k^{(j)}$ form a set of bases for the subspace $\mathcal{K}_s(\Pi_j^{(j)} A, \Pi_j^{(j)} A^j \boldsymbol{r}_k^{(j)})$. The first columns $A^i U_k^{(j)} \boldsymbol{e}_1$ for $i = 0, 1, \ldots, j$ are obtained by vector updates

$$A^i U_k^{(j)} \boldsymbol{e}_1 \equiv \Pi_i^{(j)} A^i \boldsymbol{r}_k^{(j)} = A^i \boldsymbol{r}_k^{(j)} - A^i U_k^{(j-1)} \vec{\beta}_1^{(j)} \tag{3}$$

with the computation of $\vec{\beta}_1^{(j)} \equiv \sigma_j^{-1} \vec{\rho}_1^{(j)}$ and $\vec{\rho}_1^{(j)} \equiv \tilde{R}_0^* A^j \boldsymbol{r}_k^{(j)}$. For some $q < s$, the vector $\boldsymbol{c}_q^{(j)} \equiv A(A^j U_k^{(j)} \boldsymbol{e}_q)$ is computed as the $q$th column of $A^{j+1} U_k^{(j)}$ by an explicit multiplication by $A$; after that, the $(q+1)$st columns $A^i U_k^{(j)} \boldsymbol{e}_{q+1}$ for $i = 0, 1, \ldots, j$ can

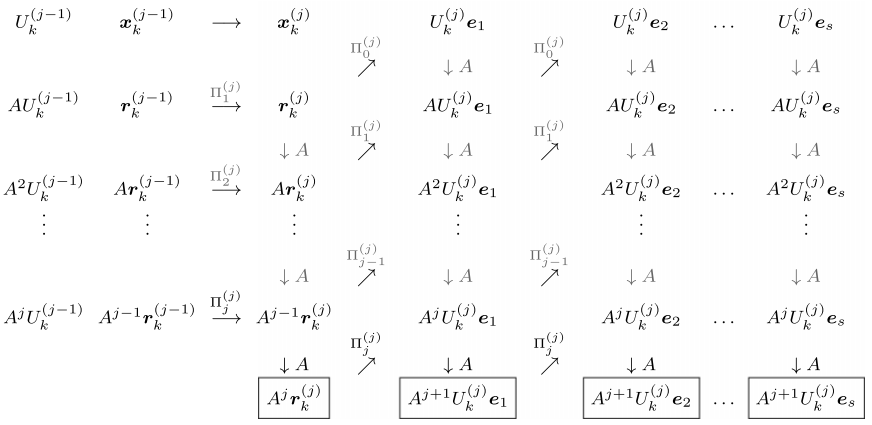

**Fig. 1.** Original scheme at the $j$th repetition.

be computed as

$$A^i U_k^{(j)} \boldsymbol{e}_{q+1} = \Pi_i^{(j)} A^{i+1} U_k^{(j)} \boldsymbol{e}_q = A^{i+1} U_k^{(j)} \boldsymbol{e}_q - A^i U_k^{(j-1)} \vec{\beta}_{q+1}^{(j)} \tag{4}$$

with the computation of $\vec{\beta}_{q+1}^{(j)} \equiv \sigma_j^{-1} \vec{\rho}_{q+1}^{(j)}$ and $\vec{\rho}_{q+1}^{(j)} \equiv \tilde{R}_0^* \boldsymbol{c}_q^{(j)}$.

At the $j$th repetition, the vectors $A^i \boldsymbol{r}_k^{(j)}$ for $i = 0, 1, \ldots, j-1$ and the columns of $A^i U_k^{(j)}$ for $i = 1, 2, \ldots, j$ belong to $\mathcal{G}_k \cap \tilde{R}_0^{\perp}$. Fig. 1 displays the scheme at the $j$th repetition. Note that the explicit multiplications by $A$ are used to obtain the boxed vectors, while one projection "$\Pi_j^{(j)}$" requires $s$ vector updates with the solution of a linear system of order $s$ and "$\Pi_i^{(j)}$", $0 \le i < j$, requires only $s$ vector updates.

### 2.2. The polynomial step

After the IDR step is repeated $\ell$ times, we have an approximation $\boldsymbol{x}_k^{(\ell)}$ and the corresponding residual $\boldsymbol{r}_k^{(\ell)}$, plus the vectors $A^i \boldsymbol{r}_k^{(\ell)}$ for $i = 1, 2, \ldots, \ell$, and $\ell + 2$ matrices $A^i U_k^{(\ell)}$ for $i = 0, 1, \ldots, \ell + 1$.

By multiplying $\boldsymbol{r}_k^{(\ell)}$ and $AU_k^{(\ell)}$ by a stabilizing polynomial of degree $\ell$, we obtain

$$\boldsymbol{r}_{k+\ell} = \boldsymbol{r}_k^{(\ell)} - \gamma_{1,k} A \boldsymbol{r}_k^{(\ell)} - \cdots - \gamma_{\ell,k} A^{\ell} \boldsymbol{r}_k^{(\ell)} \tag{5}$$

and

$$AU_{k+\ell} = AU_k^{(\ell)} - \gamma_{1,k} A^2 U_k^{(\ell)} - \cdots - \gamma_{\ell,k} A^{\ell+1} U_k^{(\ell)}, \tag{6}$$

respectively, where the scalars $\gamma_{1,k}, \gamma_{2,k}, \ldots, \gamma_{\ell,k}$ are determined by minimizing $\|\boldsymbol{r}_{k+\ell}\|_2$. We do not discuss refined variants as sketched in [11, pp. 208–209]. The next primary residual $\boldsymbol{r}_{k+\ell}$ and the columns of $AU_{k+\ell}$ belong to $\mathcal{G}_{k+\ell}$ [5]. The approximation $\boldsymbol{x}_k^{(\ell)}$ and the matrix $U_k^{(\ell)}$ are also updated to the next associated approximation $\boldsymbol{x}_{k+\ell}$ and the matrix $U_{k+\ell}$, respectively.

## 3. Alternative implementation of IDRstab

In this section, we propose an alternative recursion formula for updating the residuals to narrow the residual gap. By using the formula, we also derive an implementation of IDRstab which reduces the number of vector updates.

### 3.1. Narrowing the residual gap

The reasons for a large residual gap are closely related to the recursion formulas for updating the approximations and the corresponding residuals [8,9,14]. One of the reasons is as follows: in the formula (1), the columns of $AU_k^{(j-1)}$ are generated by vector updates instead of multiplying the columns of $U_k^{(j-1)}$ by $A$. In finite precision arithmetic, the computed vector $AU_k^{(j-1)} \vec{\alpha}^{(j)}$ may differ from $A(U_k^{(j-1)} \vec{\alpha}^{(j)})$, which is obtained by multiplication by $A$.

To narrow the residual gap, we apply the basic two-term recursion formulas which are often used in Lanczos-type methods: $\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{p}_k$, $\mathbf{r}_k = \mathbf{r}_{k-1} - A\mathbf{p}_k$, where $A\mathbf{p}_k$ is obtained by explicitly multiplying $\mathbf{p}_k$ by $A$. Specifically, we compute $A(U_k^{(j-1)} \vec{\alpha}^{(j)})$ by multiplying $U_k^{(j-1)} \vec{\alpha}^{(j)}$ by $A$ at the IDR step, and update the secondary residuals by

$$\boldsymbol{r}_k^{(j)} = \boldsymbol{r}_k^{(j-1)} - A(U_k^{(j-1)} \vec{\alpha}^{(j)}) \tag{7}$$

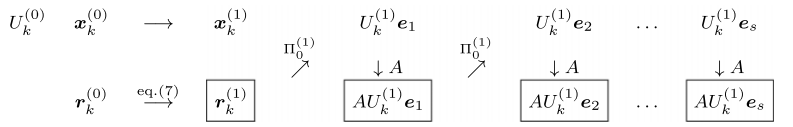

**Fig. 2.** Scheme of our variant at the first repetition.

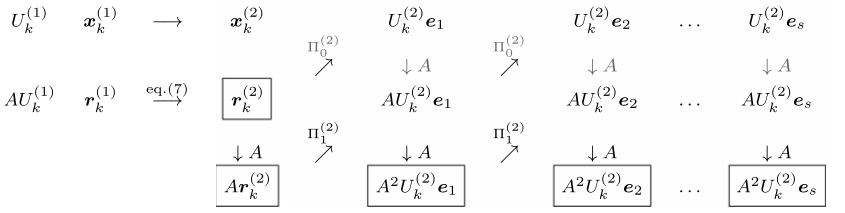

**Fig. 3.** Scheme of our variant at the second repetition.

for $j = 1, 2, \ldots, \ell$. Similarly, at the polynomial step, we update the primary residual by

$$\boldsymbol{r}_{k+\ell} = \boldsymbol{r}_k^{(\ell)} - A(\gamma_{1,k}\boldsymbol{r}_k^{(\ell)} + \gamma_{2,k}A\boldsymbol{r}_k^{(\ell)} + \cdots + \gamma_{\ell,k}A^{\ell-1}\boldsymbol{r}_k^{(\ell)}) \tag{8}$$

instead of (5), where the vector $A(\sum_{i=1}^{\ell}\gamma_{i,k}A^{i-1}\boldsymbol{r}_k^{(\ell)})$ is obtained by multiplying $\sum_{i=1}^{\ell}\gamma_{i,k}A^{i-1}\boldsymbol{r}_k^{(\ell)}$ by $A$. We show by numerical experiments in Section 4 that the formulas (7) and (8) are useful to narrow the residual gap.

### 3.2. Reducing the number of vector updates

To compensate for the extra costs induced by the additional MVs, we present a scheme that uses fewer vector updates than the original scheme.

As described in [12], we compute $\sigma_j$, $\vec{\alpha}^{(j)}$, $\vec{\rho}_1^{(j)}$, and $\vec{\rho}_{q+1}^{(j)}$ $(q < s)$ by the alternative forms $(A^*\tilde{R}_0)^*A^{j-1}U_k^{(j-1)}$, $\sigma_j^{-1}((A^*\tilde{R}_0)^*A^{j-2}\boldsymbol{r}_k^{(j-1)})$, $(A^*\tilde{R}_0)^*A^{j-1}\boldsymbol{r}_k^{(j)}$, and $(A^*\tilde{R}_0)^*A^jU_k^{(j)}\boldsymbol{e}_q$, respectively. Here the matrix $A^*\tilde{R}_0$ is computed once and stored at the initialization. These forms enable us to perform the $j$th repetition without using the matrix $A^jU_k^{(j-1)}$.

At the first repetition, i.e., for $j = 1$, after $\boldsymbol{x}_k^{(1)}$ and $\boldsymbol{r}_k^{(1)}$ are obtained, we compute the columns of $U_k^{(1)}$ by the projection $\Pi_0^{(1)}$, and then multiplying the columns of $U_k^{(1)}$ by $A$ gives $AU_k^{(1)}$ (see Fig. 2). Note that we need $AU_k^{(0)} \equiv AU_k$ to update $\boldsymbol{r}_k^{(0)}$ to $\boldsymbol{r}_k^{(1)}$ by the original formula (1), but this is not required in the alternative formula (7). At the $j$th repetition for $j = 2, 3, \ldots, \ell$, after $\boldsymbol{x}_k^{(j)}$ and $\boldsymbol{r}_k^{(j)}$ are computed, we obtain $A^i\boldsymbol{r}_k^{(j)}$ for $i = 1, 2, \ldots, j-2$, based on (2), and obtain $A^{j-1}\boldsymbol{r}_k^{(j)}$ by multiplying $A^{j-2}\boldsymbol{r}_k^{(j)}$ by $A$. We utilize the formulas (3) and (4) for $i = 0, 1, \ldots, j-1$ (see Figs. 3 and 4). If $j = \ell$, we obtain $A^\ell\boldsymbol{r}_k^{(\ell)}$, which we need to determine the scalars $\gamma_{i,k}$ at the polynomial step, by multiplying $A^{\ell-1}\boldsymbol{r}_k^{(\ell)}$ by $A$ (see Fig. 4).

As a result, $A^{j+1}U_k^{(j)}$ does not need to be computed at the $j$th repetition. Since $AU_{k+\ell}$ is not required, the update (6) is saved at the polynomial step. We will compare the computational costs of our proposed variant with those of the original IDRstab in the next subsection.

Figs. 2–4 display the schemes at the first, second, and $j$th repetition described above, respectively. Note that the residual $\boldsymbol{r}_k^{(j)}$ is obtained by vector update (1) in the original IDRstab, while the residual is updated by (7) in our variant.

The resulting algorithm is displayed in Algorithm 1. The formulas (7) and (8) are implemented as lines 12 and 27, respectively. As used in [5], we orthonormalize the columns of $U_0$ and $A^jU_k^{(j)}$ by the Arnoldi process for numerical stability in lines 5–6 and 20–21, respectively. The notation in the algorithm 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 to $\mathbf{U} = [\mathbf{U}_0; \mathbf{U}_1; \cdots; \mathbf{U}_j]$, $\mathbf{V} = [\mathbf{V}_0; \mathbf{V}_1; \cdots; \mathbf{V}_j]$, $\mathbf{r} = [\mathbf{r}_0; \mathbf{r}_1; \cdots; \mathbf{r}_j]$, and $\mathbf{u} = [\mathbf{u}_0; \mathbf{u}_1; \cdots; \mathbf{u}_j]$, respectively. Note that, at the $j$th repetition, the matrices $A^iU_k^{(j-1)}$ and $A^iU_k^{(j)}$ correspond to $\mathbf{U}_i$ and $\mathbf{V}_i$, respectively.

### 3.3. Computational costs

Table 1 summarizes the computational costs for MVs and vector updates of the form $a\boldsymbol{x} + \boldsymbol{y}$ with a scalar $a$ and the $n$-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ (AXPYs) of the original IDRstab and our variant per cycle. Here, the computations of the forms $a\boldsymbol{x}$ and $\boldsymbol{x} + \boldsymbol{y}$ are counted as $\frac{1}{2}$ AXPY, and we do not include the costs for the Arnoldi process.

**Algorithm 1.** Our proposed variant of IDRstab.

1. Select an initial guess $\mathbf{x}$ and an $(n \times s)$ matrix $\tilde{\mathbf{R}}_0$.
2. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x},\ \mathbf{r} = [\mathbf{r}_0]$
*% Generate an initial* $(n \times s)$ *matrix* $\mathbf{U} = [\mathbf{U}_0]$
3. For $q = 1, 2, \ldots, s$
4. if $q = 1,\ \mathbf{u}_0 = \mathbf{r}_0,$ else, $\mathbf{u}_0 = \mathbf{A}\mathbf{u}_0$
5. $\vec{\mu} = (\mathbf{U}_{0(:,1:q-1)})^*\mathbf{u}_0,\ \mathbf{u}_0 = \mathbf{u}_0 - \mathbf{U}_{0(:,1:q-1)}\vec{\mu}$
6. $\mathbf{u}_0 = \mathbf{u}_0/\|\mathbf{u}_0\|_2,\ \mathbf{U}_{0(:,q)} = \mathbf{u}_0$
7. End for
8. While $\|\mathbf{r}_0\|_2 > tol$
9. For $j = 1, 2, \ldots, \ell$
*% The IDR step*
10. $\sigma = (\mathbf{A}^*\tilde{\mathbf{R}}_0)^*\mathbf{U}_{j-1}$
11. if $j = 1,\ \vec{\alpha} = \sigma^{-1}(\tilde{\mathbf{R}}_0^*\mathbf{r}_0),$ else, $\vec{\alpha} = \sigma^{-1}((\mathbf{A}^*\tilde{\mathbf{R}}_0)^*\mathbf{r}_{j-2})$
12. $\mathbf{x} = \mathbf{x} + \mathbf{U}_0\vec{\alpha},\ \mathbf{r}_0 = \mathbf{r}_0 - \mathbf{A}(\mathbf{U}_0\vec{\alpha})$
13. For $i = 1, 2, \ldots, j-2$
14. $\mathbf{r}_i = \mathbf{r}_i - \mathbf{U}_{i+1}\vec{\alpha}$
15. End for
16. if $j > 1,\ \mathbf{r} = [\mathbf{r}; \mathbf{A}\mathbf{r}_{j-2}]$
17. For $q = 1, 2, \ldots, s$
18. if $q = 1,\ \mathbf{u} = \mathbf{r},$ else, $\mathbf{u} = [\mathbf{u}_1; \mathbf{u}_2; \ldots; \mathbf{u}_j]$
19. $\vec{\beta} = \sigma^{-1}((\mathbf{A}^*\tilde{\mathbf{R}}_0)^*\mathbf{u}_{j-1}),\ \mathbf{u} = \mathbf{u} - \mathbf{U}\vec{\beta},\ \mathbf{u} = [\mathbf{u}; \mathbf{A}\mathbf{u}_{j-1}]$
20. $\vec{\mu} = (\mathbf{V}_{j(:,1:q-1)})^*\mathbf{u}_j,\ \mathbf{u} = \mathbf{u} - \mathbf{V}_{(:,1:q-1)}\vec{\mu}$
21. $\mathbf{u} = \mathbf{u}/\|\mathbf{u}_j\|_2,\ \mathbf{V}_{(:,q)} = \mathbf{u}$
22. End for
23. $\mathbf{U} = \mathbf{V}$
24. End for
25. $\mathbf{r} = [\mathbf{r}; \mathbf{A}\mathbf{r}_{\ell-1}]$
*% The polynomial step*
26. $\vec{\gamma} = [\gamma_1; \gamma_2; \ldots; \gamma_\ell] = \arg\min_{\vec{\gamma}} \|\mathbf{r}_0 - [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_\ell]\vec{\gamma}\|_2$
27. $\mathbf{x} = \mathbf{x} + [\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{\ell-1}]\vec{\gamma},\ \mathbf{r}_0 = \mathbf{r}_0 - \mathbf{A}([\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{\ell-1}]\vec{\gamma})$
28. $\mathbf{U} = [\mathbf{U}_0 - \sum_{j=1}^{\ell}\gamma_j\mathbf{U}_j]$
29. End while

**Table 1**
Computational costs for MVs and AXPYs of the original IDRstab and our variant per cycle.

| Solver | MVs | AXPYs |
|---|---|---|
| IDRstab | $\ell(s+1)$ | $\frac{1}{2}\ell s(\ell+1)(s+1)+\ell(s^2+3s+2)$ |
| Our variant | $\ell(s+1)+\ell+1$ | $\frac{1}{2}\ell s(\ell+1)(s+1)+\frac{3}{2}\ell+s+\frac{1}{2}$ |

Our variant requires $\ell+1$ additional MVs per cycle, but instead, $\ell(s^2+3s+\frac{1}{2})-s-\frac{1}{2}$ AXPYs are reduced. For example, in the case of $(s, \ell) = (4, 4)$, our variant requires 25 MVs and 210.5 AXPYs per cycle, while the original IDRstab requires 20 MVs and 320 AXPYs. For sparse linear systems, we expect that the extra costs induced by the additional MVs in our variant can be compensated for by reducing the number of AXPYs.

## 4. Reliability of the recursion formulas

In this section, we discuss the reliability of the alternative recursion formulas for updating the residuals. We present numerical experiments (a) and (b) to show that the explicit multiplications by $A$ in (7) and (8) play an important role in avoiding a large residual gap.

We introduce the auxiliary vectors

$$\boldsymbol{p}_{k+\ell}^{(j)} \equiv \begin{cases} \displaystyle\sum_{i=1}^{\ell}\gamma_{i,k}A^{i-1}\boldsymbol{r}_k^{(\ell)} & \text{if } j = 0, \\ U_{k+\ell}^{(j-1)}\vec{\alpha}^{(j)} & \text{if } j = 1, 2, \ldots, \ell, \end{cases}$$

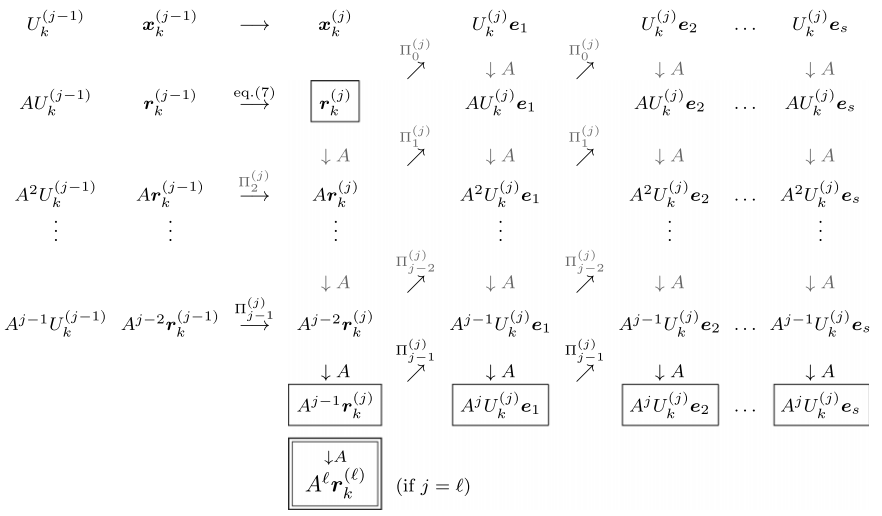

**Fig. 4.** Scheme of our variant at the $j$th repetition.

$$\boldsymbol{q}_{k+\ell}^{(j)} \equiv \begin{cases} \displaystyle\sum_{i=1}^{\ell} \gamma_{i,k} A^i \boldsymbol{r}_k^{(\ell)} & \text{if } j = 0, \\ A U_{k+\ell}^{(j-1)} \vec{\alpha}^{(j)} & \text{if } j = 1, 2, \ldots, \ell, \end{cases}$$

with $\boldsymbol{p}_0^{(j)} \equiv U_0^{(j-1)} \vec{\alpha}^{(j)}$ and $\boldsymbol{q}_0^{(j)} \equiv A U_0^{(j-1)} \vec{\alpha}^{(j)}$ for $j = 1, 2, \ldots, \ell$. The original recursion formulas (1) and (5) can be written by

$$\boldsymbol{r}_k^{(j)} = \boldsymbol{r}_k^{(j-1)} - \boldsymbol{q}_k^{(j)} \quad (j = 1, 2, \ldots, \ell) \quad \text{and} \quad \boldsymbol{r}_{k+\ell}^{(0)} = \boldsymbol{r}_k^{(\ell)} - \boldsymbol{q}_{k+\ell}^{(0)}, \tag{9}$$

respectively. The alternative recursion formulas (7) and (8) can be expressed by

$$\boldsymbol{r}_k^{(j)} = \boldsymbol{r}_k^{(j-1)} - A\boldsymbol{p}_k^{(j)} \quad (j = 1, 2, \ldots, \ell) \quad \text{and} \quad \boldsymbol{r}_{k+\ell}^{(0)} = \boldsymbol{r}_k^{(\ell)} - A\boldsymbol{p}_{k+\ell}^{(0)}, \tag{10}$$

respectively.

Numerical calculations were carried out in double-precision floating-point arithmetic with a GNU C++ 4.5.2 compiler. We set $A = \mathrm{diag}(a_1, a_2, \ldots, a_{1000})$ with $a_i \equiv \sqrt{1 + 9.999(i-1)}$ for $i = 1, 2, \ldots, 1000$. The iterations were stopped when the relative residual norms (i.e., $\|\boldsymbol{r}_k\|_2/\|\boldsymbol{b}\|_2$) become $10^{-15}$. The parameters $(s, \ell)$ were set at (4, 4), (6, 2) and (2, 6). In the figures of Sections 4.1 and 4.2, the plots show the number of MVs on the horizontal axis versus the $\log_{10}$ of the quantity (11) and (12), or (13) on the vertical axis, respectively. We refer to Section 5.1 for $\boldsymbol{b}$, $\boldsymbol{x}_0$, and $\tilde{R}_0$.

### 4.1. Numerical experiment (a)

We first examine whether the rounding errors, which arise from the vector updates and inner products, accumulate in $\boldsymbol{p}_k^{(j)}$ and $\boldsymbol{q}_k^{(j)}$ of the original IDRstab and in $\boldsymbol{p}_k^{(j)}$ and $A\boldsymbol{p}_k^{(j)}$ of our variant. Note that the rounding errors which arise from multiplying a vector by the diagonal matrix are sufficiently small to be ignored. To investigate the accumulations of rounding errors, we compute the quantities

$$\frac{\|\boldsymbol{p}_k^{(j)} - \overline{\boldsymbol{p}}_k^{(j)}\|_2}{\|\overline{\boldsymbol{p}}_k^{(j)}\|_2}, \qquad \frac{\|\boldsymbol{q}_k^{(j)} - \overline{\boldsymbol{q}}_k^{(j)}\|_2}{\|\overline{\boldsymbol{q}}_k^{(j)}\|_2} \tag{11}$$

for the original IDRstab, and

$$\frac{\|\boldsymbol{p}_k^{(j)} - \overline{\boldsymbol{p}}_k^{(j)}\|_2}{\|\overline{\boldsymbol{p}}_k^{(j)}\|_2}, \qquad \frac{\|A\boldsymbol{p}_k^{(j)} - \overline{A\boldsymbol{p}}_k^{(j)}\|_2}{\|\overline{A\boldsymbol{p}}_k^{(j)}\|_2} \tag{12}$$

for our variant. Here the vectors $\overline{\boldsymbol{p}}_k^{(j)}$, $\overline{\boldsymbol{q}}_k^{(j)}$, and $\overline{A\boldsymbol{p}}_k^{(j)}$ are computed in quad-double precision floating-point arithmetic by QD 2.3.12 [13]. We note that our variant is mathematically equivalent to the original IDRstab, and have checked by numerical computations that 16 digits of $\overline{\boldsymbol{p}}_k^{(j)}$ and $\overline{\boldsymbol{q}}_k^{(j)}$ obtained by the original IDRstab coincide with those of $\overline{\boldsymbol{p}}_k^{(j)}$ and $\overline{A\boldsymbol{p}}_k^{(j)}$, respectively, obtained by our variant.

*Numerical results*: Fig. 5 displays the histories of the quantities in (11) for the original IDRstab and those in (12) for our variant. We can see that the quantity (11) calculated by the original IDRstab and the quantity (12) calculated by our variant become $10^{-6}$ for $(s, \ell) = (4, 4)$ and (6, 2). In the case of $(s, \ell) = (2, 6)$, the quantities (11) and (12) both become $10^{-2}$.

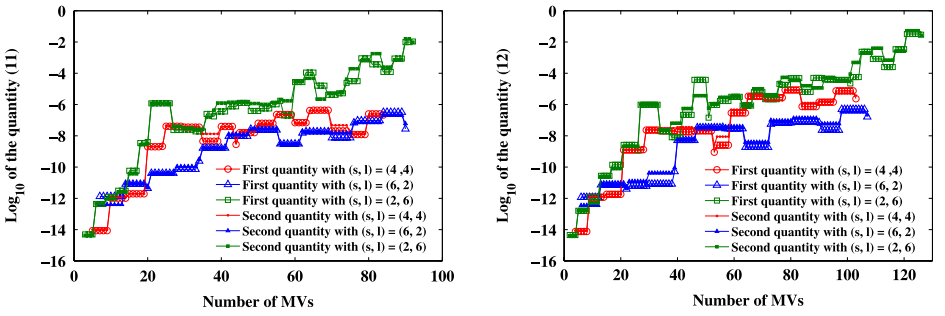


**Fig. 5.** Histories of the first and second quantities in (11) for the original IDRstab (on the left) and those in (12) for our variant (on the right).

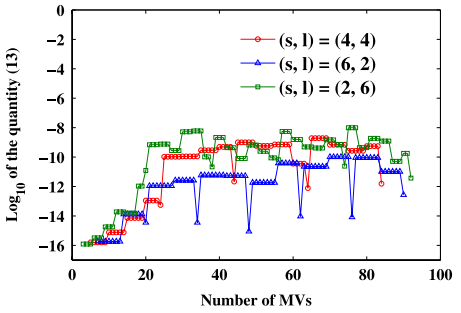


**Fig. 6.** Histories of the quantity (13) for the original IDRstab.

**Table 2**
Recursion formulas used in the original IDRstab and the variants.

| Solver | The IDR step | The polynomial step | Saving AXPYs |
|---|---|---|---|
| IDRstab | (9) | (9) | No |
| Variant 1 | If $j = 1$, (10), else, (9) | (9) | Yes |
| Variant 2 | (10) | (9) | Yes |
| Our variant | (10) | (10) | Yes |

### 4.2. Numerical experiment (b)

From experiment (a), we can observe that $\boldsymbol{p}_k^{(j)}$ and $A\boldsymbol{p}_k^{(j)}$ of our variant are as affected as $\boldsymbol{p}_k^{(j)}$ and $\boldsymbol{q}_k^{(j)}$, respectively, of the original IDRstab, by rounding errors which arise from the vector updates and inner products. Our variant essentially differs from the original IDRstab in the procedure that $A\boldsymbol{p}_k^{(j)}$ is computed by multiplying $\boldsymbol{p}_k^{(j)}$ by $A$, while $\boldsymbol{q}_k^{(j)}$ is computed by using vector updates. Here $\boldsymbol{q}_k^{(j)}$ is equivalent to $A\boldsymbol{p}_k^{(j)}$ in exact arithmetic but may differ from $A\boldsymbol{p}_k^{(j)}$ in finite precision arithmetic. For this reason, the original IDRstab may have a large residual gap. We therefore compute the quantity

$$\frac{\|\boldsymbol{q}_k^{(j)} - A\boldsymbol{p}_k^{(j)}\|_2}{\|A\boldsymbol{p}_k^{(j)}\|_2}, \tag{13}$$

and examine it for the original IDRstab, our variant, and the following two variants, which are the intermediate implementations between the original IDRstab and our variant.

*Variant* 1: We use the alternative formula (10) only for $j = 1$. At the $j$th repetition for $j = 2, 3, \ldots, \ell$ and the polynomial step, we use the original formula (9). Using (10) for $j = 1$ enables us to perform the IDR step in which the number of vector updates is reduced, as described in Section 3.2 (see also [12]). Note that, at the first repetition, the matrix $AU_k^{(1)}$ is obtained by multiplying the columns of $U_k^{(1)}$ by $A$ instead of using the projections (see Fig. 2).

*Variant* 2: We use the alternative formula (10) at the IDR step, i.e., for $j = 1, 2, \ldots, \ell$, and use the original formula (9) only at the polynomial step. As in the case of variant 1, $AU_k^{(1)}$ is obtained by multiplying the columns of $U_k^{(1)}$ by $A$ at the first repetition, and the number of vector updates can be reduced at each repetition (see Figs. 2–4).

Table 2 shows the recursion formulas used in the original IDRstab, variant 1, variant 2, and our variant.

*Numerical results*: Figs. 6 and 7 display the histories of the quantity (13) for the original IDRstab and those for variants 1 and 2, respectively. The rounding errors which arise from the matrix–vector multiplication with the diagonal matrix $A$ are

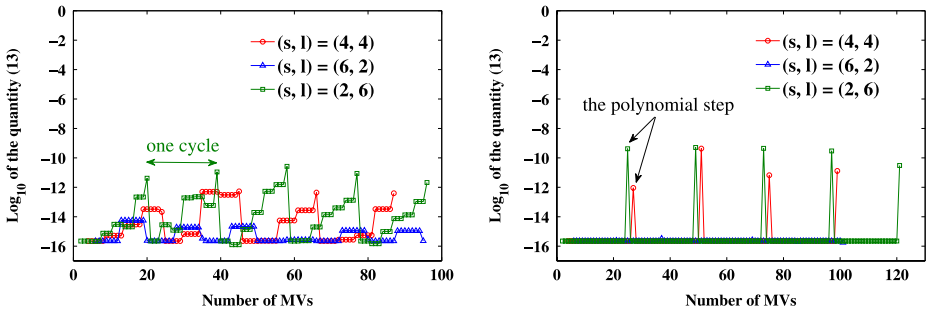


**Fig. 7.** Histories of the quantity (13) for variant 1 (on the left) and variant 2 (on the right).

**Table 3**
True relative residual norms of the original IDRstab and the variants.

| Solver | $(s, \ell)$ | | |
|---|---|---|---|
| | (4, 4) | (6, 2) | (2, 6) |
| IDRstab | 4.62E−14 | 2.90E−15 | 3.11E−12 |
| Variant 1 | 1.24E−15 | 3.60E−16 | 1.89E−14 |
| Variant 2 | 1.89E−14 | 2.53E−16 | 1.90E−12 |
| Our variant | 9.61E−16 | 2.18E−16 | 3.13E−16 |

sufficiently small. We have confirmed that 16 digits of a vector which is obtained by multiplying $\boldsymbol{p}_k^{(j)}$ by $A$ in double precision coincide with those of a vector obtained in quad-double precision. We therefore plot machine epsilon 2.22E−16 instead of (13) when using the formula (10). Table 3 shows the true relative residual 2-norms at termination.

From Figs. 6 and 7, we can observe the following. The vector $\boldsymbol{q}_k^{(j)}$ differs from $A\boldsymbol{p}_k^{(j)}$ in the original IDRstab. Since $AU_k^{(1)}$ is obtained by multiplying the columns of $U_k^{(1)}$ by $A$, the quantity (13) for variant 1 can be small at the first repetition, but increases as the IDR step repeats. The quantity for variant 2 is comparable to that of the original IDRstab for $(s, \ell) = (4, 4)$ and (2, 6). Note that the quantity for our variant can be treated as zero for all steps.

From Figs. 6 and 7 and Table 3, we can see that the residual gap becomes small as the quantity (13) decreases. We therefore conclude that, even if both $\boldsymbol{p}_k^{(j)}$ and $\boldsymbol{q}_k^{(j)}$ are affected by rounding errors (see experiment (a)), $\boldsymbol{q}_k^{(j)}$ has to coincide with $A\boldsymbol{p}_k^{(j)}$ to reduce a residual gap, i.e., the residual gap can be reduced by explicitly multiplying $\boldsymbol{p}_k^{(j)}$ by $A$.

We note that multiplying $U_k^{(j-1)}\vec{\alpha}^{(j)}$ and $\sum_{i=1}^{\ell}\gamma_{i,k}A^{i-1}\boldsymbol{r}_k^{(\ell)}$ by $A$, and computing $AU_k$ by multiplying the columns of $U_k$ by $A$ at each cycle are useful to narrow the residual gap, but it is not efficient to implement such a procedure in the original IDRstab because the computational costs are high.

## 5. Numerical experiments

In this section, we present some numerical experiments on model problems with sparse nonsymmetric matrices. The examples show that the original IDRstab has a large residual gap. We compare the convergence of our variant with that of the original IDRstab. We also compare the convergence between the original IDRstab and our variant with preconditioning.

Numerical calculations were carried out in double-precision floating-point arithmetic on a PC (Intel Core i7 2.67 GHz CPU) with an Intel C++ 11.1.048 compiler. The iterations were started with $\mathbf{0}$. The stopping criterion was set at $\|\boldsymbol{r}_k\|_2 < 10^{-12}\|\boldsymbol{b}\|_2$. The columns of $\tilde{R}_0$ were given by the orthonormalization of $s$ real random vectors in the interval (0, 1). The parameters $(s, \ell)$ were set at (2, 2), (2, 4), (2, 6), (4, 2), (4, 4), (4, 6), (6, 2), (6, 4), and (6, 6).

In Sections 5.1–5.3, the numbers of MVs and cycles are plotted on the horizontal axis in the left and right panels of Figs. 8, 10 and 12, respectively. The computation time is plotted on the horizontal axis of Figs. 9, 11 and 13. The $\log_{10}$ of the relative residual 2-norm ($\|\boldsymbol{r}_k^{(j)}\|_2/\|\boldsymbol{b}\|_2$ and $\|\boldsymbol{b} - A\boldsymbol{x}_k^{(j)}\|_2/\|\boldsymbol{b}\|_2$) is plotted on the vertical axis.

### 5.1. Example 1

As in [5], we first consider the test matrix SHERMAN5 from the Matrix-Market collection. The dimension $n$ and number of nonzero entries (abbreviated as $nnz$) of the matrix are 3312 and 20,793, respectively. The percentage of $nnz$ is 0.19 and the average $nnz$ per row is 6.28. The condition number is 1.88E+05. The right-hand side vector $\boldsymbol{b}$ is given by substituting $\hat{\boldsymbol{x}} \equiv (1, 1, \ldots, 1)^\top$ into the equation $\boldsymbol{b} = A\hat{\boldsymbol{x}}$.

Figs. 8 and 9 display the convergence histories of the original IDRstab and our variant for $(s, \ell) = (4, 2)$. Table 4 shows the numbers of cycles and MVs, the computation times, and the true relative residual 2-norms at termination, which are abbreviated as "Cycles", "MVs", "Time [s]", and "True res.", respectively.

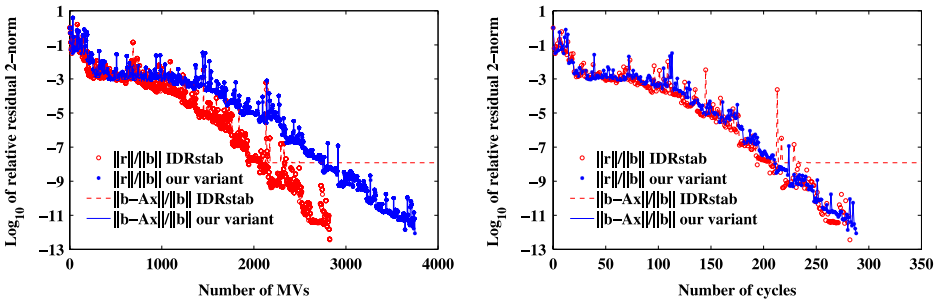


**Fig. 8.** Convergence histories plotted with the number of MVs (on the left) and cycles (on the right) of the original IDRstab and our variant with $(s, \ell) = (4, 2)$ for example 1.

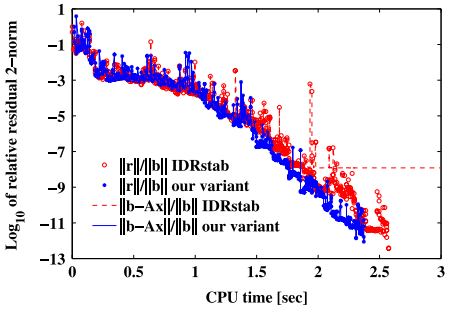


**Fig. 9.** Convergence histories plotted with the computation time of the original IDRstab and our variant with $(s, \ell) = (4, 2)$ for example 1.

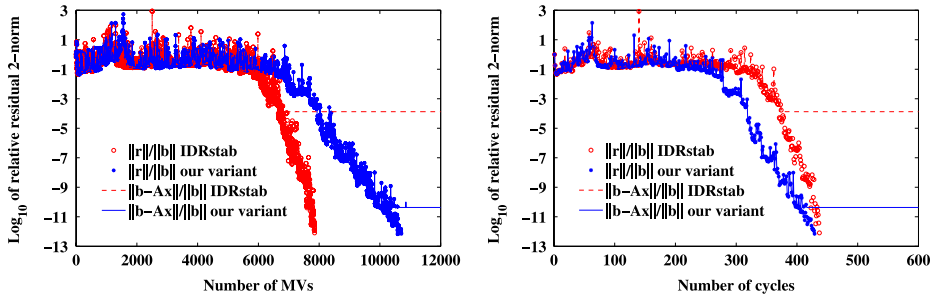


**Fig. 10.** Convergence histories plotted with the number of MVs (on the left) and cycles (on the right) of the original IDRstab and our variant with $(s, \ell) = (2, 6)$ for example 2.

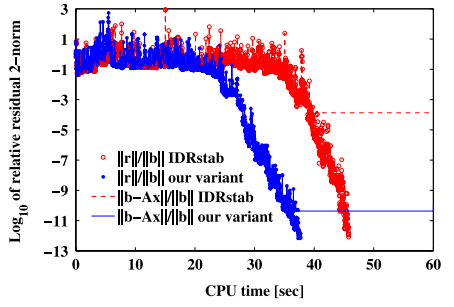


**Fig. 11.** Convergence histories plotted with the computation time of the original IDRstab and our variant with $(s, \ell) = (2, 6)$ for example 2.

From Figs. 8 and 9 and Table 4, we can observe the following. The original IDRstab has a large residual gap, i.e., the approximate solutions do not attain the required accuracy. Our variant converges with a small residual gap. The approximate solutions obtained by our variant are more accurate than those obtained by the original IDRstab. The numbers of cycles required for successful convergence of our variant are about the same as those of the original IDRstab. The computation times

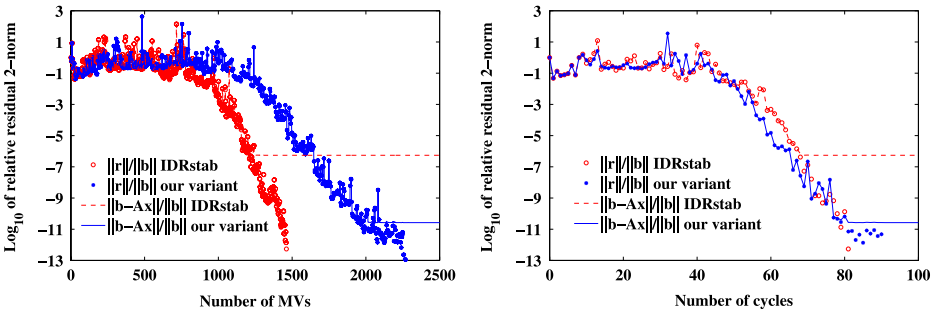


**Fig. 12.** Convergence histories plotted with the number of MVs (on the left) and cycles (on the right) of the original IDRstab and our variant with preconditioning for $(s, \ell) = (2, 6)$ in example 2 .

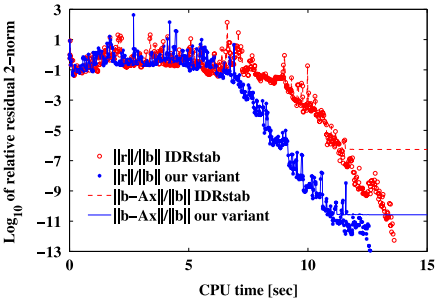


**Fig. 13.** Convergence histories plotted with the computation time of the original IDRstab and our variant with preconditioning for $(s, \ell) = (2, 6)$ in example 2.

for our variant are slightly shorter than those for the original IDRstab; therefore, the extra costs induced by the additional MVs in our variant are compensated for by reducing the number of AXPYs.

### 5.2. Example 2

Next, as shown in [15], we take up a system with a sparse nonsymmetric matrix derived from the finite difference discretization of the following partial differential equation on the unit square $\Omega = [0, 1] \times [0, 1]$:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + D\left[\left(y - \frac{1}{2}\right)\frac{\partial u}{\partial x} + \left(x - \frac{1}{3}\right)\left(x - \frac{2}{3}\right)\frac{\partial u}{\partial y}\right] - 43\pi^2 u = G(x, y),$$
$$u(x, y)|_{\partial\Omega} = 1 + xy.$$

The equation is discretized by a 5-point central difference approximation. The mesh size $h$ is chosen as $129^{-1}$ in both directions of $\Omega$. The dimension $n$ and $nnz$ of the matrix are $128^2$ and 81,408, respectively. The percentage of $nnz$ is 0.03 and the average $nnz$ per row is 4.97. The right-hand side vector $\boldsymbol{b}$ is given such that the exact solution $u(x, y)$ of the equation is $1 + xy$. The parameter $Dh$ is set at $2^{-1}$.

Figs. 10 and 11 display the convergence histories of the original IDRstab and our variant for $(s, \ell) = (2, 6)$. Table 5 shows the numbers of cycles and MVs, the computation times, and the true relative residual 2-norms at termination.

From Figs. 10 and 11 and Table 5, we can observe the following. Six digits are accurate for the approximate solutions obtained by the original IDRstab, while 11 digits are accurate for those obtained by our variant. The residual gap of the original IDRstab becomes larger as the parameter $\ell$ increases for a fixed $s$. The residual gap of our variant is much smaller than that of the original IDRstab. Note that a large residual gap, which the oscillation of the residual norms causes in our variant and the original IDRstab, cannot be avoided since we do not use the strategy described in [8,9]. Although the number of cycles required for successful convergence of our variant is sometimes more than that of the original IDRstab, the computation times for our variant are comparable to those for the original IDRstab in the cases of $(s, \ell) = (2, 2), (2, 4), (2, 6), (4, 4)$, and $(6, 2)$.

### 5.3. Experiments with preconditioning

We present the numerical experiments using preconditioning. Our proposed variant with preconditioning is displayed in Algorithm 2, which can be derived by applying Algorithm 1 to the right-preconditioned system $\tilde{A}\tilde{\boldsymbol{x}} = \boldsymbol{b}$, where $\tilde{A} = AK^{-1}$ and $\tilde{\boldsymbol{x}} = K\boldsymbol{x}$ for a preconditioner $K$.

**Table 4**
Numbers of cycles and MVs, computation times, and true relative residual norms of the original IDRstab and our variant for example 1.

| $(s, \ell)$ | Solver | Cycles | MVs | Time [s] | True res. |
|---|---|---|---|---|---|
| (2, 2) | IDRstab | 521 | 3129 | 2.40 | 1.26E−06 |
| | Our variant | 581 | 5231 | 2.57 | 6.93E−13 |
| (2, 4) | IDRstab | 225 | 2703 | 2.73 | 4.24E−09 |
| | Our variant | 237 | 4031 | 2.45 | 7.17E−13 |
| (2, 6) | IDRstab | 148 | 2667 | 3.07 | 4.22E−10 |
| | Our variant | 156 | 3902 | 2.90 | 8.21E−13 |
| (4, 2) | IDRstab | 282 | 2825 | 2.57 | 1.21E−08 |
| | Our variant | 288 | 3748 | 2.37 | 8.72E−13 |
| (4, 4) | IDRstab | 116 | 2325 | 2.59 | 8.25E−10 |
| | Our variant | 118 | 2954 | 2.39 | 4.43E−13 |
| (4, 6) | IDRstab | 78 | 2345 | 2.96 | 4.27E−09 |
| | Our variant | 82 | 3038 | 2.89 | 9.78E−13 |
| (6, 2) | IDRstab | 169 | 2373 | 2.84 | 1.87E−10 |
| | Our variant | 180 | 3066 | 2.64 | 1.35E−13 |
| (6, 4) | IDRstab | 77 | 2163 | 3.10 | 3.67E−09 |
| | Our variant | 84 | 2778 | 3.01 | 2.70E−13 |
| (6, 6) | IDRstab | 52 | 2191 | 3.68 | 8.23E−06 |
| | Our variant | 56 | 2750 | 3.56 | 1.79E−13 |

**Table 5**
Numbers of cycles and MVs, computation times, and true relative residual norms of the original IDRstab and our variant for example 2.

| $(s, \ell)$ | Solver | Cycles | MVs | Time [s] | True res. |
|---|---|---|---|---|---|
| (2, 2) | IDRstab | 1783 | 10 701 | 36.75 | 1.26E−06 |
| | Our variant | 1827 | 16 445 | 34.62 | 5.34E−11 |
| (2, 4) | IDRstab | 677 | 8 127 | 38.85 | 1.64E−06 |
| | Our variant | 764 | 12 990 | 38.42 | 4.47E−11 |
| (2, 6) | IDRstab | 436 | 7 851 | 45.76 | 1.33E−04 |
| | Our variant | 428 | 10 702 | 37.72 | 4.27E−11 |
| (4, 2) | IDRstab | 446 | 4 465 | 19.48 | 4.51E−07 |
| | Our variant | 597 | 7 765 | 23.01 | 1.32E−11 |
| (4, 4) | IDRstab | 226 | 4 525 | 27.57 | 3.12E−05 |
| | Our variant | 256 | 6 404 | 27.13 | 1.86E−11 |
| (4, 6) | IDRstab | 142 | 4 265 | 28.00 | 1.23E−04 |
| | Our variant | 259 | 9 587 | 46.42 | 1.15E−11 |
| (6, 2) | IDRstab | 315 | 4 417 | 25.82 | 8.01E−07 |
| | Our variant | 371 | 6 313 | 25.83 | 4.67E−12 |
| (6, 4) | IDRstab | 124 | 3 479 | 25.18 | 8.88E−07 |
| | Our variant | 188 | 6 210 | 33.38 | 6.43E−12 |
| (6, 6) | IDRstab | 76 | 3 199 | 28.03 | 1.03E−05 |
| | Our variant | 121 | 5 935 | 39.03 | 1.42E−11 |

By replacing $\tilde{\boldsymbol{x}}_k^{(j)}$, $\tilde{\boldsymbol{r}}_k^{(j)}$, and $\tilde{U}_k^{(j)}$ by $K\boldsymbol{x}_k^{(j)}$, $\boldsymbol{r}_k^{(j)}$, and $U_k^{(j)}$, respectively, the approximation $\boldsymbol{x}_k^{(j)}$ and the corresponding residual $\boldsymbol{r}_k^{(j)}$ are expressed by

$$\boldsymbol{x}_k^{(j)} = \boldsymbol{x}_k^{(j-1)} + K^{-1}U_k^{(j-1)}\vec{\alpha}^{(j)}, \qquad \boldsymbol{r}_k^{(j)} = \boldsymbol{r}_k^{(j-1)} - AK^{-1}U_k^{(j-1)}\vec{\alpha}^{(j)} \tag{14}$$

at the IDR step. $\boldsymbol{x}_{k+\ell}$ and $\boldsymbol{r}_{k+\ell}$ are expressed by

$$\boldsymbol{x}_{k+\ell} = \boldsymbol{x}_k^{(\ell)} + \gamma_{1,k}K^{-1}\boldsymbol{r}_k^{(\ell)} + \cdots + \gamma_{\ell,k}K^{-1}(AK^{-1})^{\ell-1}\boldsymbol{r}_k^{(\ell)}, \tag{15}$$

$$\boldsymbol{r}_{k+\ell} = \boldsymbol{r}_k^{(\ell)} - \gamma_{1,k}AK^{-1}\boldsymbol{r}_k^{(\ell)} - \cdots - \gamma_{\ell,k}(AK^{-1})^{\ell}\boldsymbol{r}_k^{(\ell)} \tag{16}$$

at the polynomial step.

By introducing $\hat{U}_k^{(j)} \equiv K^{-1}U_k^{(j)}$, (14) can be reformulated to

$$\boldsymbol{x}_k^{(j)} = \boldsymbol{x}_k^{(j-1)} + \hat{U}_k^{(j-1)}\vec{\alpha}^{(j)}, \qquad \boldsymbol{r}_k^{(j)} = \boldsymbol{r}_k^{(j-1)} - A(\hat{U}_k^{(j-1)}\vec{\alpha}^{(j)}),$$

**Algorithm 2.** Our proposed variant of IDRstab with right preconditioning.

1. Select an initial guess $\mathbf{x}$ and an $(n \times s)$ matrix $\tilde{\mathbf{R}}_0$.
2. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x},\ \mathbf{r} = [\mathbf{r}_0]$
*% Generate an initial $(n \times s)$ matrix $\hat{\mathbf{U}} = [\hat{\mathbf{U}}_0] = [\mathbf{K}^{-1}\mathbf{U}_0]$*
3. For $q = 1, \ldots, s$
4. if $q = 1,\ \hat{\mathbf{u}}_0 = \mathbf{K}^{-1}\mathbf{r}_0,$ else, $\hat{\mathbf{u}}_0 = \mathbf{K}^{-1}\mathbf{A}\hat{\mathbf{u}}_0$
*% Orthonormalize $\hat{\mathbf{u}}_0$ to the columns of $\hat{\mathbf{U}}_{0(:,1:q-1)}$*
5. $\hat{\mathbf{U}}_{0(:,q)} = \hat{\mathbf{u}}_0$
6. End for
7. While $\|\mathbf{r}_0\|_2 > tol$
*% The IDR step*
8. For $j = 1, \ldots, \ell$
9. $\sigma = (\mathbf{A}^*\tilde{\mathbf{R}}_0)^*\hat{\mathbf{U}}_{j-1}$
10. if $j = 1,\ \vec{\alpha} = \sigma^{-1}(\tilde{\mathbf{R}}_0^*\mathbf{r}_0),$ else, $\vec{\alpha} = \sigma^{-1}((\mathbf{A}^*\tilde{\mathbf{R}}_0)^*\hat{\mathbf{r}}_{j-2})$
11. $\mathbf{x} = \mathbf{x} + \hat{\mathbf{U}}_0\vec{\alpha},\ \mathbf{r}_0 = \mathbf{r}_0 - \mathbf{A}(\hat{\mathbf{U}}_0\vec{\alpha})$
12. For $i = 1, \ldots, j-2$
13. $\mathbf{r}_i = \mathbf{r}_i - \mathbf{U}_{i-1}\vec{\alpha}$
14. End for
15. if $j = 1,\ \hat{\mathbf{r}} = [\mathbf{K}^{-1}\mathbf{r}_0],$ else, $\hat{\mathbf{r}} = \hat{\mathbf{r}} - [\hat{\mathbf{U}}_1; \ldots; \hat{\mathbf{U}}_{j-1}]\vec{\alpha},\ \mathbf{r} = [\mathbf{r}; \mathbf{A}\hat{\mathbf{r}}_{j-2}],\ \hat{\mathbf{r}} = [\hat{\mathbf{r}}; \mathbf{K}^{-1}\mathbf{r}_{j-1}]$
16. For $q = 1, \ldots, s$
17. if $q = 1,\ \hat{\mathbf{u}} = \hat{\mathbf{r}},\ \mathbf{u} = [\mathbf{r}_2; \ldots; \mathbf{r}_{j-2}],$ else, $\hat{\mathbf{u}} = [\hat{\mathbf{u}}_1; \ldots; \hat{\mathbf{u}}_j],\ \mathbf{u} = [\mathbf{u}_1; \ldots; \mathbf{u}_{j-2}]$
18. $\vec{\beta} = \sigma^{-1}((\mathbf{A}^*\tilde{\mathbf{R}}_0)^*\hat{\mathbf{u}}_{j-1}),\ \hat{\mathbf{u}} = \hat{\mathbf{u}} - \hat{\mathbf{U}}\vec{\beta},$ if $j > 2,\ \mathbf{u} = \mathbf{u} - \mathbf{U}\vec{\beta}$
19. $\hat{\mathbf{u}} = [\hat{\mathbf{u}}; \mathbf{K}^{-1}\mathbf{A}\hat{\mathbf{u}}_{j-1}],$ if $j = 2,\ \mathbf{u} = [\mathbf{A}\hat{\mathbf{u}}_1],$ else if $j > 2,\ \mathbf{u} = [\mathbf{u}; \mathbf{A}\hat{\mathbf{u}}_{j-1}]$
*% Orthonormalize $\hat{\mathbf{u}}_j$ to the columns of $\hat{\mathbf{V}}_{j(:,1:q-1)}$*
20. $\hat{\mathbf{V}}_{(:,q)} = \hat{\mathbf{u}},$ if $j > 1,\ \mathbf{V}_{(:,q)} = \mathbf{u}$
21. End for
22. $\hat{\mathbf{U}} = \hat{\mathbf{V}},$ if $j > 1,\ \mathbf{U} = \mathbf{V}$
23. End for
24. $\mathbf{r} = [\mathbf{r}; \mathbf{A}\hat{\mathbf{r}}_{\ell-1}]$
*% The polynomial step*
25. $\vec{\gamma} = [\gamma_1; \ldots; \gamma_\ell] = \arg\min_{\vec{\gamma}} \|\mathbf{r}_0 - [\mathbf{r}_1, \ldots, \mathbf{r}_\ell]\vec{\gamma}\|_2$
26. $\mathbf{x} = \mathbf{x} + [\hat{\mathbf{r}}_0, \ldots, \hat{\mathbf{r}}_{\ell-1}]\vec{\gamma},\ \mathbf{r}_0 = \mathbf{r}_0 - \mathbf{A}([\hat{\mathbf{r}}_0, \ldots, \hat{\mathbf{r}}_{\ell-1}]\vec{\gamma})$
27. $\hat{\mathbf{U}} = [\hat{\mathbf{U}}_0 - \sum_{j=1}^{\ell} \gamma_j\hat{\mathbf{U}}_j]$
28. End while

where the vector $A(\hat{U}_k^{(j-1)}\vec{\alpha}^{(j)})$ is obtained by explicitly multiplying $\hat{U}_k^{(j-1)}\vec{\alpha}^{(j)}$ by $A$. Since $\hat{U}_{k+\ell}^{(0)} \equiv \hat{U}_{k+\ell}$ is expressed by

$$\hat{U}_{k+\ell} = \hat{U}_k^{(\ell)} - \gamma_{1,k}(K^{-1}A)\hat{U}_k^{(\ell)} - \cdots - \gamma_{\ell,k}(K^{-1}A)^\ell\hat{U}_k^{(\ell)}$$

at the polynomial step, we compute $(K^{-1}A)^i\hat{U}_k^{(j)}$ for $i = 0, 1, \ldots, j$ at the $j$th repetition: the columns of $(K^{-1}A)^i\hat{U}_k^{(j)}$ for $i = 0, 1, \ldots, j-1$ are obtained by vector updates, and multiplying the columns of $(K^{-1}A)^{j-1}\hat{U}_k^{(j)}$ by $K^{-1}A$ gives $(K^{-1}A)^j\hat{U}_k^{(j)}$.

By introducing $\hat{\boldsymbol{r}}_k^{(j)} \equiv K^{-1}\boldsymbol{r}_k^{(j)}$, (15) and (16) are reformulated to

$$\boldsymbol{x}_{k+\ell} = \boldsymbol{x}_k^{(\ell)} + \gamma_{1,k}\hat{\boldsymbol{r}}_k^{(\ell)} + \cdots + \gamma_{\ell,k}(K^{-1}A)^{\ell-1}\hat{\boldsymbol{r}}_k^{(\ell)},$$
$$\boldsymbol{r}_{k+\ell} = \boldsymbol{r}_k^{(\ell)} - A\{\gamma_{1,k}\hat{\boldsymbol{r}}_k^{(\ell)} + \cdots + \gamma_{\ell,k}(K^{-1}A)^{\ell-1}\hat{\boldsymbol{r}}_k^{(\ell)}\},$$

where the vector $A\{\sum_{i=1}^{\ell}\gamma_{i,k}(K^{-1}A)^{i-1}\hat{\boldsymbol{r}}_k^{(\ell)}\}$ is obtained by explicitly multiplying $\sum_{i=1}^{\ell}\gamma_{i,k}(K^{-1}A)^{i-1}\hat{\boldsymbol{r}}_k^{(\ell)}$ by $A$. The matrices $(K^{-1}A)^{i+1}\hat{U}_k^{(j-1)}$ for $i = 0, 1, \ldots, j-2$ are utilized to compute $(K^{-1}A)^i\hat{\boldsymbol{r}}_k^{(j)}$ by vector updates: $(K^{-1}A)^i\hat{\boldsymbol{r}}_k^{(j)} = (K^{-1}A)^i\hat{\boldsymbol{r}}_k^{(j-1)} - (K^{-1}A)^{i+1}\hat{U}_k^{(j-1)}\vec{\alpha}^{(j)}$. The vector $(K^{-1}A)^{j-1}\hat{\boldsymbol{r}}_k^{(j)}$ is obtained by multiplying $(K^{-1}A)^{j-2}\hat{\boldsymbol{r}}_k^{(j)}$ by $(K^{-1}A)$. Note that $(AK^{-1})^i\boldsymbol{r}_k^{(\ell)}$ for $i = 1, 2, \ldots, \ell$ are required in order to determine $\gamma_{i,k}$.

Our proposed variant with right preconditioning requires $\ell(s+1)+\ell+1$ MVs per cycle, but the number of multiplications by $K^{-1}$ is only $\ell(s+1)$.

Figs. 12 and 13 display the convergence histories of the original IDRstab and our variant with preconditioning for $(s, \ell) = (2, 6)$. Table 6 shows the numbers of cycles and MVs (the number of multiplications by $K^{-1}$), the computation

**Table 6**
Numbers of cycles and MVs (the number of multiplications by $K^{-1}$), computation times, and true relative residual norms of the original IDRstab and our variant with preconditioning for example 2.

| $(s, \ell)$ | Solver | Cycles | MVs | Time [s] | True res. |
|---|---|---|---|---|---|
| (2, 2) | IDRstab | 259 | 1557 (1556) | 8.54 | 1.09E−08 |
| | Our variant | 270 | 2432 (1622) | 7.39 | 1.06E−11 |
| (2, 4) | IDRstab | 138 | 1659 (1658) | 12.8 | 3.18E−10 |
| | Our variant | 145 | 2467 (1742) | 11.1 | 6.34E−12 |
| (2, 6) | IDRstab | 81 | 1461 (1460) | 13.6 | 5.48E−07 |
| | Our variant | 91 | 2277 (1640) | 12.7 | 2.63E−11 |
| (4, 2) | IDRstab | 97 | 975 (974) | 6.59 | 2.32E−09 |
| | Our variant | 103 | 1343 (1034) | 5.47 | 1.16E−12 |
| (4, 4) | IDRstab | 48 | 965 (964) | 8.66 | 6.44E−09 |
| | Our variant | 47 | 1179 (944) | 6.78 | 1.85E−12 |
| (4, 6) | IDRstab | 35 | 1055 (1054) | 11.8 | 2.08E−08 |
| | Our variant | 34 | 1262 (1024) | 9.67 | 1.00E−12 |
| (6, 2) | IDRstab | 57 | 805 (804) | 6.90 | 1.20E−07 |
| | Our variant | 64 | 1094 (902) | 5.77 | 1.13E−12 |
| (6, 4) | IDRstab | 25 | 707 (706) | 8.13 | 1.80E−10 |
| | Our variant | 25 | 831 (706) | 6.45 | 6.59E−13 |
| (6, 6) | IDRstab | 20 | 847 (846) | 12.1 | 1.65E−04 |
| | Our variant | 18 | 888 (762) | 9.00 | 1.57E−12 |

times, and the true relative residual 2-norms at termination. ILU(0) was used as the preconditioner. We refer to Section 5.2 for the system and the computational conditions.

From Figs. 12 and 13 and Table 6, we can observe the following. The approximate solutions obtained by our variant with preconditioning are more accurate than those obtained by the original preconditioned IDRstab. The numbers of cycles required for successful convergence of the original IDRstab and our variant with preconditioning are almost the same. The computation times for our variant with preconditioning are shorter than those for the original preconditioned IDRstab. Note that, from Tables 5 and 6, the convergence of the original IDRstab and our variant is enhanced by preconditioning, and the residual gap is slightly narrowed.

## 6. Concluding remarks

We have proposed a variant of IDRstab to narrow the residual gap. Our variant requires $\ell + 1$ additional MVs per cycle, but the extra costs can be compensated for by reducing the number of AXPYs in the case of sparse linear systems. Numerical experiments show that the explicit multiplications by $A$ play an important role in avoiding a large residual gap. The approximate solutions obtained by our proposed variant are more accurate than those obtained by the original IDRstab. As future work, we will analyze the influence of rounding errors on the residual gap and clarify the reasons why the vector $\boldsymbol{q}_k^{(j)}$ differs from $A\boldsymbol{p}_k^{(j)}$ in the original IDRstab.

## Acknowledgments

The authors would like to thank the reviewers for their careful reading and helpful suggestions. The authors would like to also thank Dr. Gerard L. G. Sleijpen (Utrecht University) for his helpful suggestions and the offering of MATLAB code of the original IDRstab. This work was partly supported by Grants-in-Aid for Scientific Research (C) No. 22560067 of Japan Society for the Promotion of Science (JSPS).

## Appendix

In order to reduce the residual gap, it is useful to replace the recursively computed residuals by the true residuals, but this approach may lead to a slower convergence speed (see [8, Section 3.1]). A strategy to cumulate groups of updates (referred to as *group-wise update*) for computing the approximations and residuals, which narrows the residual gap without loss of convergence speed, has been proposed in [8]. We apply the strategy of group-wise update to the original IDRstab and our proposed variant, and show the effects through numerical experiments.

We briefly explain the group-wise update [8]. Suppose that the approximations and the corresponding residuals are updated by

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \boldsymbol{p}_{k-1}, \qquad \boldsymbol{r}_k = \boldsymbol{r}_{k-1} - A\boldsymbol{p}_{k-1}.$$

**Table A.7**
Numbers of cycles and true relative residual norms of the original IDRstab and our variant with the strategy of group-wise update for examples 1 and 2.

| $(s, \ell)$ | Solver | Example 1 | | Example 2 | |
|---|---|---|---|---|---|
| | | Cycles | True res. | Cycles | True res. |
| (2, 2) | IDRstab | 508 | 4.89E−11 | 2145 | 2.93E−07 |
| | Our variant | 607 | 1.44E−13 | 1795 | 4.80E−13 |
| (2, 6) | IDRstab | 149 | 2.51E−10 | 559 | 4.60E−07 |
| | Our variant | 156 | 4.11E−13 | 458 | 2.31E−13 |
| (4, 4) | IDRstab | 114 | 5.74E−10 | 231 | 9.75E−07 |
| | Our variant | 123 | 9.73E−13 | 273 | 1.59E−13 |
| (6, 2) | IDRstab | 173 | 9.32E−09 | 326 | 4.77E−09 |
| | Our variant | 182 | 4.92E−13 | 391 | 7.23E−13 |
| (6, 6) | IDRstab | 52 | 3.70E−08 | 88 | 1.47E−07 |
| | Our variant | 65 | 4.85E−13 | 136 | 3.66E−13 |

Then the group-wise update is performed by

$$\boldsymbol{x}_k = \boldsymbol{x}_{\pi(j)} + \boldsymbol{q}_j, \qquad \boldsymbol{r}_k = \boldsymbol{r}_{\pi(j)} - A\boldsymbol{q}_j \tag{A.1}$$

with $\boldsymbol{q}_j \equiv \boldsymbol{p}_{\pi(j)} + \boldsymbol{p}_{\pi(j)+1} + \cdots + \boldsymbol{p}_{k-1}$ for $k = \pi(j+1)$, where $\pi(j) \in \mathbb{N}, \pi(0) = 0$ is an increasing sequence which is chosen strategically. The vector $\boldsymbol{q}_j$ is iteratively computed as $\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{p}_i$ with a starting vector $\boldsymbol{x} = \boldsymbol{0}$ for $\pi(j) \le i < \pi(j+1)$. $\boldsymbol{r}_{\pi(j+1)}$ is referred to as a *true local residual*.

For the original IDRstab and our variant, we perform the group-wise update of the approximations and residuals as follows. We compute the approximation by using the first formula in (A.1) if

$$\|\boldsymbol{r}_{k+\ell}\|_2 < \delta\|\boldsymbol{b}\|_2 \quad \text{and} \quad \|\boldsymbol{b}\|_2 \le \max_i \|\boldsymbol{r}_i\|_2, \tag{A.2}$$

where the residual $\boldsymbol{r}_{k+\ell}$ is obtained at the polynomial step, and $\max_i \|\boldsymbol{r}_i\|_2$ is the maximum of the primary residual norms since the previous group-wise update of the approximation. Following [8], we replace the recursively computed residual $\boldsymbol{r}_{k+\ell}$ by the true local residual, which is explicitly computed by using the second formula in (A.1), if

$$\|\boldsymbol{r}_{k+\ell}\|_2 < \delta \max_j \|\boldsymbol{r}_j\|_2 \quad \text{and} \quad \|\boldsymbol{b}\|_2 \le \max_j \|\boldsymbol{r}_j\|_2 \tag{A.3}$$

or (A.2) holds, where $\max_j \|\boldsymbol{r}_j\|_2$ is the maximum of the primary residual norms since the previous computation of the true local residual. As a result, the following procedures 1–4 are repeated after setting $\boldsymbol{x}' = \boldsymbol{x}_0$, $\boldsymbol{b}' = \boldsymbol{r}_0$, and $\boldsymbol{x}_0 = \boldsymbol{0}$ at the initialization.

1. $\boldsymbol{x}_{k+\ell}$ and $\boldsymbol{r}_{k+\ell}$ are computed recursively at the polynomial step.
2. Set $\max_i \|\boldsymbol{r}_i\|_2$ and $\max_j \|\boldsymbol{r}_j\|_2$ in (A.2) and (A.3), respectively.
3. If (A.2) or (A.3) holds, replace $\boldsymbol{r}_{k+\ell}$ by $\boldsymbol{b}' - A\boldsymbol{x}_{k+\ell}$.
4. If (A.2) holds, compute $\boldsymbol{x}' = \boldsymbol{x}' + \boldsymbol{x}_{k+\ell}$, and set $\boldsymbol{x}_{k+\ell} = \boldsymbol{0}$ and $\boldsymbol{b}' = \boldsymbol{r}_{k+\ell}$.

We compute $\boldsymbol{x}_{k+\ell} = \boldsymbol{x}' + \boldsymbol{x}_{k+\ell}$ when the iterations are terminated. Procedures 3 and 4 involve computation of the true local residual and the group-wise update of the approximation, respectively.

Table A.7 shows the numbers of cycles and the true relative residual 2-norms at termination of the original IDRstab and our variant with the strategy of group-wise update for examples 1 and 2 in Sections 5.1 and 5.2. The parameters $(s, \ell)$ were set at (2, 2), (2, 6), (4, 4), (6, 2), and (6, 6). The parameter $\delta$ was chosen as $10^{-3}$. We refer to Sections 5.1 and 5.2 for the computational conditions.

From Tables 4, 5 and A.7, we can observe the following. The residual gap of the original IDRstab is slightly reduced by the strategy. In the case of $(s, \ell) = (2, 2)$ for example 1, 10 digits are accurate for the approximate solutions obtained by the original IDRstab with the strategy, while 5 digits are accurate for those obtained by the original IDRstab (without the strategy). In the case of $(s, \ell) = (2, 6)$ for example 2, 6 digits are accurate for the original IDRstab with the strategy, while 3 digits are accurate for this case without the strategy. The approximate solutions obtained by our variant with the strategy attain the required accuracy for examples 1 and 2. Therefore, the strategy is useful for reducing the residual gap, but may not be sufficient for obtaining the required accuracy for the original IDRstab. The convergence of the original IDRstab and our variant with the strategy is comparable to that without the strategy in respect of the number of cycles.

## References

[1] P. Sonneveld, M.B. van Gijzen, IDR($s$): a family of simple and fast algorithms for solving large nonsymmetric linear systems, SIAM J. Sci. Comput. 31 (2008) 1035–1062.
[2] 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) 631–644.

[3] G.L.G. Sleijpen, P. Sonneveld, M.B. van Gijzen, Bi-CGSTAB as an induced dimension reduction method, Appl. Numer. Math. 60 (2010) 1100–1114.
[4] M. Tanio, M. Sugihara, GBi-CGSTAB($s$, $L$): IDR($s$) with higher-order stabilization polynomials, J. Comput. Appl. Math. 235 (2010) 765–784.
[5] G.L.G. Sleijpen, M.B. van Gijzen, Exploiting BiCGstab($\ell$) strategies to induce dimension reduction, SIAM J. Sci. Comput. 32 (2010) 2687–2709.
[6] O. Rendel, A. Rizvanolli, J.-P.M. Zemke, IDR: a new generation of Krylov subspace methods? Linear Algebra Appl. 439 (2013) 1040–1061.
[7] G.L.G. Sleijpen, D.R. Fokkema, BiCGstab($\ell$) for linear equations involving unsymmetric matrices with complex spectrum, Electron. Trans. Numer. Anal. 1 (1993) 11–32.
[8] G.L.G. Sleijpen, H.A. van der Vorst, Reliable updated residuals in hybrid Bi-CG methods, Computing 56 (1996) 141–163.
[9] G.L.G. Sleijpen, H.A. van der Vorst, D.R. Fokkema, BiCGstab($\ell$) and other hybrid Bi-CG methods, Numer. Algorithms 7 (1994) 75–109.
[10] J. van den Eshof, G.L.G. Sleijpen, Inexact Krylov subspace methods for linear systems, SIAM J. Matrix Anal. Appl. 26 (2004) 125–153 (electronic).
[11] G.L.G. Sleijpen, H.A. van der Vorst, Maintaining convergence properties of BiCGstab methods in finite precision arithmetic, Numer. Algorithms 10 (1995) 203–223.
[12] K. Aihara, K. Abe, E. Ishiwata, An alternative implementation of the IDRstab method saving vector updates, JSIAM Lett. 3 (2011) 69–72.
[13] D.H. Bailey, QD (C++/Fortran-90 double–double and quad-double package), http://crd-legacy.lbl.gov/~dhbailey/.
[14] A. Greenbaum, Estimating the attainable accuracy of recursively computed residual methods, SIAM J. Matrix Anal. Appl. 18 (1997) 535–551.
[15] W. Joubert, Lanczos methods for the solution of nonsymmetric systems of linear equations, SIAM J. Matrix Anal. Appl. 13 (1992) 926–943.