Journal of Computational and Applied Mathematics 236 (2011) 621–630

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

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

A variant of the IDR(s) method with the quasi-minimal residual strategy

Lei Du[a,*], Tomohiro Sogabe[b], Shao-Liang Zhang[a]

[a] Department of Computational Science and Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
[b] Graduate School of Information Science and Technology, Aichi Prefectural University, Nagakute-cho, Aichi-gun, Aichi, 480-1198, Japan

ARTICLE INFO

Keywords:
Induced dimension reduction
IDR(s)
Krylov subspace methods
Linear systems
QMR
QMRCGSTAB
QMRIDR(s)

ABSTRACT

The IDR(s) method proposed by Sonneveld and van Gijzen is an effective method for solving nonsymmetric linear systems, but usually with irregular convergence behavior. In this paper, we reformulate the relations of residuals and their auxiliary vectors generated by the IDR(s) method in matrix form. Then, using this new formulation and motivated by other QMR-type methods, we propose a variant of the IDR(s) method, called QMRIDR(s), for overcoming the disadvantage of its irregular convergence behavior. Both fast and smooth convergence behaviors of the QMRIDR(s) method can be shown. Numerical experiments are reported to show the efficiency of our proposed method.

1. Introduction

In this paper, we consider the solution of large, sparse and nonsymmetric linear systems of the form

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

where A is a nonsingular $n \times n$ real matrix, and $\boldsymbol{b}$ is a real vector of order n.

Over the past few decades, Krylov subspace methods for solving nonsymmetric linear systems (1) have been studied in depth since the appearance of pioneering work [1,2], such as the biconjugate gradient method (Bi-CG) [3], the generalized minimal residual method (GMRES) [4], the conjugate gradient squared method (CGS) [5], the biconjugate gradient stabilized method (Bi-CGSTAB) [6] and the generalized product-type method based on Bi-CG (GPBi-CG) [7]. There are more discussions on the Krylov subspace methods in the review papers [8–11] and books [12,13].

IDR(s), a class of generalized algorithms for the IDR method [14] for linear systems (1), was recently proposed by Sonneveld and van Gijzen in [15]. The IDR(s) method is different from traditional Krylov subspace methods, but with the relationship that IDR(1) is mathematically equivalent to Bi-CGSTAB for even IDR(1) residuals, and the IDR(s) method with $s > 1$ is competitive with the Krylov subspace methods. The relation between IDR and Bi-CGSTAB was discussed in [16]. IDR was explained by Gutknecht in [17]; it was viewed as a Petrov–Galerkin method and a Ritz–IDR version was given in [18]. Some variants based on the IDR method have been proposed. A new IDR(s) variant obtained by imposing bi-orthogonalization conditions was developed in [19]. Exploiting the merits of BiCGstab(ℓ) [20] for avoiding the potential breakdown, especially for skew-symmetric or nearly skew-symmetric systems, IDRStab and GBi-CGSTAB(s, L) were proposed with higher order stabilization polynomials in [21,22], respectively. A block version of IDR(s) was presented

[*] Corresponding author.
E-mail addresses: lei-du@na.cse.nagoya-u.ac.jp, du3stone@gmail.com (L. Du), sogabe@ist.aichi-pu.ac.jp (T. Sogabe), zhang@na.cse.nagoya-u.ac.jp (S.-L. Zhang).

0377-0427/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2011.07.027

for solving linear systems with multiple right-hand sides in [23]. The IDR method was also used to solve eigenvalue problems in [24].

Although the IDR(s) method is different from traditional Krylov subspace methods, its convergence history for the norms of residuals also shows quite an irregular convergence behavior like the Lanczos-type product methods. In practice, a method with smoother convergence behavior is more desirable. For example, when linear solvers are used as the intermediate steps for solving the partial differential equations (PDEs), a usual way of fixing the number of iteration steps is adopted to stop the inner iterations. In this case, better approximations can be obtained by the linear solvers with smoother convergence behavior. In this paper, we reformulate the relations of residuals and their auxiliary vectors generated by the IDR(s) method in matrix form. Then, on the basis of this new formulation and motivated by the QMR-type methods [25–27], we propose a variant of the IDR(s) method, called QMRIDR(s), to overcome the disadvantage of its irregular convergence behavior. Both fast and smooth convergence behaviors of the QMRIDR(s) method are expected.

This paper is organized as follows. A brief review of the IDR(s) method is given in Section 2. In Section 3, we reformulate the relations of residuals and their auxiliary vectors generated by the IDR(s) method and discuss the derivative process of the QMRIDR(s) method. Numerical experiments are reported in Section 4. Finally, we make some concluding remarks in Section 5.

Throughout this paper, for a vector $\boldsymbol{u}$ and a matrix M, let $\boldsymbol{u}^T$ be the transpose of the vector; $\|\boldsymbol{u}\|$ always denotes the Euclidean norm $\|\boldsymbol{u}\|_2 = \sqrt{\boldsymbol{u}^T\boldsymbol{u}}$. $\boldsymbol{u}_i$ denotes its ith element and $M_{i,j}$ is the entry in the ith row and jth column. I_n represents the identity matrix of size n. We use MATLAB notation $X(:, i : j)$ to denote a submatrix of X which consists of columns i to j. We will not emphasize the size of a matrix or vector if it is apparent from the context and there is no confusion.

2. The IDR(s) method

In this section, we review the IDR(s) method. Before that some preliminary information is given.

Definition 2.1. The Krylov subspace $\mathcal{K}_k(A, \boldsymbol{y})$ of order k for the matrix A and generating vector $\boldsymbol{y}$ is defined as

$$\mathcal{K}_k(A, \boldsymbol{y}) = \operatorname{span}\{\boldsymbol{y}, A\boldsymbol{y}, \ldots, A^{k-1}\boldsymbol{y}\}.$$

From the definition, we know that $\mathcal{K}_k(A, \boldsymbol{y}) \subseteq \mathcal{K}_{k+1}(A, \boldsymbol{y})$. Furthermore, there is a value v, called the grade of $\boldsymbol{y}$, satisfying $\mathcal{K}_{v-1}(A, \boldsymbol{y}) \subset \mathcal{K}_v(A, \boldsymbol{y}) = \mathcal{K}_{v+1}(A, \boldsymbol{y}) = \cdots$, which implies that subspace $\mathcal{K}_k(A, \boldsymbol{y})$ is invariant for $k \geq v$.

Let $\boldsymbol{x}_0$ be an initial approximation of systems (1) and $\boldsymbol{r}_0$ be the corresponding residual $\boldsymbol{b} - A\boldsymbol{x}_0$. Suppose that $\mathcal{G}_0 = \mathcal{K}_v(A, \boldsymbol{r}_0)$ with v being the grade of $\boldsymbol{r}_0$ and let $\mathcal{S}$ be a subspace in $\mathbb{R}^n$. Define a sequence of subspaces $\mathcal{G}_j$ by recursion as

$$\mathcal{G}_j = (I - \omega_j A)(\mathcal{G}_{j-1} \cap \mathcal{S})$$

in which the ω_j's are nonzero parameters.

Under the assumption that subspace $\mathcal{S} \cap \mathcal{G}_0$ does not contain a nontrivial invariant subspace of A, the following result of the IDR theorem [15,14] is obtained: $\mathcal{G}_j \subset \mathcal{G}_{j-1}$, i.e., $\mathcal{G}_j$ is a proper subset of $\mathcal{G}_{j-1}$. This fact implies that the sequence of nested subspaces $\mathcal{G}_j$ is finite and $\mathcal{G}_j = \{\boldsymbol{0}\}$ for some $j \leq v$.

On the basis of the theorem, the IDR(s) method was proposed in [15]. When the $s + 1$ residuals $\boldsymbol{r}_{k-s}, \ldots, \boldsymbol{r}_k$ in $\mathcal{G}_{j-1}$ have been constructed, the idea of the IDR(s) method is to construct the next $s + 1$ residuals $\boldsymbol{r}_{k+1}, \ldots, \boldsymbol{r}_{k+s+1}$ in $\mathcal{G}_j$. The construction process continues with the proper choice of the parameter ω_j, and approximations $\boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_{k+s+1}$ are extracted simultaneously. Finally, when the residual is constructed in $\{\boldsymbol{0}\}$, the exact solution will be obtained. The main derivative process of the IDR(s) method is shown as follows.

Let P be a prescribed matrix with the order of $n \times s$. The null space of the transpose of P is set as the subspace $\mathcal{S}$, i.e., $\mathcal{S} = \mathcal{N}(P^T)$. The suggestion of orthogonalizing a set of random vectors for P was made in [15].

Suppose that $\Delta\boldsymbol{r}_i = \boldsymbol{r}_{i+1} - \boldsymbol{r}_i$; an auxiliary vector $\boldsymbol{v}_k$ is defined as

$$\boldsymbol{v}_k = \boldsymbol{r}_k - \sum_{l=1}^{s} \gamma_l \Delta\boldsymbol{r}_{k-l} \tag{2}$$

$$= (1 - \gamma_1)\boldsymbol{r}_k + \sum_{l=1}^{s-1} (\gamma_l - \gamma_{l+1})\boldsymbol{r}_{k-l} + \gamma_s \boldsymbol{r}_{k-s} \tag{3}$$

in which the γ_l's are parameters. It is obvious that $\boldsymbol{v}_k \in \mathcal{G}_{j-1}$ for any γ_l's. Moreover, $\boldsymbol{v}_k$ is also limited in $\mathcal{S}$ by the setting of parameters γ_l in the IDR(s) method. From the definition of $\mathcal{S}$, the parameters γ_l can be determined under the condition that

$$P^T\boldsymbol{v}_k = 0. \tag{4}$$

Using $\boldsymbol{v}_k$, a new residual $\boldsymbol{r}_{k+1}$ is constructed as

$$\boldsymbol{r}_{k+1} = (I - \omega_j A)\boldsymbol{v}_k. \tag{5}$$

From the definition of the subspace $\mathcal{G}_j$, we see that $\boldsymbol{r}_{k+1} \in \mathcal{G}_j$. The parameter ω_j is determined by minimizing the 2-norm of $\boldsymbol{r}_{k+1}$, which gives $\omega_j = \boldsymbol{v}_k^T A\boldsymbol{v}_k / \|A\boldsymbol{v}_k\|^2$.

Suppose that $\Delta\boldsymbol{x}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i$; the relation $A\Delta\boldsymbol{x}_i = -\Delta\boldsymbol{r}_i$ always holds. Now, the corresponding approximation $\boldsymbol{x}_{k+1}$ can be derived from relations (2) and (5) as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \omega_j \boldsymbol{v}_k - \sum_{l=1}^{s} \gamma_l \Delta\boldsymbol{x}_{k-l}. \tag{6}$$

In the following s iterations, the foremost residual is replaced by the new one and the above process can be cycled to construct the other intermediate residuals $\boldsymbol{r}_{k+2}, \ldots, \boldsymbol{r}_{k+s+1}$. Giving consideration to the computational cost, parameter ω_j may be kept the same in these s iterations. Finally, all $s+1$ residuals $\boldsymbol{r}_{k+1}, \ldots, \boldsymbol{r}_{k+s+1}$ in $\mathcal{G}_j$ are obtained. Naturally, $s+1$ more residuals in $\mathcal{G}_{j+1}$ can be constructed in the same way, and so on. We conclude this section by recalling the algorithm of the IDR(s) method [15] in Algorithm 1.

Algorithm 1 IDR(s)

```
1:  Suppose that $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, P \in \mathbb{R}^{n\times s}, j = 1$;
2:  for $i = 1, 2, \ldots, s$ do
3:    $\boldsymbol{v} = A\boldsymbol{r}_{i-1}, \omega = \boldsymbol{v}^T\boldsymbol{r}_{i-1}/\boldsymbol{v}^T\boldsymbol{v}$;
4:    $\Delta X(:, i) = \omega\boldsymbol{r}_{i-1}, \Delta R(:, i) = -\omega\boldsymbol{v}$;
5:    $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \Delta X(:, i), \boldsymbol{r}_i = \boldsymbol{r}_{i-1} + \Delta R(:, i)$;
6:  end for
7:  $M = P^T\Delta R; \boldsymbol{h} = P^T\boldsymbol{r}_i$;
8:  while the stopping criterion is not satisfied do
9:    for $k = 0, 1, \ldots, s$ do
10:     Solve $c$ from $M\boldsymbol{c} = \boldsymbol{h}$;
11:     $\boldsymbol{q} = -\Delta R\boldsymbol{c}, \boldsymbol{v} = \boldsymbol{r}_i + \boldsymbol{q}$;
12:     if $k == 0$ then
13:       $\boldsymbol{t} = A\boldsymbol{v}, \omega = \boldsymbol{t}^T\boldsymbol{v}/\boldsymbol{t}^T\boldsymbol{t}$;
14:       $\Delta R(:, j) = \boldsymbol{q} - \omega\boldsymbol{t}, \Delta X(:, j) = -\Delta X\boldsymbol{c} + \omega\boldsymbol{v}$;
15:     else
16:       $\Delta X(:, j) = -\Delta X\boldsymbol{c} + \omega\boldsymbol{v}, \Delta R(:, j) = -A\Delta X(:, j)$;
17:     end if
18:     $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \Delta R(:, j), \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta X(:, j)$;
19:     $\delta\boldsymbol{m} = P^T\Delta R(:, j), M(:, j) = \delta\boldsymbol{m}, \boldsymbol{h} = \boldsymbol{h} + \delta\boldsymbol{m}$;
20:     $i = i + 1, j = j + 1, j = \mathrm{mod}(j - 1, s) + 1$;
21:   end for
22: end while
```

3. A QMR variant of the IDR(s) method

In this section, we reconsider the relations of residuals and their auxiliary vectors generated by the IDR(s) method and discuss the derivative process of the QMRIDR(s) method.

First we define some new notation; suppose that

$$\boldsymbol{y}_k = \begin{cases} \boldsymbol{r}_k & k = 0, 1, \ldots, s-1, \\ \boldsymbol{v}_k & \text{otherwise,} \end{cases}$$

and define

$$Y_k = [\boldsymbol{y}_0 \quad \boldsymbol{y}_1 \quad \cdots \quad \boldsymbol{y}_{k-1}],$$
$$W_{k+1} = [\boldsymbol{r}_0 \quad \boldsymbol{r}_1 \quad \cdots \quad \boldsymbol{r}_{k-1} \quad \boldsymbol{r}_k].$$

In the initialization part of the IDR(s) algorithm, the residual $\boldsymbol{r}_k$ is generated as

$$\boldsymbol{r}_k = (I - \omega_k A)\boldsymbol{r}_{k-1}, \quad k = 1, 2, \ldots, s, \tag{7}$$

and ω_k is determined by minimizing $\|\boldsymbol{r}_k\|$ per iteration. Suppose that $h_{k,k} = 1/\omega_k$ and $h_{k+1,k} = -1/\omega_k$ for $k = 1, 2, \ldots, s$; we can transform the formula (7) into

$$A\boldsymbol{r}_{k-1} = h_{k,k}\boldsymbol{r}_{k-1} + h_{k+1,k}\boldsymbol{r}_k, \quad k = 1, 2, \ldots, s. \tag{8}$$

The formula $\boldsymbol{r}_{k+1} = (I - \omega_{k+1}A)\boldsymbol{v}_k$ always holds since $k = s$. Parameter ω_{k+1} will be computed only when $\text{mod}(k + 1, s + 1) = 0$, which means that $\omega_{j(s+1)} = \omega_{j(s+1)+1} = \cdots = \omega_{j(s+1)+s}$ for $j = 1, 2, \ldots$. From the relations (3) and (4), suppose that $h_{k+1-s,k+1} = \gamma_s/\omega_{k+1}$, $h_{k+1,k+1} = (1 - \gamma_1)/\omega_{k+1}$, $h_{k+2,k+1} = -1/\omega_{k+1}$ and $h_{k+1-l,k+1} = (\gamma_l - \gamma_{l+1})/\omega_{k+1}$ for $l = 1, 2, \ldots, s - 1$; formula $\boldsymbol{r}_{k+1} = (I - \omega_{k+1}A)\boldsymbol{v}_k$ can be reformulated as

$$\begin{aligned} A\boldsymbol{v}_k &= (\boldsymbol{v}_k - \boldsymbol{r}_{k+1})/\omega_{k+1} = \left((1-\gamma_1)\boldsymbol{r}_k + \sum_{l=1}^{s-1}(\gamma_l - \gamma_{l+1})\boldsymbol{r}_{k-l} + \gamma_s\boldsymbol{r}_{k-s} - \boldsymbol{r}_{k+1}\right) \Big/ \omega_{k+1} \\ &= h_{k+2,k+1}\boldsymbol{r}_{k+1} + h_{k+1,k+1}\boldsymbol{r}_k + \sum_{l=1}^{s-1} h_{k+1-l,k+1}\boldsymbol{r}_{k-l} + h_{k+1-s,k+1}\boldsymbol{r}_{k-s}. \end{aligned} \tag{9}$$

Now, we can write the series of relations (8) and (9) in matrix form as

$$AY_k = W_{k+1}H_k \tag{10}$$

where

$$H_k = \begin{pmatrix} h_{1,1} & \cdots & h_{1,s+1} & 0 & \\ h_{2,1} & \cdots & h_{2,s+1} & \ddots & \\ & \ddots & \vdots & \ddots & h_{k-s,k} \\ & & h_{s+2,s+1} & \ddots & \vdots \\ & 0 & & \ddots & \vdots \\ & & & & h_{k+1,k} \end{pmatrix}$$

is a $(k + 1) \times k$ upper Hessenberg matrix with the bandwidth of $s + 2$.

One main relation that we should mention here is that the column vectors of Y_k and W_k will span the same subspace of $\mathcal{K}_k(A, \boldsymbol{r}_0)$.

Our idea of this paper is to construct a new approximation $\tilde{\boldsymbol{x}}_k$ by using the basis of $\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k-1}$ generated by the IDR(s) method. We represent $\tilde{\boldsymbol{x}}_k$ as $\tilde{\boldsymbol{x}}_k = \boldsymbol{x}_0 + Y_k\boldsymbol{z}_k$ for $\boldsymbol{z}_k \in \mathbb{R}^k$. From the relation (10), the residual $\tilde{\boldsymbol{r}}_k = \boldsymbol{b} - A\tilde{\boldsymbol{x}}_k$ can be rewritten as

$$\begin{aligned} \tilde{\boldsymbol{r}}_k &= \boldsymbol{r}_0 - AY_k\boldsymbol{z}_k \\ &= W_{k+1}(\boldsymbol{e}_1 - H_k\boldsymbol{z}_k), \end{aligned} \tag{11}$$

where $\boldsymbol{e}_1 = [1\,0\,\cdots\,0]^T \in \mathbb{R}^{k+1}$.

The ideal case for $\tilde{\boldsymbol{x}}_k$ is to determine $\boldsymbol{z}_k$ by minimizing the 2-norm of $\tilde{\boldsymbol{r}}_k$, i.e., $\boldsymbol{z}_k = \arg\min_{\boldsymbol{z}} \|W_{k+1}(\boldsymbol{e}_1 - H_k\boldsymbol{z})\|$. As the vectors $\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_k$ are not orthogonal to each other, it is costly to orthogonalize them and the orthogonality cannot be guaranteed in finite precision computations. In addition, all basis vectors have to be stored. These features make it impractical to obtain $\tilde{\boldsymbol{x}}_k$ by minimizing $\|\tilde{\boldsymbol{r}}_k\|$. As a compromise between the optimality and storability, we apply the strategy of the QMR method to the relation (11), considering the banded structure of H_k. In other words, we solve the least-squares problem

$$\min_{\boldsymbol{z}} \|\boldsymbol{e}_1 - H_k\boldsymbol{z}\| \tag{12}$$

instead. Usually a scaling matrix $\Omega_{k+1} = \text{diag}(\delta_0, \ldots, \delta_k)$ is used to narrow the gap between $\|\tilde{\boldsymbol{r}}_k\|$ and $\|\boldsymbol{e}_1 - T_k\boldsymbol{z}_k\|$. Here, we define $\delta_i = \|\boldsymbol{r}_i\|$ to make the columns of W_{k+1} of unit norm, i.e., $\tilde{\boldsymbol{r}}_k = W_{k+1}\Omega_{k+1}^{-1}\Omega_{k+1}(\boldsymbol{e}_1 - H_k\boldsymbol{z}_k)$. Suppose that $\tilde{H}_k = \Omega_{k+1}H_k$; the problem (12) will be replaced by

$$\min_{\boldsymbol{z}} \|\delta_0\boldsymbol{e}_1 - \tilde{H}_k\boldsymbol{z}\|, \tag{13}$$

and we call $\delta_0\boldsymbol{e}_1 - \tilde{H}_k\boldsymbol{z}$ a quasi-residual.

A brief derivation is presented for computing the approximation $\tilde{\boldsymbol{x}}_k$ in the following steps. To solve problem (13), QR decomposition with Givens rotations is the best choice due to the special structure of $\tilde{H}_k$. We just need to do k Givens rotations and the ith Givens rotation G_i will zero out $\delta_i h_{i+1,i}$ for $i = 1, \ldots, k$. G_i is defined as

$$G_i = \begin{bmatrix} I_{i-1} & & & \\ & c_i & s_i & \\ & -s_i & c_i & \\ & & & I_{k-i} \end{bmatrix}$$

in which parameters $c_i, s_i \in \mathbb{R}$ and $c_i^2 + s_i^2 = 1$. The setting of c_i, s_i will be given later.

Suppose that $Q_k = G_k G_{k-1} \cdots G_1$; the QR decomposition of $\tilde{H}_k$ is written as

$$\tilde{H}_k = Q_k^T \begin{bmatrix} R_k \\ \mathbf{0} \end{bmatrix},$$

where R_k is a upper triangular $k \times k$ matrix with the bandwidth of $s+2$ as

$$R_k = \begin{pmatrix} r_{1,1} & \cdots & r_{1,s+2} & 0 & \\ & \ddots & \ddots & \ddots & \\ & & r_{s+2,s+2} & \ddots & r_{k-s-1,k} \\ & 0 & & \ddots & \vdots \\ & & & & r_{k,k} \end{pmatrix}. \tag{14}$$

As Q_k is an orthonormal matrix, problem (13) is equivalent to

$$\min_{\mathbf{z}} \|\delta_0 Q_k \mathbf{e}_1 - Q_k \tilde{H}_k \mathbf{z}\| = \min_{\mathbf{z}} \left\| \delta_0 Q_k \mathbf{e}_1 - \begin{bmatrix} R_k \\ \mathbf{0} \end{bmatrix} \mathbf{z} \right\|. \tag{15}$$

Suppose that $\delta_0 Q_k \mathbf{e}_1 = [\tau_1, \ldots, \tau_k, \eta_{k+1}]^T = [\mathbf{t}_k^T, \eta_{k+1}]^T$ and consider the definition of Q_k; the following iterative relations always hold:

$$\eta_1 = \delta_0, \quad \tau_i = c_i \eta_i \quad \text{and} \quad \eta_{i+1} = -s_i \eta_i \quad \text{for } i = 1, \ldots, k. \tag{16}$$

When we have obtained the $k-1$ pairs of $(c_1, s_1), \ldots, (c_{k-1}, s_{k-1})$ for the Givens rotations $G_1, \ldots, G_{k-1}$ and we suppose that $ks = \min\{k-1, s+1\}$, then the following relation holds:

$$\begin{bmatrix} r_{1,k} \\ \vdots \\ r_{k-1,k} \\ \mu \\ \delta_k h_{k+1,k} \end{bmatrix} = Q_{k-1} \tilde{H}_k \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{17}$$

in which $r_{1,k}, \ldots, r_{k-ks-1,k}$ are zero entries, and we can obtain the values of $r_{k-ks,k}, \ldots, r_{k-1,k}$ simultaneously. Meanwhile $[\mathbf{t}_{k-1}^T, \eta_k, 0]^T = \delta_0 Q_{k-1} \mathbf{e}_1$. To zero out the last element $\delta_k h_{k+1,k}$ in formula (17), we can set the pair (c_k, s_k) for G_k as

$$c_k = |\mu| / \sqrt{\mu^2 + (\delta_k h_{k+1,k})^2},$$
$$s_k = c_k \delta_k h_{k+1,k} / \mu \quad (\text{or } s_k = 1 \text{ when } \mu = 0).$$

Applying G_k to formula (17), we can get $r_{k,k} = c_k \mu + s_k \delta_k h_{k+1,k}$. Values of τ_k and η_{k+1} can be computed according to relations (16).

We note that at most $s+2$ Givens rotations are applied to each column of $\tilde{H}_k$ in the QR decomposition, not all the k Givens rotations. From relation (15), we can determine $\mathbf{z}_k$ by solving $R_k \mathbf{z}_k = \mathbf{t}_k$, and represent our approximation $\tilde{\mathbf{x}}_k$ explicitly as

$$\tilde{\mathbf{x}}_k = \mathbf{x}_0 + Y_k R_k^{-1} \mathbf{t}_k. \tag{18}$$

Here we will show another way to update the approximation $\tilde{\mathbf{x}}_k$, instead of using the formula (18) directly, in order to reduce the memory requirement for storing all basis vectors. Suppose that

$$[\mathbf{f}_1 \mathbf{f}_2 \cdots \mathbf{f}_{k-1}] = Y_{k-1} R_{k-1}^{-1}, \tag{19}$$

$ks = \min\{k-1, s+1\}$ and $\mathbf{w} = [0, \ldots, 0, r_{k-ks,k}, \ldots, r_{k-1,k}]^T$; then we have the relations

$$\begin{aligned} Y_k R_k^{-1} &= [Y_{k-1}, \mathbf{y}_{k-1}] \begin{bmatrix} R_{k-1} & \mathbf{w} \\ 0 & r_{k,k} \end{bmatrix}^{-1} \\ &= [Y_{k-1}\ \mathbf{y}_{k-1}] \begin{bmatrix} R_{k-1}^{-1} & -R_{k-1}^{-1} \mathbf{w} / r_{k,k} \\ 0 & 1/r_{k,k} \end{bmatrix} \\ &= [Y_{k-1} R_{k-1}^{-1} (\mathbf{y}_{k-1} - Y_{k-1} R_{k-1}^{-1} \mathbf{w}) / r_{k,k}] \\ &= \left[\mathbf{f}_1 \mathbf{f}_2 \cdots \mathbf{f}_{k-1} \left(\mathbf{y}_{k-1} - \sum_{i=1}^{ks} r_{k-i,k} \mathbf{f}_{k-i} \right) \Big/ r_{k,k} \right]. \end{aligned} \tag{20}$$

Table 1
A comparison of the computational costs.

	Mvs	DOTs	AXPYs
QMRCGSTAB	2	6	8
QMRIDR(s)	$s+1$	s^2+2s+3	$3s^2+6s+4$
IDR(s)	$s+1$	s^2+s+2	$2s^2+7s/2+5/2$

Table 2
The order (n) and the number of nonzero elements (nnz) of the test matrices.

Matrix	n	nnz	Application discipline
Cdde1	961	4 681	Computational fluid dynamics
Pde2961	2961	14 585	Partial differential equations
Rdb1250	1250	7 300	Chemical engineering
Sherman4	1104	3 786	Oil reservoir modeling

From relations (19) and (20), we see that the columns of $Y_{k-1}R_{k-1}^{-1}$ are equal to the corresponding columns of $Y_kR_k^{-1}$. The last column $\boldsymbol{f}_k$ of $Y_kR_k^{-1}$ can be computed as

$$\boldsymbol{f}_k = \left(\boldsymbol{y}_{k-1} - \sum_{i=1}^{ks} r_{k-i,k}\boldsymbol{f}_{k-i}\right) \Big/ r_{k,k}. \tag{21}$$

According to the relations (16) and (19)–(21), we can update the approximation $\tilde{\boldsymbol{x}}_k$ using the form

$$\tilde{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_{k-1} + \tau_k\boldsymbol{f}_k. \tag{22}$$

The resulting algorithm is referred to as QMRIDR(s), which is given in Algorithm 2. Approximate solutions are obtained by using the QMR strategy from the ($s+1$)th iteration; the first s iterations generate the same approximations as the IDR(s) method and new notation is used in Algorithm 2. Although the IDR(1) method is mathematically equivalent to the Bi-CGSTAB method, we emphasize that there are some differences between their variants QMRIDR(1) and QMRCGSTAB. The main difference is that the former method is based on a tridiagonal matrix, but the latter is based on a lower bidiagonal matrix. The techniques used for deriving them are also different.

Two kinds of stopping criteria can be used to end the iterations in our algorithm. One choice is $\|\tilde{\boldsymbol{r}}_k\|$ and the residual $\tilde{\boldsymbol{r}}_k$ can be updated at low cost per iteration. The other is to use the estimate value $\sqrt{k+1}|\eta_{k+1}|$, which satisfies $\|\tilde{\boldsymbol{r}}_k\| \le \sqrt{k+1}|\eta_{k+1}|$. A mixed strategy of adopting both criteria can be used—for example, using the estimate value in the first stage and then computing the norm of $\tilde{\boldsymbol{r}}_k$.

A comparison of the computational costs between the IDR-type methods and the QMRCGSTAB method is given in the Table 1. Under the assumption of taking the same number of matrix–vector products per iteration, we see that the QMRIDR(s) method needs slightly more AXPY operations than the other two methods.

4. Numerical experiments

In this section, we report some numerical experiments with the IDR(s), QMRIDR(s) and QMRCGSTAB methods. Parameter s was equal to 1 and 4. In all cases, the iterations were started with $\boldsymbol{x}_0 = \boldsymbol{0}$, and $\boldsymbol{b} = [1, \dots, 1]^T$. Elements of the matrix $P^{n\times s}(s > 1)$ in (QMR)IDR(s) algorithms were random values distributed in the interval (0, 1). The initial vector $\tilde{\boldsymbol{r}}_0$ in the QMRCGSTAB algorithm and $P^{n\times 1}$ in (QMR)IDR(1) were set as the first column of $P^{n\times s}$. The IDR(s) algorithm used as the stopping criterion $\|\boldsymbol{r}_k\|/\|\boldsymbol{b}\| \le 10^{-8}$; both the QMRIDR($s$) and QMRCGSTAB methods stopped when $\sqrt{k+1}|\eta_{k+1}|/\|\boldsymbol{b}\| \le 10^{-8}$.

Experiments were performed on a Redhat linux system (64 bit) with an AMD Phenom(tm) 9500 Quad-Core Processor using double-precision arithmetic. Codes were written in the C++ language and compiled with GCC 4.1.2. All test matrices in this section were taken from the Matrix Market collection [28] and the University of Florida sparse matrix collection [29]. The convergence histories show the number of matrix–vector products (on the horizontal axis) versus the $\log_{10}$ of the true relative residual 2-norms (on the vertical axis) defined as $\|\boldsymbol{b} - A\boldsymbol{x}_k\|/\|\boldsymbol{b}\|$ in all figures.

4.1. Experiments without preconditioning

In this subsection, several matrices of small size were used as our test matrices. The orders, the numbers of nonzero elements and the application disciplines of these test matrices are listed in Table 2.

The results shown in Figs. 1–4 were computed without preconditioning. From Figs. 1–4, we have the following observations. (a) Except in Fig. 3, all methods converged; IDR(4) and QMRIDR(4) converged the fastest. Indeed the QMRIDR(s)

Algorithm 2 QMRIDR(s)

```
1: Initialize $\boldsymbol{x}_0, P \in \mathbb{R}^{n\times s}$ and compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$;
2: for $i = 0, 1, \ldots, s-1$ do
3:   $\boldsymbol{v} = A\boldsymbol{r}_i, \omega = \boldsymbol{v}^T\boldsymbol{r}_i/\boldsymbol{v}^T\boldsymbol{v}, \Delta X(:, i+1) = \omega\boldsymbol{r}_i, \Delta R(:, i+1) = -\omega\boldsymbol{v}$;
4:   $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta X(:, i+1), \boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \Delta R(:, i+1), H_{i,i} = \|\boldsymbol{r}_i\|/\omega, H_{i+1,i} = -\|\boldsymbol{r}_{i+1}\|/\omega$;
5: end for
6: $i = s$;
7: while the stopping criterion is not satisfied do
8:   $M = P^T\Delta R, \boldsymbol{h} = P^T\boldsymbol{r}_i$;
9:   for $k = 0, 1, \ldots, s$ do
10:    Solve $\boldsymbol{c} = [c_1, \ldots, c_s]^T$ from $M\boldsymbol{c} = \boldsymbol{h}, \boldsymbol{q} = -\Delta R\boldsymbol{c}, \boldsymbol{v} = \boldsymbol{r}_i + \boldsymbol{q}$;
11:    if $k == 0$ then
12:      $\boldsymbol{t} = A\boldsymbol{v}, \omega = \boldsymbol{t}^T\boldsymbol{v}/\boldsymbol{t}^T\boldsymbol{t}, \Delta\boldsymbol{r} = \boldsymbol{q} - \omega\boldsymbol{t}, \Delta\boldsymbol{x} = -\Delta X\boldsymbol{c} + \omega\boldsymbol{v}$;
13:    else
14:      $\Delta\boldsymbol{x} = -\Delta X\boldsymbol{c} + \omega\boldsymbol{v}, \Delta\boldsymbol{r} = -A\Delta\boldsymbol{x}$;
15:    end if
16:    $\Delta R(:, 1:s-1) = \Delta R(:, 2:s), \Delta R(:, s) = \Delta\boldsymbol{r}, \Delta X(:, 1:s-1) = \Delta X(:, 2:s), \Delta X(:, s) = \Delta\boldsymbol{x}$;
17:    $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \Delta\boldsymbol{r}, M(:, 1:s-1) = M(:, 2:s), \delta\boldsymbol{m} = P^T\Delta\boldsymbol{r}, M(:, s) = \delta\boldsymbol{m}, \boldsymbol{h} = \boldsymbol{h} + \delta\boldsymbol{m}$;
18:    if $i == s$ then
19:      $H = (h_{j,k}) = 0\ (j = 1, \ldots, s+2, k = 1, \ldots, s+1)$;
20:      $h_{1,s+1} = c_1\|\boldsymbol{r}_0\|/\omega, h_{s+1,s+1} = (1-c_s)\|\boldsymbol{r}_s\|/\omega, h_{s+2,s+1} = -\|\boldsymbol{r}_{s+1}\|/\omega$;
21:      for $l = 2, 3, \ldots, s$ do
22:        $h_{l,s+1} = (c_l - c_{l-1})\|\boldsymbol{r}_{l-1}\|/\omega$;
23:      end for
24:      $G = I_{s+2}, \boldsymbol{e}^T = [\|\boldsymbol{r}_0\|, 0, \ldots, 0]_{s+2}$;
25:      for $l = 1, 2, \ldots, s+1$ do
26:        $\tilde{c}_l = |h_{l,l}|/\sqrt{h_{l,l}^2 + h_{l+1,l}^2}, \tilde{s}_l = \tilde{c}_l h_{l+1,l}/h_{l,l}$ (when $h_{l,l} = 0, \tilde{s}_l = 1$);
27:        $G_{l,l} = \tilde{c}_l, G_{l+1,l+1} = \tilde{c}_l, G_{l,l+1} = \tilde{s}_l, G_{l+1,l} = \tilde{s}_l, H = GH, \boldsymbol{e} = G\boldsymbol{e}$;
28:      end for
29:      $F = [\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{s-1}, \boldsymbol{r}]H^{-1}(1:s+1, 1:s+1), \boldsymbol{x}_{i+1} = \boldsymbol{x}_0 + F\boldsymbol{e}(1:s+1), \eta = e_{s+2}$;
30:    else
31:      $\beta_1 = 0, \beta_2 = c_1\|\boldsymbol{r}_{i-s}\|/\omega, \beta_{s+2} = (1-c_s)\|\boldsymbol{r}_i\|/\omega, \beta_{s+3} = -\|\boldsymbol{r}_{i+1}\|/\omega$;
32:      for $l = 3, 4, \ldots, s+1$ do
33:        $\beta_l = (c_{l-1} - c_{l-2})\|\boldsymbol{r}_{i-s+l-2}\|/\omega$;
34:      end for
35:      for $l = 1, 2, \ldots, s+1$ do
36:        $$\begin{bmatrix}\beta_l\\ \beta_{l+1}\end{bmatrix} = \begin{bmatrix}\tilde{c}_{i-s+l} & \tilde{s}_{i-s+l}\\ -\tilde{s}_{i-s+l} & \tilde{c}_{i-s+l}\end{bmatrix}\begin{bmatrix}\beta_l\\ \beta_{l+1}\end{bmatrix};$$
37:      end for
38:      $\tilde{c}_{i+2} = |\beta_{s+2}|/\sqrt{\beta_{s+2}^2 + \beta_{s+3}^2}, \tilde{s}_{i+2} = \tilde{c}_{i+2}\beta_{s+3}/\beta_{s+2}$ (when $\beta_{s+2} = 0, \tilde{s}_{i+2} = 1$)
39:      $\beta_{s+2} = \tilde{c}_{i+2}\beta_{s+2} + \tilde{s}_{i+2}\beta_{s+3}, \tau = \tilde{c}_{i+2}\eta, \eta = -\tilde{s}_{i+2}\eta$;
40:      $\boldsymbol{f}_{i+1} = \left(\boldsymbol{v} - \sum_{l=0}^{s}\beta_{l+1}\boldsymbol{f}_{i-s+l}\right)/\beta_{s+2}, \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \tau\boldsymbol{f}_{i+1}$;
41:    end if
42:    $i = i + 1$;
43:  end for
44: end while
```

method had smoother convergence curves than the IDR(s) method; peaks that appeared in the convergence histories of the IDR(s) method disappeared in those of the QMRIDR(s) method. Even though the two methods used different stopping criteria, the QMRIDR(s) method just needed a few more iteration steps to terminate the algorithm than the IDR(s) method. (b) In Fig. 3, QMRIDR(4) stopped when the stopping criterion was satisfied; the convergence history of its true relative residual 2-norm stagnated, but close to converging. (c) The QMRIDR(1) and QMRCGSTAB methods converged for all examples, QMRCGSTAB cost a few more matrix–vector products than QMRIDR(1) in Fig. 2, but QMRCGSTAB converged faster than QMRIDR(1) in Fig. 3.

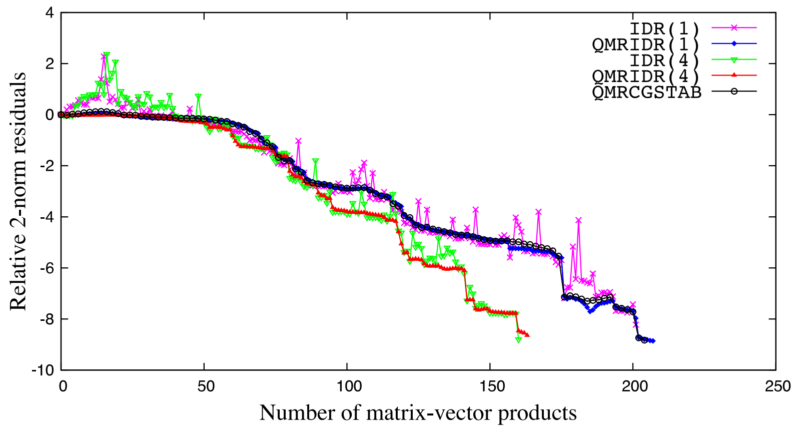

Fig. 1. Cdde1.

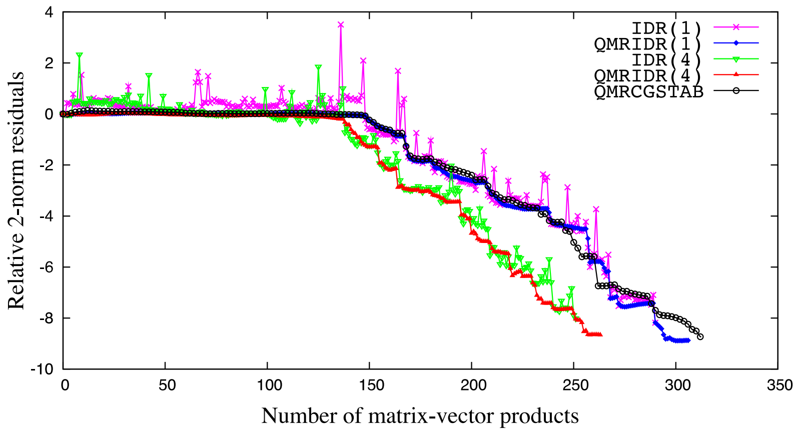

Fig. 2. Pde2961.

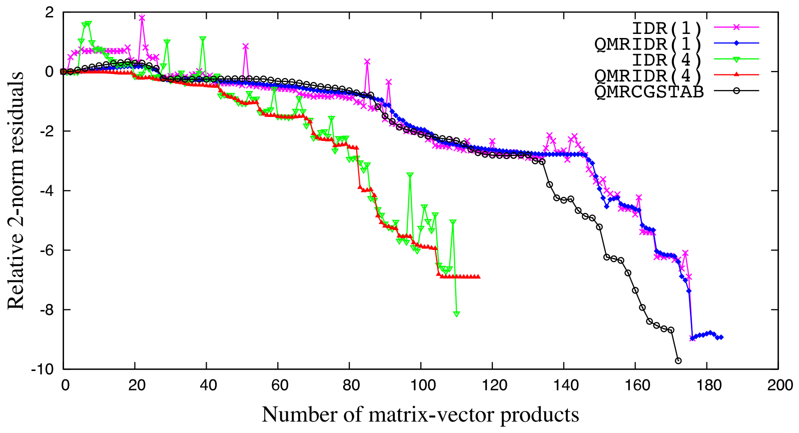

Fig. 3. Rdb1250.

4.2. *Experiments with preconditioning*

As preconditioning is essential for the successful use of iterative methods, we also check whether preconditioning is appropriate for our proposed method. The algorithm of preconditioned QMRIDR(s) can be easily obtained like for other IDR-type methods; see, e.g., [22]. Here, ILU(0) was used as the preconditioner. A bigger test matrix *Oberwolfach*/*chipcool*0 from a model reduction problem was used. The order of *Oberwolfach*/*chipcool*0 is 20,082, and the number of nonzero elements is 281,150.

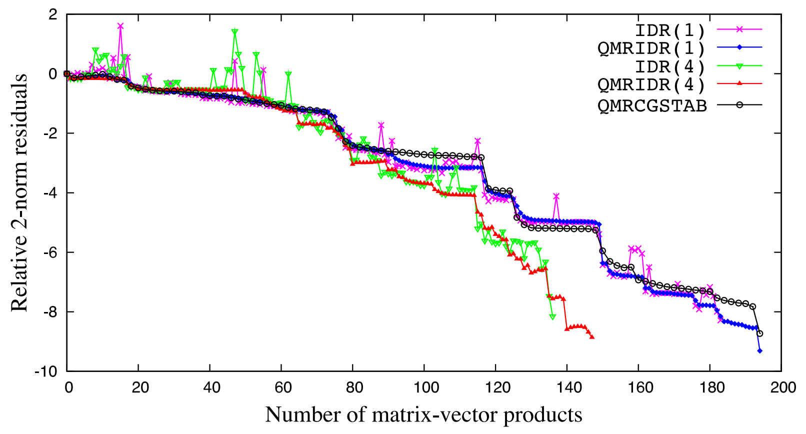

Fig. 4. Sherman4.

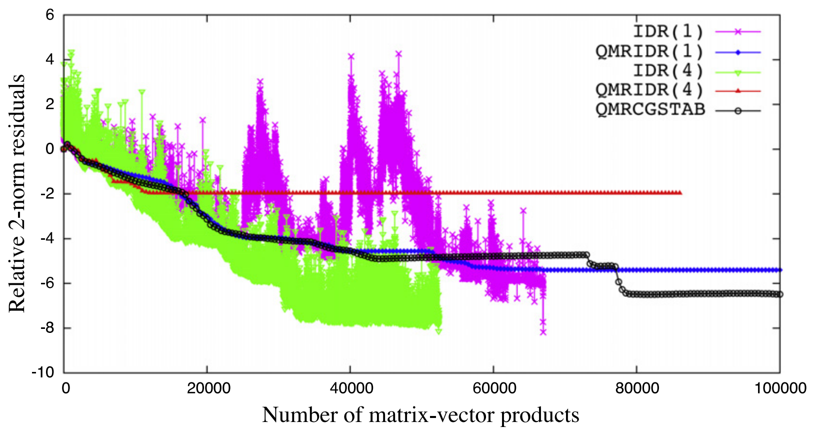

Fig. 5. Oberwolfach/chipcool0 without preconditioning.

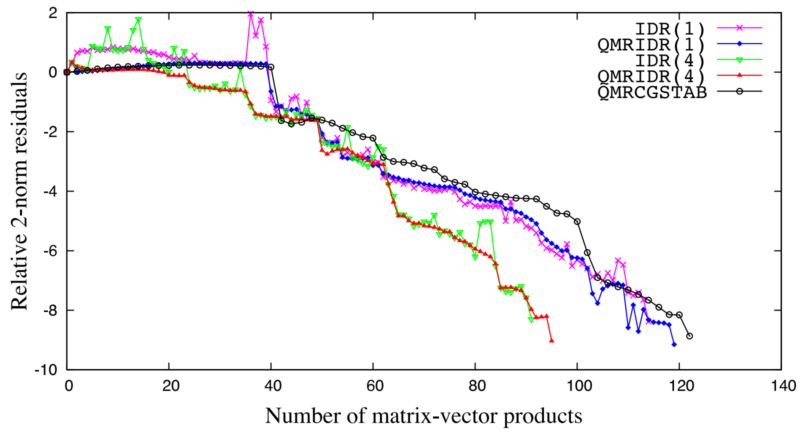

Fig. 6. Oberwolfach/chipcool0 with ILU(0) preconditioning.

We first applied methods to the test problem without preconditioning. From Fig. 5, we see that only IDR(1) and IDR(4) converged, and other QMR-type methods did not converge within 100,000 matrix–vector products. Even though the IDR(s) method could converge, it took more than 50,000 matrix–vector products for both IDR(1) and IDR(4). For QMR-type methods, the residual convergence histories are much smoother and all of them stagnated. QMRIDR(1) and QMRCGSTAB stagnated at a higher level than QMRIDR(4). Though QMRIDR(4) encountered the stagnation problem, like in Fig. 3, it reached the stopping criterion condition and terminated in less than 100,000 matrix–vector products.

Then, corresponding preconditioned methods were applied. From Fig. 6, we see that all methods converged quite quickly, and the preconditioned QMRIDR(s) also showed smooth behavior. Although preconditioned QMRIDR(s) took a few more iterations than preconditioned IDR(s) to reach the stopping criterion condition, it obtained a more accurate solution as in Section 4.1.

5. Conclusions

In this paper, we have proposed the QMRIDR(s) method for ameliorating the irregular convergence behavior of the IDR(s) method. To define our method, we reformulated the relations of residuals and their auxiliary vectors generated by the IDR(s) method in matrix form, and then constructed approximations based on a new basis by using the QMR strategy. Numerical experiments showed that the QMRIDR(s) method had smooth convergence behavior and converged almost as fast as the IDR(s) method. Furthermore, the QMRIDR(s) method with larger s converged faster than QMRCGSTAB. Even though IDR(1) is mathematically equivalent to Bi-CGSTAB, their corresponding QMR-type variants are mathematically different, as was also demonstrated by the numerical results.

Acknowledgments

The authors are grateful to the anonymous referees for their valuable and helpful comments which led to improvement of this paper. Du was supported in part by the China Scholarship Council, and Hori Sciences and Arts Foundation, Sogabe was partially supported by a Grant-in-Aid for Scientific Research under grant No. 21760058 and Zhang was funded in part by a Grant-in-Aid for Scientific Research under grant Nos. 19560065 and 22104004.

References

[1] M.R. Hestenes, E.L. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand. 49 (1952) 409–436.
[2] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Natl. Bur. Stand. 49 (1952) 33–53.
[3] R. Fletcher, Conjugate gradient methods for indefinite systems, in: Lecture Notes in Math., vol. 506, Springer-Verlag, Berlin, Heidelberg, New York, 1976, pp. 73–89.
[4] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[5] P. Sonneveld, CGS: a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 10 (1989) 36–52.
[6] 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. Stat. Comput. 13 (1992) 631–644.
[7] S.-L. Zhang, GPBi-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 18 (1997) 537–551.
[8] R.W. Freund, G.H. Golub, N.M. Nachtigal, Iterative solution of linear systems, Acta Numer. 1 (1992) 57–100.
[9] M.H. Gutknecht, Lanczos-type solvers for nonsymmetric linear systems of equations, Acta Numer. 6 (1997) 271–397.
[10] Y. Saad, H.A. van der Vorst, Iterative solution of linear systems in the 20th century, J. Comput. Appl. Math. 123 (2000) 1–33.
[11] V. Simoncini, D.B. Szyld, Recent computational developments in Krylov subspace methods for linear systems, Numer. Linear Algebra Appl. 14 (2007) 1–59.
[12] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia, PA, 2003.
[13] H.A. van der Vorst, Iterative Krylov Methods for Large Linear Systems, Cambridge University Press, Cambridge, New York, 2003.
[14] P. Wesseling, P. Sonneveld, Numerical experiments with a multiple grid and a preconditioned Lanczos type method, in: Lecture Notes in Mathematics, vol. 771, Springer-Verlag, Berlin, Heidelberg, New York, 1980, pp. 543–562.
[15] P. Sonneveld, 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) 1035–1062.
[16] 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.
[17] M.H. Gutknecht, IDR explained, Electron. Trans. Numer. Anal. 36 (2010) 126–148.
[18] V. Simoncini, D.B. Szyld, Interpreting IDR as a Petrov–Galerkin method, SIAM J. Sci. Comput. 32 (2010) 1898–1912.
[19] M.B. van Gijzen, P. Sonneveld, An elegant IDR(s) variant that efficiently exploits bi-orthogonality properties, Report 08-21, Department of Applied Mathematical Analysis, Delft University of Technology, 2008.
[20] G.L.G. Sleijpen, D.R. Fokkema, BiCGstab(ℓ) for linear equations involving unsymmetric matrices with complex spectrum, Electron. Trans. Numer. Anal. 1 (1993) 11–32.
[21] G.L.G. Sleijpen, M.B. van Gijzen, Exploiting BiCGstab(ℓ) strategies to induce dimension reduction, SIAM J. Sci. Comput. 32 (2010) 2687–2709.
[22] M. Tanio, M. Sugihara, GBi-CGSTAB(s, L): IDR(s) with higher-order stabilization polynomials, J. Comput. Appl. Math. 235 (2010) 765–784.
[23] L. Du, T. Sogabe, B. Yu, Y. Yamamoto, S.-L. Zhang, A block IDR(s) method for nonsymmetric linear systems with multiple right-hand sides, J. Comput. Appl. Math. 235 (2011) 4095–4106.
[24] M.H. Gutknecht, J.P.M. Zemke, Eigenvalue computations based on IDR, Research Report No. 2010-13, SAM, ETH Zurich & Bericht 145 INS, TUHH.
[25] T.F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, C. Tong, A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems, SIAM J. Sci. Comput. 15 (1994) 338–347.
[26] R.W. Freund, N.M. Nachtigal, QMR: a quasi-minimal residual method for non-Hermitian linear systems, Numer. Math. 60 (1991) 315–339.
[27] R.W. Freund, A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, SIAM J. Sci. Comput. 14 (1993) 470–482.
[28] Matrix Market. Available online at: http://math.nist.gov/MatrixMarket/.
[29] T.A. Davis, University of Florida Sparse Matrix Collection. Available online at: http://www.cise.ufl.edu/research/sparse/matrices.