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(ℓ) 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(ℓ) 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(ℓ), 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 kth 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 ω_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 ℓ. 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: ℓ *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 ℓ times by using the projections $\Pi_i^{(j)}$ for $i = 0, 1, \ldots, j, j = 1, 2, \ldots, \ell$ which are defined by $\Pi_i^{(j)} \equiv I - A^iU_k^{(j-1)}\sigma_j^{-1}\tilde{R}_0^*A^{j-i}$ with $\sigma_j \equiv \tilde{R}_0^*A^jU_k^{(j-1)}$. Here, $\tilde{R}_0^*$ denotes the conjugate transpose of $\tilde{R}_0$, and the superscript "(j)" denotes the jth 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 $A^iU_k^{(j-1)}$ for $i = 0, 1, \ldots, j$, are generated at the $(j-1)$st ($j \leq \ell$) 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 jth 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 $A^iU_k^{(j-1)}$ are updated to $A^iU_k^{(j)}$ for $i = 0, 1, \ldots, j$ such that the columns of $A^jU_k^{(j)}$ 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 $A^iU_k^{(j)}\boldsymbol{e}_1$ for $i = 0, 1, \ldots, j$ are obtained by vector updates

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

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 $\boldsymbol{c}_q^{(j)} \equiv A(A^jU_k^{(j)}\boldsymbol{e}_q)$ is computed as the qth column of $A^{j+1}U_k^{(j)}$ by an explicit multiplication by A; after that, the $(q+1)$st columns $A^iU_k^{(j)}\boldsymbol{e}_{q+1}$ for $i = 0, 1, \ldots, j$ can