Journal of Computational and Applied Mathematics 274 (2015) 35–43

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

IDR(s) for solving shifted nonsymmetric linear systems

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

[a] Faculty of Engineering, Information and Systems, University of Tsukuba, Tennodai 1-1-1, Tsukuba, 305-8573, Japan

[b] Graduate School of Information Science and Technology, Aichi Prefectural University, Nagakute-cho, Aichi-gun, Aichi, 480-1198, Japan

[c] Department of Computational Science and Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan

ARTICLE INFO

Article history:
Received 30 May 2012
Received in revised form 29 November 2013

Keywords:
Krylov subspace
Induced dimension reduction
IDR(s)
Shifted IDR(s)
Shifted linear systems

ABSTRACT

The IDR(s) method by Sonneveld and van Gijzen (2008) has recently received tremendous attention since it is effective for solving nonsymmetric linear systems. In this paper, we generalize this method to solve shifted nonsymmetric linear systems. When solving this kind of problem by existing shifted Krylov subspace methods, we know one just needs to generate one basis of the Krylov subspaces due to the shift-invariance property of Krylov subspaces. Thus the computation cost required by the basis generation of all shifted linear systems, in terms of matrix–vector products, can be reduced. For the IDR(s) method, we find that there also exists a shift-invariance property of the Sonneveld subspaces. This inspires us to develop a shifted version of the IDR(s) method for solving the shifted linear systems.

1. Introduction

We consider the solutions of shifted nonsymmetric linear systems of the form

$$(A + \sigma_i I)\boldsymbol{x}^{(i)} = \boldsymbol{b}, \quad i = 1, 2, \ldots, L, \tag{1}$$

where scalar shift parameter $\sigma_i \in \mathbb{R}$, I is the identity matrix, the matrix $A \in \mathbb{R}^{n\times n}$ is large sparse, nonsingular and the right-hand side vector $\boldsymbol{b} \in \mathbb{R}^n$. The shifted linear systems arise in a rich variety of applications such as higher-order implicit methods for solving time-dependent partial differential equations and quantum chromodynamics (QCD) (see [1] and references therein).

It is well known that Krylov subspace methods are widely used for the solution of linear systems. Denoting the k-dimensional Krylov subspace with respect to A and $\boldsymbol{b}$ by

$$\mathcal{K}_k(A, \boldsymbol{b}) := \text{span}\{\boldsymbol{b}, A\boldsymbol{b}, \ldots, A^{k-1}\boldsymbol{b}\}, \tag{2}$$

we can observe that the relation described below always holds for the shifted matrices in (1).

$$\mathcal{K}_k(A, \boldsymbol{b}) = \mathcal{K}_k(A + \sigma_i I, \boldsymbol{b}), \quad i = 1, 2, \ldots, L. \tag{3}$$

This implies that if we choose initial vectors $\boldsymbol{x}_0^{(i)}$ properly (for example, all initial guesses are zero), once a basis has been generated for one of these linear systems, it could also be used for all other linear systems. Therefore, there is no need to generate bases of all Krylov subspaces $\mathcal{K}_k(A+\sigma_i I, \boldsymbol{b})$ separately, and thus the computation cost required by the basis generation,

* Corresponding author.
E-mail addresses: du3stone@gmail.com, du@mma.cs.tsukuba.ac.jp, dulei@dlut.edu.cn (L. Du), sogabe@ist.aichi-pu.ac.jp (T. Sogabe), zhang@na.cse.nagoya-u.ac.jp (S.-L. Zhang).

http://dx.doi.org/10.1016/j.cam.2014.07.004
0377-0427/

in terms of matrix–vector products, can be reduced. This leads to shifted Krylov subspace methods for solving the shifted linear systems.

For shifted nonsymmetric (non-Hermitian) linear systems, methods such as the shifted BiCGStab(ℓ) [2], the shifted (TF)QMR [3], the restarted shifted FOM [4,5], and the shifted restarted GMRES [1,6,7] have been developed. Shifted algorithms of short-term recurrence methods of CG [8], CR [9], BiCG [10] and BiCGStab [11] were discussed in [12]. For the case of shifted (complex) symmetric linear systems, the shifted COCG and shifted COCR methods have been proposed in [13–15], respectively. The shifted QMR_SYM(B) method with the weighted quasi-residual minimization strategy was studied in [16].

Here, we consider the IDR(s) method [17], which is a generation of the IDR method [18]. This method has recently received tremendous attention since it is effective for solving nonsymmetric linear systems. Although the idea of the IDR(s) method is different from traditional Krylov subspace methods, there are the relations that IDR(1) is mathematically equivalent to BiCGStab and IDR(s) with $s > 1$ being related to the ML(k)BiCGSTAB method [19,20]. Since then IDR type methods have been intensively studied. For instance, relations between IDR and BiCGStab were discussed in [21]. More detailed IDR explanations have been made in [22–24]. Some variants of the IDR(s) method have been proposed in [22,25–30]. IDR has also been extended to the solution of some other problems; see, e.g., [31] for an eigenvalue solver.

A variant of the QMRIDR(s) method for solving shifted linear systems has recently been proposed by M.B. van Gijzen et al. [32]. The idea is closely related to the that of multi-shift QMR and multi-shift TFQMR. Their elegant work gave us a simple and natural question: Does there exist a variant of the original IDR(s) method for shifted linear systems? Our major contribution is to answer this question. The answer is yes, but it requires a different idea from that used in multi-shift QMRIDR(s). The idea is based on a shift-invariance property of the Sonneveld subspaces. As a result, a feature of the resulting algorithm is that the residual vectors for *seed* system and *add* system at the same iteration step lie in the same dimensional Sonneveld subspace.

The paper is organized as follows. In Section 2, we briefly review the IDR(s) method. In the next section, we discuss the shift-invariance property and then show how to generalize the IDR(s) method for solving the shifted linear systems. Implementation details will be described. In Section 4, numerical experiments are reported to show the effectiveness of our method. Finally, some concluding remarks are made in Section 5.

Throughout this paper, the transpose of vector $\boldsymbol{u}$ is denoted by $\boldsymbol{u}^{\mathrm{T}}$. $\|\boldsymbol{u}\|$ represents the Euclidean norm $\|\boldsymbol{u}\| = \sqrt{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}$. The kth column of matrix M is specified by "colon" notation $M(:, k)$. $\mathcal{N}(M)$ and $\mathcal{R}(M)$ denote the null space and the range of M, respectively. We will omit the subscript whenever it is clear from the context.

2. The IDR(s) method

In this section, we briefly review the IDR(s) method. For more detailed discussions, one may refer to [17].

Let $\boldsymbol{x}_0$ be an initial guess of the standard linear systems $A\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{r}_0$ be the corresponding residual vector. Define $\mathcal{G}_0 := \mathcal{K}_v(A, \boldsymbol{r}_0)$ with v being the grade of $\boldsymbol{r}_0$ with respect to A, and S be a subspace of $\mathbb{R}^n$. A sequence of subspaces $\mathcal{G}_j$ (also called Sonneveld subspace) is recursively defined as

$$\mathcal{G}_j := (I - \omega_j A)(\mathcal{G}_{j-1} \cap S), \quad j = 1, 2, \ldots, \tag{4}$$

where parameters ω_j are nonzero. Then, the IDR theorem [17,18] can be described as follows.

Theorem 2.1. *Under the assumption that subspace $\mathcal{G}_0 \cap S$ does not contain a nontrivial invariant subspace of A, the following results hold:*

1. $\mathcal{G}_j \subset \mathcal{G}_{j-1}$, $j = 1, 2, \ldots$.
2. $\exists j_0 \leq n$, *for all* $j \geq j_0$ $\mathcal{G}_j = \{0\}$.

Theorem 2.1 shows that the Sonneveld subspaces $\mathcal{G}_j$ can be finite until $\mathcal{G}_j = \{\boldsymbol{0}\}$. Based on this theorem, the IDR(s) method was developed. Assuming that $s + 1$ residual vectors $\boldsymbol{r}_{k-s}, \ldots, \boldsymbol{r}_k$ in $\mathcal{G}_{j-1}$ are known, the idea of the IDR(s) method is to construct the following $s + 1$ residual vectors $\boldsymbol{r}_{k+1}, \ldots, \boldsymbol{r}_{k+s+1}$ in $\mathcal{G}_j$. The construction process continues with proper choice of the parameter ω_j, and approximations $\boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_{k+s+1}$ can be extracted directly. Finally, when a residual vector is constructed in $\{\boldsymbol{0}\}$, the exact solution will be reached. We briefly describe the derivation process of the IDR(s) method below.

Let $P = [p_1, p_2, \ldots, p_s] \in \mathbb{R}^{n\times s}$. Subspace S can be chosen as the null space of P^{T}, i.e., $S = \mathcal{N}(P^{\mathrm{T}})$. It was suggested to orthogonalize a set of random vectors for P in [17].

Denote $\Delta\boldsymbol{r}_i := \boldsymbol{r}_{i+1} - \boldsymbol{r}_i$, an auxiliary vector $\boldsymbol{v}_k$ is defined as

$$\boldsymbol{v}_k = \boldsymbol{r}_k - \sum_{l=1}^{s} \gamma_l \Delta\boldsymbol{r}_{k-l} \tag{5}$$

$$= (1 - \gamma_1)\boldsymbol{r}_k + \sum_{l=1}^{s-1} (\gamma_l - \gamma_{l+1})\boldsymbol{r}_{k-l} + \gamma_s \boldsymbol{r}_{k-s}, \tag{6}$$

in which γ_l's are parameters. It is obvious that $\boldsymbol{v}_k \in \mathcal{G}_{j-1}$ for any γ_l's as we have assumed these $s + 1$ residual vectors $\boldsymbol{r}_{k-s}, \ldots, \boldsymbol{r}_k$ in $\mathcal{G}_{j-1}$ above. Moreover, $\boldsymbol{v}_k$ is also limited in S by setting parameters γ_l's in the IDR(s) method. From the

definition of subspace S, parameters γ_l's can be determined per iteration step by solving the $s \times s$ linear system (derived from $P^{\mathrm{T}}\boldsymbol{v}_k = 0$),

$$P^{\mathrm{T}}[\Delta\boldsymbol{r}_{k-1}, \Delta\boldsymbol{r}_{k-2}, \ldots, \Delta\boldsymbol{r}_{k-s}] \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_s \end{bmatrix} = P^{\mathrm{T}}\boldsymbol{r}_k. \tag{7}$$

Then, a new residual vector $\boldsymbol{r}_{k+1}$ can be generated using $\boldsymbol{v}_k$ as:

$$\boldsymbol{r}_{k+1} = (I - \omega_j A)\boldsymbol{v}_k. \tag{8}$$

From the definition of subspace $\mathcal{G}_j$, we see that $\boldsymbol{r}_{k+1} \in \mathcal{G}_j$. We also should determine the value of parameter ω_j, one choice is to minimize the 2-norm of $\boldsymbol{r}_{k+1}$, which gives $\omega_j = (A\boldsymbol{v}_k)^{\mathrm{T}}\boldsymbol{v}_k / \|A\boldsymbol{v}_k\|^2$.

Denote $\Delta\boldsymbol{x}_i := \boldsymbol{x}_{i+1} - \boldsymbol{x}_i$. Now, the corresponding approximation $\boldsymbol{x}_{k+1}$ can be derived by relations $A\Delta\boldsymbol{x}_i = -\Delta\boldsymbol{r}_i$, (5) and (8) as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \omega_j\boldsymbol{v}_k - \sum_{l=1}^{s} \gamma_l \Delta\boldsymbol{x}_{k-l}. \tag{9}$$

In the following s iterations, each foremost residual vector will be replaced by the new one and the above process can be cycled to construct the other s intermediate residual vectors $\boldsymbol{r}_{k+2}, \ldots, \boldsymbol{r}_{k+s+1}$. In consideration of the computation cost, parameter ω_j may keep 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. This process continues until a satisfactory approximation is reached. We end this section by recalling the IDR(s) algorithm [17] in Algorithm 1.

Algorithm 1 IDR(s)

Let $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $P \in \mathbb{R}^{n\times s}$, $j = 1$;
% The initial step
for $i = 1, 2, \ldots, s$ **do**
  $\boldsymbol{v} = A\boldsymbol{r}_{i-1}$, $\omega = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{r}_{i-1}/\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}$;
  $\Delta X(:, i) = \omega\boldsymbol{r}_{i-1}$, $\Delta R(:, i) = -\omega\boldsymbol{v}$;
  $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \Delta X(:, i)$, $\boldsymbol{r}_i = \boldsymbol{r}_{i-1} + \Delta R(:, i)$;
end for
$M = P^{\mathrm{T}}\Delta R$; $\boldsymbol{h} = P^{\mathrm{T}}\boldsymbol{r}_i$;
% The IDR step
while stopping criterion is not satisfied **do**
  for $k = 0, 1, \ldots, s$ **do**
    Solve $\boldsymbol{c}$ from $M\boldsymbol{c} = \boldsymbol{h}$;
    $\boldsymbol{q} = -\Delta R\boldsymbol{c}$, $\boldsymbol{v} = \boldsymbol{r}_i + \boldsymbol{q}$;
    if $k == 0$ **then**
      $\boldsymbol{t} = A\boldsymbol{v}$; $\omega = \boldsymbol{t}^{\mathrm{T}}\boldsymbol{v}/\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t}$;
      $\Delta R(:, j) = \boldsymbol{q} - \omega\boldsymbol{t}$, $\Delta X(:, j) = -\Delta X\boldsymbol{c} + \omega\boldsymbol{v}$;
    else
      $\Delta X(:, j) = -\Delta X\boldsymbol{c} + \omega\boldsymbol{v}$, $\Delta R(:, j) = -A\Delta X(:, j)$;
    end if
    $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \Delta R(:, j)$, $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta X(:, j)$;
    $\delta\boldsymbol{m} = P^{\mathrm{T}}\Delta R(:, j)$, $M(:, j) = \delta\boldsymbol{m}$, $\boldsymbol{h} = \boldsymbol{h} + \delta\boldsymbol{m}$;
    $i = i + 1, j = j + 1, j = \mathrm{mod}(j - 1, s) + 1$;
  end for
end while

3. A shifted IDR(s) method

In this section, we generalize the IDR(s) method and propose a shifted IDR(s) method to solve the shifted linear systems. Detailed implementations of the shifted IDR(s) method will be discussed. An observation is that the shifted IDR(s) method has the feature that all linear systems are solved by the IDR(s) method with different implementations. More precisely, the shifted IDR(s) method solves one of the linear systems (called *seed* system) as the Algorithm 1. All other linear systems (called *add* system) will be solved by the IDR(s) method that adopts different strategies to determine parameter ω_j for solving the add system.

We know that there exists the shift-invariance property of Krylov subspaces for the shifted linear systems. Similar result can be obtained for the Sonneveld subspaces in Theorem 2.1. Under some suitable conditions, we easily have the following corollary from the IDR theorem:

Corollary 3.1. *Let $\boldsymbol{r}_0^{(i)}$ $(i = 1, \ldots, m)$ be collinear to each other, $\mathcal{G}_0^{(i)} = \mathcal{K}_n(A + \sigma_i I, \boldsymbol{r}_0^{(i)})$, S be a subspace of $\mathbb{R}^n$, and define sequences of subspaces $\mathcal{G}_j^{(i)}$ as*

$$\mathcal{G}_j^{(i)} = \left(I - \omega_j^{(i)}(A + \sigma_i I)\right)(\mathcal{G}_{j-1}^{(i)} \cap S), \quad \omega_j^{(i)} \neq 0,\ j = 1, 2, \ldots$$

then it holds that $\mathcal{G}_j^{(1)} = \mathcal{G}_j^{(2)} = \cdots = \mathcal{G}_j^{(m)}$.

As discussed in Section 2, the IDR(s) method tries to construct residual vectors in those Sonneveld subspaces. By the shift-invariance property of Corollary 3.1, we can construct residual vectors of all shifted linear systems simultaneously using only one matrix–vector product per iteration step like other existing shifted methods.

In addition to the shift-invariance property of Krylov subspaces, residual collinearity is the other key ingredient to the implementations of all existing shifted methods. For traditional Krylov subspace methods with the (*Petrov*)–*Galerkin* condition, there exists the following theorem. For simplicity of expression, we denote the *seed* system and the *add* system by $A\boldsymbol{x} = \boldsymbol{b}$ and $(A + \sigma I)\boldsymbol{x}^\sigma = \boldsymbol{b}$, respectively.

Theorem 3.2 ([2]). *Let $\mathcal{W}_1 \subset \mathcal{W}_2 \subset \cdots \subset \mathcal{W}_j$ be a sequence of nested subspaces of $\mathbb{C}^n$ such that $\mathcal{W}_i$ has dimension i and $\mathcal{W}_i \cap (\mathcal{K}_{i+1}(A, \boldsymbol{b}))^\perp = \{\boldsymbol{0}\}$, $i = 1, \ldots, j$. If residual vectors $\boldsymbol{r}_i := R_i(A)\boldsymbol{b}$ and $\boldsymbol{r}_i^\sigma := R_i^\sigma(A + \sigma I)\boldsymbol{b}$ satisfy*

$$\boldsymbol{r}_i, \boldsymbol{r}_i^\sigma \perp \mathcal{W}_i, \quad i = 1, \ldots, j, \tag{10}$$

where $R_i(\lambda)$ and $R_i^\sigma(\lambda)$ are residual polynomials of degree at most i. Then $\boldsymbol{r}_i$ and $\boldsymbol{r}_i^\sigma$ are collinear, i.e., there exists $\pi_i^\sigma \in \mathbb{C}$ such that $\boldsymbol{r}_i = \pi_i^\sigma \boldsymbol{r}_i^\sigma$.

From this theorem, it is known that when we apply Krylov subspace methods, such as CG, and Bi-CG, to solve (1) separately, their residual vectors will be collinear at the same number of iteration steps. For most of other Krylov methods such as BiCGStab(ℓ) [33] and GMRES [34], residual vectors of the *seed* system and *add* system will not be collinear any more. To reduce the computation cost and without additional matrix–vector products for the *add* system, the *add* system can be solved by using the assumed condition that residual vectors of the *seed* system and *add* system remain collinear, which gives the shifted BiCGStab(ℓ) method and the restarted shifted GMRES method, respectively. Although we should face up to the fact that methods applied to the *seed* system are different from that applied to the *add* system, their variant methods have been shown to be effective for solving the *add* system.

When applying Algorithm 1 to both the *seed* and *add* systems, we also meet the problem that their residual vectors will not be collinear either. Motivated by other shifted methods such as the shifted BiCGStab(ℓ), we also employ the collinear condition and construct residual vectors for the *add* system, which will give our shifted version of the IDR(s) method for solving the shifted linear systems.

Proposition 3.3. *Start with $\boldsymbol{r}_0 = \pi_0^\sigma \boldsymbol{r}_0^\sigma$, then there exist $\omega^\sigma \in \mathbb{R}$ and $\gamma_l^\sigma \in \mathbb{R}$ $(l = 1, 2, \ldots, s)$ per iteration step such that $\boldsymbol{r}_{k+1}^\sigma$ is collinear with residual vector $\boldsymbol{r}_{k+1}$ generated by Algorithm 1 for $k = 0, 1, \ldots$.*

Proof. See the computation steps below. □

Now we discuss how to compute $\boldsymbol{r}_{k+1}^\sigma$ to be collinear with $\boldsymbol{r}_{k+1}$ when $\boldsymbol{r}_i = \pi_i^\sigma \boldsymbol{r}_i^\sigma$ $(\pi_i^\sigma \in \mathbb{R},\ i = 0, 1, \ldots, k)$. In the initial step of Algorithm 1, residual vectors of *seed* system are updated as

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \omega_k A\boldsymbol{r}_k \quad (k = 0, \ldots, s-1). \tag{11}$$

We construct residual vectors of the *add* system in the form of (11), i.e.,

$$\begin{aligned} \boldsymbol{r}_{k+1}^\sigma &= \boldsymbol{r}_k^\sigma - \omega_k^\sigma (A + \sigma I)\boldsymbol{r}_k^\sigma \\ &= (1 - \sigma\omega_k^\sigma)\boldsymbol{r}_k^\sigma - \omega_k^\sigma A\boldsymbol{r}_k^\sigma, \end{aligned} \tag{12}$$

where unknown parameter $\omega_k^\sigma \in \mathbb{R}$. In order to make $\boldsymbol{r}_{k+1} = \pi_{k+1}^\sigma \boldsymbol{r}_{k+1}^\sigma$, substituting previous relations $\boldsymbol{r}_i = \pi_i^\sigma \boldsymbol{r}_i^\sigma$ $(i = 0, \ldots, k)$ into (11), we can rewrite (11) in the form of

$$\boldsymbol{r}_{k+1}^\sigma = \frac{\pi_k^\sigma}{\pi_{k+1}^\sigma}\boldsymbol{r}_k^\sigma - \omega_k \frac{\pi_k^\sigma}{\pi_{k+1}^\sigma} A\boldsymbol{r}_k^\sigma. \tag{13}$$

Meanwhile, by relations $\boldsymbol{r}_i = \pi_i^\sigma \boldsymbol{r}_i^\sigma$, (11) and residual polynomial $R_i(0) = 1$ (that $\boldsymbol{r}_i = R_i(A)\boldsymbol{r}_0$), we have

$$\begin{cases} \pi_k^\sigma = R_k(-\sigma) \\ \pi_{k+1}^\sigma = \pi_k^\sigma + \omega_k \sigma \pi_k^\sigma. \end{cases}$$

Parameter π_{k+1}^σ in (13) can be calculated when π_k^σ is known. Now, comparing the coefficients of (12) and (13), we can directly obtain $\omega_k^\sigma = \omega_k \frac{\pi_k^\sigma}{\pi_{k+1}^\sigma}$. The corresponding approximation of the *add* system can be computed as

$$\boldsymbol{x}_{k+1}^\sigma = \boldsymbol{x}_k^\sigma + \omega_k^\sigma \boldsymbol{r}_k^\sigma. \tag{14}$$

Next, we show the derivation process of $\boldsymbol{r}_{k+1}^{\sigma}$ by $\boldsymbol{r}_{k+1}$ in the IDR step of Algorithm 1. From relations (5) and (8), we have

$$\boldsymbol{r}_{k+1} = (I - \omega_j A)\Big((1-\gamma_1)\boldsymbol{r}_k + \sum_{l=1}^{s-1}(\gamma_l - \gamma_{l+1})\boldsymbol{r}_{k-l} + \gamma_s \boldsymbol{r}_{k-s}\Big). \tag{15}$$

Similar to the construction of initial residual vectors, we define residual vectors of the *add* system in the same form of (15), i.e.,

$$\begin{aligned}\boldsymbol{r}_{k+1}^{\sigma} &= \Big(I - \omega_j^{\sigma}(A+\sigma I)\Big)\Big((1-\gamma_1^{\sigma})\boldsymbol{r}_k^{\sigma} + \sum_{l=1}^{s-1}(\gamma_l^{\sigma} - \gamma_{l+1}^{\sigma})\boldsymbol{r}_{k-l}^{\sigma} + \gamma_s^{\sigma}\boldsymbol{r}_{k-s}^{\sigma}\Big)\\ &= \Big(I - \frac{\omega_j^{\sigma}}{1-\sigma\omega_j^{\sigma}}A\Big)(1-\sigma\omega_j^{\sigma})\Big((1-\gamma_1^{\sigma})\boldsymbol{r}_k^{\sigma} + \sum_{l=1}^{s-1}(\gamma_l^{\sigma} - \gamma_{l+1}^{\sigma})\boldsymbol{r}_{k-l}^{\sigma} + \gamma_s^{\sigma}\boldsymbol{r}_{k-s}^{\sigma}\Big),\end{aligned} \tag{16}$$

where unknown parameters $\omega_j^{\sigma}, \gamma_1^{\sigma}, \ldots, \gamma_s^{\sigma} \in \mathbb{R}$. Substituting $\boldsymbol{r}_i = \pi_i^{\sigma}\boldsymbol{r}_i^{\sigma}$ $(i = 0, 1, \ldots, k+1)$ into (15), we can rewrite (15) in the form of

$$\boldsymbol{r}_{k+1}^{\sigma} = (I - \omega_j A)\Big((1-\gamma_1)\frac{\pi_k^{\sigma}}{\pi_{k+1}^{\sigma}}\boldsymbol{r}_k^{\sigma} + \sum_{l=1}^{s-1}(\gamma_l - \gamma_{l+1})\frac{\pi_{k-l}^{\sigma}}{\pi_{k+1}^{\sigma}}\boldsymbol{r}_{k-l}^{\sigma} + \gamma_s\frac{\pi_{k-s}^{\sigma}}{\pi_{k+1}^{\sigma}}\boldsymbol{r}_{k-s}^{\sigma}\Big). \tag{17}$$

Together with relations (15), $\boldsymbol{r}_i = \pi_i^{\sigma}\boldsymbol{r}_i^{\sigma}$ $(i = 0, 1, \ldots, k+1)$ and $R_i(0) = 1$, we can compute π_{k+1}^{σ} as follows:

$$\pi_{k+1}^{\sigma} = (1+\omega_j\sigma)\Big((1-\gamma_1)\pi_k^{\sigma} + \sum_{l=1}^{s-1}(\gamma_l - \gamma_{l+1})\pi_{k-l}^{\sigma} + \gamma_s\pi_{k-s}^{\sigma}\Big). \tag{18}$$

Comparing the corresponding coefficients of (16) and (17), we can obtain

$$\begin{cases}\dfrac{\omega_j^{\sigma}}{1-\sigma\omega_j^{\sigma}} = \omega_j\\ 1-\gamma_1^{\sigma} = \dfrac{1-\gamma_1}{1-\sigma\omega_j^{\sigma}}\cdot\dfrac{\pi_k^{\sigma}}{\pi_{k+1}^{\sigma}}\\ \gamma_l^{\sigma} - \gamma_{l+1}^{\sigma} = \dfrac{\gamma_l - \gamma_{l+1}}{1-\sigma\omega_j^{\sigma}}\cdot\dfrac{\pi_{k-l}^{\sigma}}{\pi_{k+1}^{\sigma}}, \quad l = 1, 2, \ldots, s-1\\ \gamma_s^{\sigma} = \dfrac{\gamma_s}{1-\sigma\omega_j^{\sigma}}\cdot\dfrac{\pi_{k-s}^{\sigma}}{\pi_{k+1}^{\sigma}},\end{cases} \tag{19}$$

from which unknowns $\omega_k^{\sigma}, \gamma_1^{\sigma}, \ldots, \gamma_s^{\sigma}$ can be easily determined as

$$\begin{cases}\omega_j^{\sigma} = \dfrac{\omega_j}{1+\sigma\omega_j}\\ \gamma_1^{\sigma} = 1 - (1-\gamma_1)(1+\sigma\omega_j)\pi_k^{\sigma}/\pi_{k+1}^{\sigma}\\ \gamma_{l+1}^{\sigma} = \gamma_l^{\sigma} - (\gamma_l - \gamma_{l+1})(1+\sigma\omega_j)\pi_{k-l}^{\sigma}/\pi_{k+1}^{\sigma}, \quad l = 1, 2, \ldots, s-1.\end{cases} \tag{20}$$

Now, approximation $\boldsymbol{x}_{k+1}^{\sigma}$ can be directly derived from (16) as

$$\boldsymbol{x}_{k+1}^{\sigma} = \boldsymbol{x}_k^{\sigma} + \omega_j^{\sigma}\boldsymbol{v}_k^{\sigma} - \sum_{l=1}^{s}\gamma_l^{\sigma}\Delta\boldsymbol{x}_{k-l}^{\sigma}, \tag{21}$$

where $\boldsymbol{v}_k^{\sigma} = \boldsymbol{r}_k^{\sigma} - \sum_{l=1}^{s}\gamma_l^{\sigma}\Delta\boldsymbol{r}_{k-l}^{\sigma}$. In practical computation, we can express $\boldsymbol{v}_k^{\sigma}$ by using the residual vectors $\boldsymbol{r}_{k-s}, \ldots, \boldsymbol{r}_k$ of the *seed* system.

We summarize all computation steps above and give the resulting algorithm in Algorithm 2, which is referred to as shifted IDR(s). Note that before using Algorithm 2, a *seed* system should be given. Since it is hard to make an optimal choice of the *seed* system in advance, the seed switching technique described in [13] may be an option.

We develop Algorithm 2 based on the residual collinearity and we can see the *seed* system is solved by the IDR(s) method (same as Algorithm 1) in Algorithm 2, but where implementations for the *add* system are not equivalent to that of *seed* system. Through the computation steps, we also see that implementations for both the *seed* system and the *add* system are under the same framework of the IDR theorem. In other words, all linear systems are solved by the IDR(s) method, but with different strategies for determining parameter ω_j and ω_j^{σ}.

Algorithm 2 Shifted IDR(s) $(A+\sigma_f I)\boldsymbol{x}=\boldsymbol{b}$ as *seed* system

Let $\boldsymbol{x}_0^{(i)}=0$; $\boldsymbol{r}_0=\boldsymbol{b}$; $P\in\mathbb{R}^{n\times s}$; $\pi_0^{(f,i)}=1$;
for $k=0,1,\ldots,s-1$ **do**
  $\boldsymbol{v}=(A+\sigma_f I)\boldsymbol{r}_k$; $\omega=\boldsymbol{v}^{\mathrm{T}}\boldsymbol{r}_k/\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}$;
  $\Delta X(:,k+1)=\omega\boldsymbol{r}_k$; $\Delta R(:,k+1)=-\omega\boldsymbol{v}$;
  $\boldsymbol{x}_{k+1}^{(f)}=\boldsymbol{x}_k^{(f)}+\Delta X(:,k+1)$; $\boldsymbol{r}_{k+1}=\boldsymbol{r}_k+\Delta R(:,k+1)$;
  % Iteration for *add* system
  for $i(\neq f)=1,2,\ldots,m$ **do**
    if $\|\boldsymbol{r}_k^{(i)}\|>\varepsilon\|\boldsymbol{b}\|$ **then**
      $\pi_{k+1}^{(f,i)}=\pi_k^{(f,i)}+\omega(\sigma_i-\sigma_f)\pi_k^{(f,i)}$; $\omega^{(f,i)}=\frac{\omega}{1+\omega(\sigma_i-\sigma_f)}$;
      $\boldsymbol{x}_{k+1}^{(i)}=\boldsymbol{x}_k^{(i)}+\frac{\omega^{(f,i)}}{\pi_k^{(f,i)}}\boldsymbol{r}_k$;
    end if
  end for
end for
$j=1$; $k=s$; $M=P^{\mathrm{T}}\Delta R$; $\boldsymbol{h}=P^{\mathrm{T}}\boldsymbol{r}_k$;
while stopping criterion is not satisfied **do**
  for $l=0,1,\ldots,s$ **do**
    Solve c from $M\boldsymbol{c}=\boldsymbol{h}$;
    $\boldsymbol{q}=-\Delta R\boldsymbol{c}$; $\boldsymbol{v}=\boldsymbol{r}_k+\boldsymbol{q}$;
    if $l==0$ **then**
      $\boldsymbol{t}=(A+\sigma_f I)\boldsymbol{v}$; $\omega=\boldsymbol{t}^{\mathrm{T}}\boldsymbol{v}/\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t}$;
      $\Delta R(:,j)=\boldsymbol{q}-\omega\boldsymbol{t}$; $\Delta X(:,j)=-\Delta X\boldsymbol{c}+\omega\boldsymbol{v}$;
    else
      $\Delta X(:,j)=-\Delta X\boldsymbol{c}+\omega\boldsymbol{v}$; $\Delta R(:,j)=-(A+\sigma_f I)\Delta X(:,j)$;
    end if
    $\boldsymbol{r}_{k+1}=\boldsymbol{r}_k+\Delta R(:,j)$; $\boldsymbol{x}_{k+1}^{(f)}=\boldsymbol{x}_k^{(f)}+\Delta X(:,j)$;
    $\delta\boldsymbol{m}=P^{\mathrm{T}}\Delta R(:,j)$; $M(:,j)=\delta\boldsymbol{m}$; $\boldsymbol{h}=\boldsymbol{h}+\delta\boldsymbol{m}$;
    % Iteration for *add* system
    $\gamma_1=\boldsymbol{c}_{j-1}$, $\gamma_2=\boldsymbol{c}_{j-2}$, $\ldots$, $\gamma_{j-1}=\boldsymbol{c}_1$;
    $\gamma_j=\boldsymbol{c}_s$, $\gamma_{j+2}=\boldsymbol{c}_{s-1}$, $\ldots$, $\gamma_s=\boldsymbol{c}_j$;
    for $i(\neq f)=1,2,\ldots,m$ **do**
      if $\|\boldsymbol{r}_k^{(i)}\|>\varepsilon\|\boldsymbol{b}\|$ **then**
        $\alpha_i=1+\omega(\sigma_i-\sigma_f)$;
        $\pi_{k+1}^{(f,i)}=\alpha_i\Big((1-\gamma_1)\pi_k^{(f,i)}+\sum_{g=1}^{s-1}(\gamma_g-\gamma_{g+1})\pi_{k-g}^{(f,i)}+\gamma_s\pi_{k-s}^{(f,i)}\Big)$;
        $\gamma_1^{(f,i)}=1-\alpha_i(1-\gamma_1)\pi_k^{(f,i)}/\pi_{k+1}^{(f,i)}$;
        $\gamma_{g+1}^{(f,i)}=\gamma_g^{(f,i)}-(\gamma_g-\gamma_{g+1})\alpha_i\pi_{k-g}^{(f,i)}/\pi_{k+1}^{(f,i)}$ $(g=1,\ldots,s-1)$;
        $\boldsymbol{x}_{k+1}^{(i)}=\boldsymbol{x}_k^{(i)}+\omega\boldsymbol{v}/\pi_{k+1}^{(f,i)}-\sum_{g=1}^{s}\gamma_g^{(f,i)}(\boldsymbol{x}_{k+1-g}^{(i)}-\boldsymbol{x}_{k-g}^{(i)})$;
      end if
    end for
    $k=k+1$; $j=j+1$; $j=\mathrm{mod}(j-1,s)+1$;
  end for
end while

4. Numerical experiments

In this section, we report some results of numerical experiments. We compare our proposed method with the IDR(s) method, the shifted restarted GMRES method and the shifted BiCGStab(ℓ) method, which are abbreviated to IDR(s), sIDR(s), sGMRES(m) and sBiCGStab(ℓ) hereafter, respectively.

All the numerical experiments were performed on MacBook Air with 1.7 GHz processor and 4 GB of RAM. Algorithms were implemented in MATLAB 2013a with double precision arithmetic. The vector of right-hand side $\boldsymbol{b}=[1,\ldots,1]^{\mathrm{T}}$ and matrix $P=rand(n,s)$, where the *rand* function generated a random matrix with the size of $n\times s$. Elements of the random matrix are distributed in (0, 1). In all these experiments, the initial guesses were taken as $\boldsymbol{x}_0^{(i)}=\boldsymbol{0}$ and stopping criterion was either $\|\boldsymbol{r}_k^{(i)}\|/\|\boldsymbol{b}\|<10^{-8}$ for $i=0,\ldots,L$ or when this condition of the relative residual was not satisfied within n iterations for all linear systems (denoted by ‡). We note that it is not necessary to explicitly compute the residual of *add*

Table 1
Sizes (n) and numbers of nonzero elements (nnz) of test matrices.

Matrix	n	nnz	Application discipline
add32	4960	23 884	Electronic circuit design
cavity05	1182	32 747	Finite element modeling
fidap037	3565	67 591	Finite element modeling
pde2961	2961	14 585	Partial differential equations
sherman4	1104	3786	Oil reservoir modeling
wang1	2903	19 093	Semiconductor device problem

Table 2
Number of matrix–vector products and computation time in seconds (in parentheses).

	IDR(4)	sIDR(4)	sGMRES(16)	sBiCGStab(4)
add32	3832 (2.04)	100 (0.40)	119 (0.14)	104 (0.43)
cavity05	27 369 (10.95)	695 (1.46)	‡	1136 (1.32)
fidap037	8553 (11.08)	85 (0.78)	153 (0.33)	96 (0.68)
pde2961	24 359 (9.16)	247 (1.93)	425 (0.45)	320 (1.99)
sherman4	13 712 (3.05)	140 (0.62)	969 (0.47)	176 (0.55)
wang1	21 735 (10.62)	423 (1.75)	‡	640 (2.00)

system due to the collinearity. Meanwhile, the numbers of matrix–vector products of the IDR-type methods, sGMRES(m) and sBiCGStab(ℓ) per iteration are different. $s = 4$, $m = 16$ and $\ell = 4$ were set for both the IDR-type methods, sGMRES(m) and sBiCGStab(ℓ), respectively.

4.1. Experiment 1

All test matrices in this experiment are from the Matrix Market collection [35] and the University of Florida sparse matrix collection [36]. Properties of test matrices (matrix size and number of nonzero elements etc.) are listed in Table 1. Here, the values of shift parameter are $\sigma_i = 0.0001 * (i - 1),\ i = 1, \ldots, 100$. The first linear system $(A + \sigma_1 I)\boldsymbol{x}^{(1)} = \boldsymbol{b}$ was used as the *seed* system.

The numerical results shown in Table 2 are the number of matrix–vector products and the computation time in seconds (in parentheses) for IDR(s), sIDR(s), sGMRES(m) and sBiCGStab(ℓ). From Table 2, some observations are made as follows.

- Methods except sGMRES(16) solved all linear systems within the maximum iterations.
- Of all the methods, sIDR(4) needed the least matrix–vector products. For all test problems that could be solved, sGMRES(16) took the minimal computation time even with more matrix–vector products. This is because there are other two kinds of operations, inner products and vector updates, besides the matrix–vector products in all these methods. The computation time taken by the inner products and vector updates corresponding to one matrix–vector product of sIDR(4) and sBiCGStab(4) is more than that of sGMRES(16). We also see that sIDR(4) is competitive with sBiCGStab(4).

4.2. Experiment 2

In this experiment, we consider two examples used in other literature. The test matrix A of example 1 used in [4,37,5] is upper bidiagonal with the size of 100. The diagonal vector is given as [0.001, 0.002, 0.003, 0.004, 10, 11, ..., 105] and all superdiagonal elements are 1. Here we chose $\sigma_1 = 0.5$ and $\sigma_2 = 1.0$ for the *seed* system and the *add* system, respectively.

The second example used in [4,38,39] is a matrix defined as

$$A = \begin{bmatrix}
1 & 0.21 & 1.2 & 0 & 0.13 & 1.42 & & & & \\
0.45 & 2 & 0.21 & 1.2 & 0 & 0.13 & 1.42 & & & \\
0 & 0.45 & 3 & 0.21 & 1.2 & 0 & \ddots & \ddots & & \\
0.12 & 0 & 0.45 & 4 & 0.21 & \ddots & \ddots & \ddots & \ddots & \\
0.11 & 0.12 & 0 & 0.45 & \ddots & \ddots & \ddots & \ddots & 0.13 & 1.42 \\
 & 0.11 & 0.12 & \ddots & \ddots & \ddots & \ddots & 1.2 & 0 & 0.13 \\
 & & \ddots & \ddots & \ddots & \ddots & 197 & 0.21 & 1.2 & 0 \\
 & & & \ddots & \ddots & 0 & 0.45 & 198 & 0.21 & 1.2 \\
 & & & & 0.11 & 0.12 & 0 & 0.45 & 199 & 0.21 \\
 & & & & & 0.11 & 0.12 & 0 & 0.45 & 200
\end{bmatrix}. \tag{22}$$

For this example, we chose $\sigma_1 = -0.5$ and $\sigma_2 = 0.5$ for the *seed* system and the *add* system, respectively.

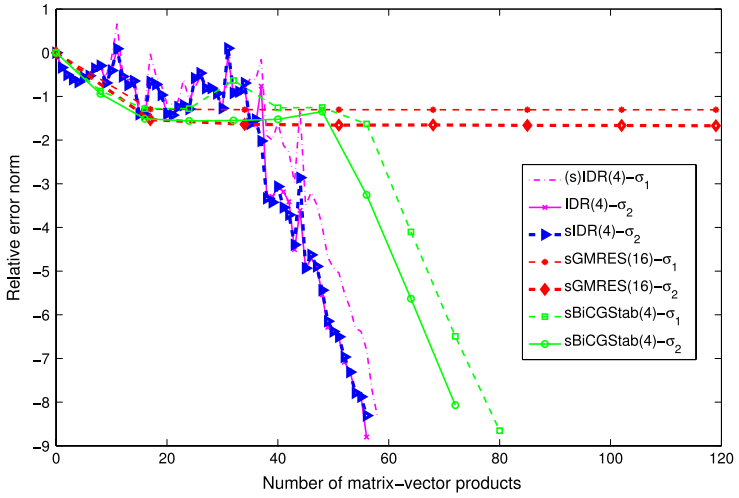

Fig. 1. Relative residual histories for example 1.

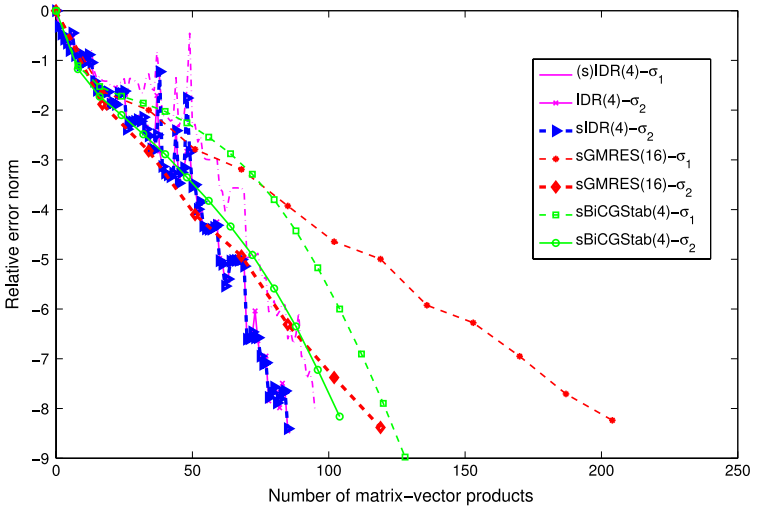

Fig. 2. Relative residual histories for example 2.

In order to see the convergence changes more clearly, relative residual histories of IDR(s), sIDR(s), sGMRES(m) and sBiCGStab(ℓ) corresponding to each linear system are also presented in Figs. 1 and 2. The vertical axis shows the log10 of relative residual norms, and the horizontal axis represents the number of matrix–vector products. For simplicity, for example sIDR(4)-σ_2 denotes relative residual norms of linear system $(A + \sigma_2 I)\boldsymbol{x} = \boldsymbol{b}$ solved by sIDR(4). Because IDR(4) and sIDR(4) obtained the same relative residual histories for solving $(A+\sigma_1 I)\boldsymbol{x} = \boldsymbol{b}$, we plot them using the same dash–dot line.

The results for example 1 are given in Fig. 1. For example 1, sGMRES(16) stagnated on both linear systems and could not converge. Other methods converged for both linear systems and the IDR-type methods took less matrix–vector products than sBiCGStab(4). Meanwhile, we see that sIDR(4) took almost the same matrix–vector products as that of IDR(4) for the *add* system and showed similar convergence behavior.

The results for example 2 are shown in Fig. 2. All methods converged for this example. We see that sGMRES(16) and (s)IDR(4) took the most and least matrix–vector products, respectively, for solving the *seed* and *add* systems. sIDR(4) and IDR(4) also took almost the same matrix–vector products and showed similar convergence behavior for the *add* system.

5. Conclusions

In this paper, we presented a shifted IDR(s) algorithm for solving shifted nonsymmetric linear systems. We observe that there exists a shift-invariance property for the Sonneveld subspaces. Together with the residual collinearity, we proposed the shifted IDR(s) method. Compared with the IDR(s) method for solving shifted linear systems individually, numerical results show that the convergence residual histories of shifted IDR(s) for the same linear system seem close to each other. Numerical

results also show that shifted IDR(s) is competitive with other shifted methods. Therefore, we believe that our shifted IDR(s) method can be used as an alternative method for solving the shifted linear systems.

Acknowledgments

This work was supported in part by the China Scholarship Council, Hori Sciences and Arts Foundation, JST/CREST and KAKENHI (Grant Nos. 21760058, 19560065 and 22104004).

References

[1] A. Frommer, U. Glässner, Restarted GMRES for shifted linear systems, SIAM J. Sci. Comput. 19 (1998) 15–26.
[2] A. Frommer, BiCGStab(ℓ) for families of shifted linear systems, Computing 70 (2003) 87–109.
[3] R.W. Freund, Solution of shifted linear systems by quasi-minimal residual iterations, in: L. Reichel, A. Ruttan, R.S. Varga (Eds.), Numerical Linear Algebra, W. de Gruyter, Berlin, 1993, pp. 101–121.
[4] Y.-F. Jing, T.-Z. Huang, Restarted weighted full orthogonalization method for shifted linear systems, Comput. Math. Appl. 57 (2009) 1583–1591.
[5] V. Simoncini, Restarted full orthogonalization method for shifted linear systems, BIT 43 (2003) 459–466.
[6] G.-D. Gu, Restarted GMRES augmented with harmonic Ritz vectors for shifted linear systems, Int. J. Comput. Math. 82 (2005) 837–849.
[7] G.-D. Gu, J.-J. Zhang, Z.-W. Li, Restarted GMRES augmented with eigenvectors for shifted linear systems, Int. J. Comput. Math. 80 (2003) 1037–1047.
[8] M.R. Hestenes, E.L. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand. 49 (1952) 409–436.
[9] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, PA, 2003.
[10] 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.
[11] 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.
[12] B. Jegerlehner, Krylov space solvers for shifted linear systems, December, 1996. Arxiv Preprint hep-lat/9612014.
[13] T. Sogabe, T. Hoshi, S.-L. Zhang, T. Fujiwara, A numerical method for calculating the Green's function arising from electronic structure theory, in: Y. Kaneda, H. Kawamura, M. Sasai (Eds.), Frontiers of Computational Science, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 189–195.
[14] R. Takayama, T. Hoshi, T. Sogabe, S.-L. Zhang, T. Fujiwara, Linear algebraic calculation of Green's function for large-scale electronic structure theory, Phys. Rev. B 73 (2006) 1–9.
[15] T. Sogabe, S.-L. Zhang, An extension of the COCR method to solving shifted linear systems with complex symmetric matrices, East Asian J. Appl. Math. 1 (2011) 97–107.
[16] T. Sogabe, T. Hoshi, S.-L. Zhang, T. Fujiwara, On a weighted quasi-residual minimization strategy for solving complex symmetric shifted linear systems, Electron. Trans. Numer. Anal. 31 (2008) 126–140.
[17] 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.
[18] 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.
[19] M.C. Yeung, T.F. Chan, ML(K)BiCGSTAB: a BiCGSTAB variant based on multiple Lanczos starting vectors, SIAM J. Sci. Comput. 21 (1999) 1263–1290.
[20] M.C. Yeung, ML(N)BiCGSTAB: reformulation, analysis and implementation, November, 2010. Arxiv Preprint arXiv:1011.5314.
[21] G.L.G. Sleijpen, P. Sonneveld, M.B. van Gijzen, Bi-CGSTAB as an induced dimension reduction method, Appl. Numer. Math. 60 (2010) 1110–1114.
[22] T.P. Collignon, M.B. van Gijzen, Minimizing synchronization in IDR(s), Numer. Linear Algebra Appl. 18 (2011) 805–825.
[23] M.H. Gutknecht, IDR explained, Electron. Trans. Numer. Anal. 36 (2010) 126–148.
[24] V. Simoncini, D.B. Szyld, Interpreting IDR as a Petrov–Galerkin method, SIAM J. Sci. Comput. 32 (2010) 1898–1912.
[25] 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.
[26] L. Du, T. Sogabe, S.-L. Zhang, Quasi-minimal residual smoothing technique for the IDR(s) method, JSIAM Lett. 3 (2011) 13–16.
[27] L. Du, T. Sogabe, S.-L. Zhang, A variant of the IDR(s) method with the quasi-minimal residual strategy, J. Comput. Appl. Math. 236 (2011) 621–630.
[28] G.L.G. Sleijpen, M.B. van Gijzen, Exploiting BiCGstab(ℓ) strategies to induce dimension reduction, SIAM J. Sci. Comput. 32 (2010) 2687–2709.
[29] M. Tanio, M. Sugihara, GBi-CGSTAB(s, L): IDR(s) with higher-order stabilization polynomials, J. Comput. Appl. Math. 235 (2010) 765–784.
[30] 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.
[31] M.H. Gutknecht, J.P.M. Zemke, Eigenvalue computations based on IDR, Research Report No. 2010-13, SAM, ETH Zurich & Bericht 145 INS, TUHH.
[32] M.B. van Gijzen, G.L.G. Sleijpen, J.P.M. Zemke, Flexible and multi-shift induced dimension reduction algorithms for solving large sparse linear systems, Report 11-06, DIAM, TU Delft & Bericht 156, INS, TU Hamburg-Hamburg, 2011.
[33] 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.
[34] 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.
[35] Matrix Market. Available online at http://math.nist.gov/MatrixMarket/.
[36] T.A. Davis, Y.F. Hu, The University of Florida sparse matrix collection, ACM Trans. Math. Software 38 (2011) http://www.cise.ufl.edu/research/sparse/matrices/.
[37] V. Simoncini, On the convergence of restarted Krylov subspace methods, SIAM J. Matrix Anal. Appl. 22 (2000) 430–452.
[38] S.H. Lam, The role of CSP in reduced chemistry modeling, in: IMA Symposium, University of Minnesota, 1999, pp. 13–19.
[39] H. Saberi Najafi, H. Ghazvini, Weighted restarting methods in the weighted Arnoldi algorithm for computing the eigenvalues of a nonsymmetric matrix, Appl. Math. Comput. 175 (2006) 1276–1287.