A flexible and adaptive Simpler GMRES with deflated restarting for shifted linear systems

Zhong, Hong-Xiu; Gu, Xian-Ming

Published in:
Computers & Mathematics with Applications

DOI:
10.1016/j.camwa.2019.03.017

Publication date:
2019

Computers and Mathematics with Applications 78 (2019) 997–1007

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

A flexible and adaptive Simpler GMRES with deflated restarting for shifted linear systems

Hong-Xiu Zhong [a,1], Xian-Ming Gu [b,c,*,2]

[a] School of Science, Jiangnan University, Wuxi, Jiangsu 214122, PR China
[b] School of Economic Mathematics/Institute of Mathematics, Southwestern University of Finance and Economics, Chengdu, Sichuan 611130, PR China
[c] Johann Bernoulli Institute of Mathematics and Computer Science, University of Groningen, Nijenborgh 9, P.O. Box 407, 9700 AK Groningen, The Netherlands

ARTICLE INFO

Article history:
Received 18 April 2018
Received in revised form 23 July 2018
Accepted 11 March 2019
Available online 2 April 2019

Keywords:
Shifted linear system
Adaptive Simpler GMRES
Flexible preconditioning
Deflated restarting

ABSTRACT

In this paper, two efficient iterative algorithms based on the Simpler GMRES method are proposed for solving shifted linear systems. To make full use of the shifted structure, the proposed algorithms utilizing the deflated restarting strategy and flexible preconditioning can significantly reduce the number of matrix–vector products and the elapsed CPU time. Numerical experiments are reported to illustrate the performance and effectiveness of the proposed algorithms.

1. Introduction

In this study, we are interested in efficient simultaneous solutions of the following large shifted linear systems:

$$(A + \alpha_j I)x(\alpha_j) = b, \quad j = 1, \ldots, s. \tag{1.1}$$

In general, $A \in \mathbb{C}^{n\times n}$ is non-singular and non-Hermitian, $\alpha_j \in \mathbb{C}$ is the shift such that $A + \alpha_j I$ is also non-singular, and α_j varies in a wide range; the right-hand side $b \in \mathbb{C}^n$ is fixed. Usually we take $\alpha_1 = 0$ as default; otherwise, Eq. (1.1) can be reset after a shift α_1. The first linear system is called the seed system, and others are the add systems. Such problem occurs in many scientific and engineering applications, such as structural dynamics [1,2], quantum chromodynamics [3], web search ranking [4], control theory [5,6] and so on. Therefore, there is a strong need for establishing efficient solutions of Eq. (1.1).

Many traditional methods (such as direct and iterative linear systems solvers) for the above problem are to solve $(A + \alpha_j I)x(\alpha_j) = b$ for each α_j, this trick can be quite expensive and prohibited when s and n are large. Fortunately, owing to the shift-invariance property of Krylov subspace, the Krylov subspace methods can solve Eq. (1.1) simultaneously [7]. That is, the Krylov subspace holds that

$$\mathcal{K}_m(A, b) = \mathcal{K}_m(A + \alpha_j I, b), \quad \forall \alpha_j \in \mathbb{C}.$$

Hence, all approximate solutions for (1.1) can be sought in a single space generated by the matrix A with the vector b.

* Corresponding author.
E-mail addresses: zhonghongxiu@126.com (H.-X. Zhong), guxianming@live.cn, guxm@swufe.edu.cn (X.-M. Gu).

1 This author is supported by NSFC (11701225), and the Natural Science Foundation of Jiangsu Province (BK20170173).

2 This author is supported by NSFC (11801463), the Fundamental Research Funds for the Central Universities (JBK1902028), and the Ministry of Education of Humanities and Social Science Project (19JYA790094).

https://doi.org/10.1016/j.camwa.2019.03.017
0898-1221/

The GMRES algorithm [8] is such a famous Krylov subspace method that it calculates the basis for $\mathcal{K}_k(A, b)$ by Arnoldi process with the initial guess $x_0 = 0$, hence the shifted system (1.1) can be solved cheaply if GMRES is performed for it simultaneously [5]. However, since the residuals $r_m(\alpha_j) = b - (A + \alpha_j I)x_m(\alpha_j)$ are not colinear, $\mathcal{K}_m(A, r_m) \neq \mathcal{K}_m(A + \alpha_j I, r_m(\alpha_j))$ with m being the restarting frequency. As a remedy, Frommer and Glässner have forced the residual vectors to be colinear [9], then restarts can again solve Eq. (1.1) cheaply. There are many variants based on GMRES for solving shifted linear systems. For instance, Gu, Zhang and Li proposed a variant of the restarted GMRES augmented with some approximate eigenvectors for the shifted system (1.1), refer to [10] for details. Later, Gu improved the restarted GMRES by augmenting the Krylov subspace with harmonic Ritz vectors for Eq. (1.1) [11]. By deflating eigenvalues for matrices that have a few small eigenvalues, Darnell, Morgan and Wilcox [12] presented an improved GMRES method with deflated restarting to accelerate the convergence. Gu, Zhou and Lin from another aspect of enhancing the convergence speed, proposed a flexible preconditioned Arnoldi method that is needed to exactly solve a linear system with the coefficient matrix $A + \sigma_k I$ at the kth iteration, where σ_k is the precondition reference value that draws near α_j. They also showed that their proposed method is greatly faster than the traditional preconditioning strategies [13]. Saibaba, Bakhos and Kitanidis have further extended the flexible preconditioning idea for solving generalized shifted linear systems arising from oscillatory hydraulic tomography [14]. Sun, Huang and Jing et al. [15,16] promoted the block version of GMRES method with deflated restarting for solving linear systems with multiple shifts and multiple right-hand sides. For other related methods, refer oneself to some studies in [17–26] and references therein.

As a cheaper implementation of GMRES, the Simpler GMRES algorithm (SGMRES) is another famous Krylov subspace method [27]. It runs the Arnoldi process to begin with Ar_0 instead of r_0, where $r_0 = b - Ax_0$. At each iteration, it only requires to solve an upper-triangular least-squares problem rather than an upper Hessenberg least-squares problem of GMRES, thus the SGMRES solver often spends less computational cost. Recently, Jing, Yuan and Huang applied the SGMRES and its stable variant: adaptive SGMRES (Ad-SGMRES) to solve the shifted system (1.1) [28]. For dealing with the non-colinearity of r_m and $r_m(\alpha_j)$, Jing, Yuan and Huang provided a remedy by forcing $r_m(\alpha_j) \perp A\mathcal{K}_m(A, r_0)$. Besides this advanced point, at each iteration step, from the non-converged systems, they took the linear system with the maximum residual norm as the seed system of the restart iteration.

However, in each cycle of the restarted methods, the convergence will slow down, since the dimension of the Krylov subspace is limited [13,29–32], especially for the problem with $A + \alpha_j I$ having small eigenvalues (in modulas). The main reason is that at each cycle, the Krylov subspace does not contain good approximations of the eigenvectors corresponding to such small eigenvalues. These make the thick-restarting and preconditioning techniques beneficial for solving Eq. (1.1). Unfortunately, as far as we know, unlike the shifted GMRES, there are not so many improved strategies applied to accelerate SGMRES for solving shifted linear systems (1.1). Thus, in this paper, we will first apply the flexible preconditioning technique [33] to the Ad-SGMRES for solving shifted linear systems (1.1), then consider restarting the new algorithm with the deflated restarting strategy introduced in [30,31]. The flexible preconditioning technique we used in this paper is the inexact preconditioning [34] instead of exact which is used in [13]. The details will be located in Section 2.

The remainder of this paper is organized as follows. In Section 2, we first give a brief description of the adaptive Simpler GMRES method (Ad-SGMRES), then present two variants of Ad-SGMRES for shifted linear system (1.1). Numerical examples in Section 3 will illustrate the effectiveness of the proposed algorithms. In Section 4, the paper closes with some conclusions.

2. A flexible and adaptive simpler GMRES algorithm with deflated restarting for shifted linear systems

In this section, applying the flexible preconditioning technique [14,33], we first derive a flexible adaptive Simpler GMRES algorithm (FAd-SGMRES-Sh) for solving shifted linear systems (1.1) simultaneously. Then based on it, we thick-restart the new algorithm by using the deflated restarting strategy [30,31,35]. Hence, a flexible and adaptive Simpler GMRES algorithm with deflated restarting (FAd-SGMRES-DR-Sh) will be achieved for solving Eq. (1.1).

Before giving the new algorithms, we will first briefly review the adaptive Simpler GMRES method. By introducing a threshold parameter $\nu \in [0, 1]$, Jiránek and Rozložník proposed the adaptive Simpler GMRES (Ad-SGMRES) [36], which is more stable than the Simpler GMRES, for solving the linear system $Ax = b$. The following algorithm is just the practical implementation of Ad-SGMRES.

Algorithm 1 (*The Adaptive Simpler GMRES (Ad-SGMRES)*)**.**

1. Given the initial guess x_0, a tolerance tol, a threshold parameter $\nu \in [0, 1]$, let m the maximal dimension of the solving subspace, $r_0 = b - Ax_0$;

2. For $k = 1, \ldots, m$, do

(1) $z_k = \begin{cases} r_0/\|r_0\|_2, & \text{if } k = 1, \\ r_{k-1}/\|r_{k-1}\|_2, & \text{if } k > 1, \text{ and } \|r_{k-1}\|_2 \leq \nu\|r_{k-2}\|_2, \\ v_{k-1}, & \text{otherwise.} \end{cases}$

(2) $v_k = Az_k$,

(3) for $i = 1, \ldots, k-1$

$u_{ik} = v_i^H v_k$, $v_k = v_k - u_{ik} v_i$.

end

(4) $u_{kk} = \|v_k\|_2$, $v_k = v_k/\|v_k\|_2$.
(5) $\xi_k = v_k^H r_{k-1}$, $r_k = r_{k-1} - v_k\xi_k$, if $\|r_k\|_2 \le tol$, then go to **Step 3**.

end

3. Let k be the final iteration number of **Step 2**, solve: $y_k = U_k^{-1}[\xi_1, \ldots, \xi_k]^H$. Set $x_k = x_0 + Z_k y_k$.

In Algorithm 1, the definitions of U_k and V_k can be found in the next section.

2.1. Flexible preconditioning

Suppose $r_0 = b - Ax_0 \neq 0$, where x_0 is the initial guess. At kth iteration of Ad-SGMRES (stated in Algorithm 1) for solving the seed system $Ax = b$, we have

$$AZ_k = V_k U_k, \tag{2.1}$$

where $Z_k = [z_1, \ldots, z_k] \in \mathbb{C}^{n\times k}$ is the basis of $\mathcal{K}_k(A, r_0)$, $V_k = [v_1, \ldots, v_k] \in \mathbb{C}^{n\times k}$ is the orthogonal basis of $A\mathcal{K}_k(A, r_0)$, $U_k = [u_{ij}] \in \mathbb{C}^{k\times k}$, $i, j = 1, \ldots, k$ is upper triangular, so U_k is non-singular because the coefficient matrix A is non-singular.

In [13], Gu, Zhou and Lin proposed a flexible preconditioning strategy for GMRES that it is needed to exactly solve a linear system with the coefficient matrix $A + \sigma_k I$ at the kth iteration, and it will cost a lot of time especially for large size problems. In this section, we will use the inexact flexible preconditioning [33,34,37] instead of exact. It is known that the traditional right preconditioning is applied to solve a modified system such as $AM^{-1}(Mx) = b$, where AM^{-1} is well conditioned. The inexact flexible preconditioning is actually a modification to the right preconditioning, i.e., M_k replaces M, so that inexact solver can be used. Based on such ideas, at each kth iteration, we set $w_k = M_k^{-1} z_k$, where M_k is a variable preconditioner. Denote $W_k = [w_1, \ldots, w_k]$, obviously, the columns of W_k may not span a Krylov subspace. For the absence of misunderstanding, we still use notions V_k and U_k. The relation (2.1) can be rewritten in the following matrix equation:

$$AW_k = V_k U_k. \tag{2.2}$$

For seed system, we seek the approximate solution $x_k = x_0 + W_k y_k$ in the affine subspace $x_0 + span\{W_k\}$, $y_k \in \mathbb{C}^k$ is a vector to be determined. Meanwhile, we seek the approximate solution $x_k(\alpha_j) = x_0(\alpha_j) + W_k y_k(\alpha_j)$ in the affine subspace $x_0(\alpha_j) + span\{W_k\}$ for add systems, where $y_k(\alpha_j) \in \mathbb{C}^k$ is a vector to be determined. For the add systems, we have

$$\begin{aligned}(A + \alpha_j I)W_k &= AW_k + \alpha_j W_k \\ &= V_k U_k + \alpha_j W_k.\end{aligned}$$

Since for W_k cannot be expressed by V_k, therefore, similar as in SGMRES [27], there exists no $U_k(\alpha_j)$ for the add systems to keep a similar relation to (2.2). Hence, it is impossible to force the residual vectors $r_k(\alpha_j)$ to be colinear to r_k.

For the seed system $Ax = b$, since the orthogonal condition is $r_k \perp span\{AW_k\}$, i.e., $r_k \perp span\{V_k\}$, then using (2.2), we get

$$\begin{aligned}0 &= V_k^H(b - Ax_k) \\ &= V_k^H(r_0 - AW_k y_k) \\ &= V_k^H r_0 - U_k y_k,\end{aligned} \tag{2.3}$$

and

$$\begin{aligned}r_k &= b - Ax_k \\ &= r_0 - V_k U_k y_k \\ &= r_0 - V_k V_k^H r_0 \\ &= r_{k-1} - v_k\xi_k,\end{aligned} \tag{2.4}$$

where $\xi_k = v_k^H r_0 = v_k^H r_{k-1}$. Thus (2.3) can be rewritten as

$$[\xi_1, \ldots, \xi_k]^H = U_k y_k. \tag{2.5}$$

Similar to the strategy in [28], for the add systems, we require the residual vector $r_k(\alpha_j) = b - (A + \alpha_j I)x_k(\alpha_j)$ being orthogonal to $span\{AW_k\}$, together with (2.2), we have

$$\begin{aligned}0 &= V_k^H[b - (A + \alpha_j I)x_k(\alpha_j)] \\ &= V_k^H(r_0(\alpha_j) - (AW_k + \alpha_j W_k)y_k(\alpha_j)) \\ &= V_k^H r_0(\alpha_j) - (U_k + \alpha_j V_k^H W_k)y_k(\alpha_j).\end{aligned} \tag{2.6}$$

Thus, after solving (2.5) and (2.6) to obtain y_k and $y_k(\alpha_k)$, the approximate solution of (1.1) is immediately accessed, and then

$$r_k(\alpha_j) = r_0(\alpha_j) - (AW_k + \alpha_j W_k)y_k(\alpha_j) = r_0(\alpha_j) - (V_k U_k + \alpha_j W_k)y_k(\alpha_j). \tag{2.7}$$

With the same seed system selection strategy in [28,38], we summarize our flexible and adaptive Simpler GMRES for solving shifted linear systems (FAd-SGMRES-Sh) in Algorithm 2. If the α_1 in seed system is not zero, we can reset

$$A \doteq A - \alpha_1 I,$$
$$\alpha_j \doteq \alpha_j - \alpha_1,$$

thus we take $\alpha_1 = 0$ as default.

Algorithm 2 (*A Flexible and Adaptive Simpler GMRES for Shifted Linear Systems (Fad-SGMRES-Sh)*)**.**

1. Start: Given the initial guess $x_0(\alpha_j)$, a tolerance tol, a threshold parameter $\nu \in [0, 1]$, let m be the maximal dimension of the solving subspace, $r_0(\alpha_j) = b - Ax_0(\alpha_j)$;
2. Select seed system: At the first iteration (after the second iteration), for all systems (for non-converged systems), find $ss \in \{1, \ldots, s\}$, where s is adjusted by the number of non-converged systems, such that

   $$\|r_0(\alpha_{ss})\|_2 = \max_{1\le j\le s} \|r_0(\alpha_j)\|_2.$$

   Re-order $r_0(\alpha_1), \ldots, r_0(\alpha_s)$, so that the residual of the seed system is placed in the first place. Thus, after re-ordering, $ss = 1$;
3. Iterate: for $k = 1, \ldots, m$, do

   (1) $z_k = \begin{cases} r_0/\|r_0\|_2, & \text{if } k = 1, \\ r_{k-1}/\|r_{k-1}\|_2, & \text{if } k > 1, \text{ and } \|r_{k-1}\|_2 \le \nu\|r_{k-2}\|_2, \\ v_{k-1}, & \text{otherwise.} \end{cases}$

   (2) $w_k = M_k^{-1} z_k$,

   (3) $v_k = Aw_k$,

   (4) for $i = 1, \ldots, k-1$
   $u_{ik} = v_i^H v_k$, $v_k = v_k - u_{ik} v_i$.
   end

   (5) $u_{kk} = \|v_k\|_2$, $v_k = v_k/u_{kk}$.

   (6) $\xi_k = v_k^H r_{k-1}$, $r_k = r_{k-1} - v_k \xi_k$, if $\|r_k\|_2 \le tol$, then go to **Step 4**.

   end
4. Let k be the final iteration number of **Step 3**.
   For seed system, solve (2.5);
   For add systems, $j = 2, \ldots, s$, solve (2.6), and update $r_k(\alpha_j)$ using (2.7);
5. Set $x_k(\alpha_j) = x_0(\alpha_j) + W_k y_k(\alpha_j)$, $j = 1, \ldots, s$. For the non-converged systems, reset $r_0(\alpha_j) = r_k(\alpha_j)$, $x_0(\alpha_j) = x_k(\alpha_j)$, $j = 1, \ldots, s$, go to **step 2**.

Some remarks of the implementation details for FAd-SGMRES-Sh are as follows.

Remark 1. In **Step 3**, M_k is the flexible preconditioner in the kth step. To get the effect of preconditioning, M_k is usually selected to be the matrix near A. In our algorithm, we choose to solve $Aw_k = z_k$ inexactly for the process $w_k = M_k^{-1} z_k$. There are many choices of inexact solvers, such as ILU [39], IHSS [40], IGMRES [39], ISOR [39], IQR [41], and so on. In numerical examples section, we select IGMRES with 10 iterations as the preconditioner.

Remark 2. In **Step 4**, for add systems, the matrix $U_k + \alpha_j V_k^H W_k$ is generally not upper triangular. Because we usually choose a small value $m \ll n$, such as 20, thus for the solving step $V_k^H r_0(\alpha_j) = (U_k + \alpha_j V_k^H W_k)y_k(\alpha_j)$, the MATLAB function "\" can be directly used to get $y_k(\alpha_j)$. In addition, from (2.7), we can see the update of the residual vectors will also cost some time. Consequently, for solving add systems, similar to SGMRES [28], FAd-SGMRES-Sh may not be faster than GMRES [17]. But fortunately, for seed system, due to without solving an upper Hessenberg least-square problem, and with inexact preconditioning, FAd-SGMRES-Sh is much faster than SGMRES, GMRES and FGMRES [13], especially for large-scale problems. Numerical experiments will illustrate the effect later.

2.2. Thick-restarting

Actually, some inexact preconditioned systems may still encounter the issues with small eigenvalues, thus it is necessary to consider to restart Algorithm 2 with the deflated restarting strategy [35,42,43]. Our aim is to improve the convergence of FAd-SGMRES-Sh by using the spectral information of the preconditioned seed system at restart. There are

two keys involved. The first is how to compute the spectral information at each restart. The second is how to apply these information with a low computation cost at restart.

In fact, we use the harmonic Ritz value information of the seed system $Ax = b$ at each restart. That is required, after one cycle, the harmonic Ritz pair $(\lambda_i, q_i \equiv W_m g_i)$ of A in $span\{W_m\}$ and orthogonal to $span\{AW_m\}$ satisfying [44]:

$$AW_m g_i - \lambda_i W_m g_i \perp span\{AW_m\} \Leftrightarrow (V_m U_m)^H (AW_m g_i - \lambda_i W_m g_i) = 0.$$

From (2.2), and U_m non-singular, the above equation is equivalent to

$$U_m g_i = \lambda_i V_m^H W_m g_i. \tag{2.8}$$

Consequently, the harmonic Ritz pairs can be calculated at each iteration of FAd-SGMRES-Sh. Let (λ_i, g_i), $i = 1, \ldots, e (e \leq m)$ are the eigenpairs of the reduced generalized eigenvalues problem (2.8). Set $G_e = [g_1, \ldots, g_e]$, suppose that $P_e L_e = G_e$ is the QR decomposition of G_e, where matrix $P_e = [p_1, \ldots, p_e] \in \mathbb{C}^{k \times e}$ is orthogonal. Postmultiplying (2.2) by P_e yields

$$AW_m P_e = V_m U_m P_e. \tag{2.9}$$

Let $U_m P_e = \widehat{P}_e U_e^{new}$ be the QR decomposition, then from (2.9) we have

$$AW_m P_e = V_m \widehat{P}_e U_e^{new}.$$

Define $W_e^{new} = W_m P_e$ and $V_e^{new} = V_m \widehat{P}_e$, then we obtain

$$AW_e^{new} = V_e^{new} U_e^{new},$$

where $V_e^{new} \in \mathbb{C}^{n \times e}$ is orthogonal, $U_e^{new} \in \mathbb{C}^{e \times e}$ is upper triangular. Let $W_e = W_e^{new}$, $V_e = V_e^{new}$ and $U_e = U_e^{new}$. To establish Eq. (2.2) for the current cycle, the flexible and adaptive Simpler GMRES with deflated restarting executes the remaining $(m - e)$ steps with $w_i = M_i^{-1} z_i (e + 1 \leq i \leq m)$ where M_i is the flexible preconditioner and

$$z_i = \begin{cases} r_e / \|r_e\|_2, & if\ i = e + 1, \\ r_{i-1} / \|r_{i-1}\|_2, & if\ i > e + 1\ and\ \|r_{i-1}\|_2 \leq \nu \|r_{i-2}\|_2, \\ v_{i-1}, & otherwise. \end{cases}$$

After each cycle of the new algorithm, we restart the algorithm by setting $x_0^{new}(\alpha_j) = x_m(\alpha_j)$ and $r_0^{new}(\alpha_j) = r_m(\alpha_j)$. We use the symbols such as $x_m^{new}(\alpha_j)$, $r_m^{new}(\alpha_j)$, W_m^{new}, V_m^{new} and U_m^{new} for current cycle to distinguish the ones from the last cycle.

For the seed system, after one cycle of FAd-SGMRES-Sh, from (2.4), we have

$$r_0^{new} = r_m = r_0 - V_m V_m^H r_0,$$

and

$$r_e^{new} = r_0^{new} - V_e^{new} (V_e^{new})^H r_0^{new}.$$

Note that

$$(V_e^{new})^H r_0^{new} = \widehat{P}_e^H V_m^H (r_0 - V_m V_m^H r_0) = 0.$$

Thus

$$r_e^{new} = r_0^{new}, \quad \xi_i^{new} = (v_i^{new})^H r_0^{new} = 0, \quad i = 1, \ldots, e,$$

then from (2.3) and (2.4), we need to solve

$$U_m^{new} y_m^{new} = [0, \ldots, 0, \xi_{e+1}^{new}, \ldots, \xi_m^{new}]^T, \tag{2.10}$$

where $\xi_i^{new} = (v_i^{new})^H r_0^{new} = (v_i^{new})^H r_{i-1}^{new}$, $i = e + 1, \ldots, m$, and update

$$r_i^{new} = r_{i-1}^{new} - v_i^{new} \xi_i^{new}. \tag{2.11}$$

For add systems, from (2.6) we can get

$$(V_e^{new})^H r_0(\alpha_j)^{new} = \widehat{P}_e^H V_m^H r_m(\alpha_j) = 0,$$

thus,

$$(V_m^{new})^H r_0(\alpha_j)^{new} = [0, \ldots, 0, \xi_{e+1}(\alpha_j)^{new}, \ldots, \xi_m(\alpha_j)^{new}]^T,$$

where $\xi_i(\alpha_j)^{new} = (v_i^{new})^H r_0(\alpha_j)^{new}$, $i = e + 1, \ldots, m$. Consequently, from (2.6), we need to solve

$$[0, \ldots, 0, \xi_{e+1}(\alpha_j)^{new}, \ldots, \xi_m(\alpha_j)^{new}]^T = (U_k^{new} + \alpha_j (V_k^{new})^H W_k^{new}) y_k(\alpha_j)^{new}, \tag{2.12}$$

and we still exploit (2.7) to update the residual vector. Now it is ready to present the main algorithm of this paper.

Table 1
Main computational costs per cycle for GMRES-Sh, Ad-SGMRES-Sh and FAd-SGMRES-Sh.

	GMRES-Sh	Ad-SGMRES-Sh	FAd-SGMRES-Sh
mv	m	m	m
Dot products	$m(\sum_{k=1}^{m}(k-1)+1)$	$m(\sum_{k=1}^{m}(k-1)+1+s)$	$m(\sum_{k=1}^{m}(k-1)+1+s)$
saxpy	$m(\sum_{k=1}^{m}(k-1)+1)+m+s$	$m(\sum_{k=1}^{m}(k-1)+1)+2s$	$m(\sum_{k=1}^{m}(k-1)+1)+2s$
op_{M_k}	0	0	m
Vector updates	$m+s+1$	$2m+2s$	$2m+2s$
G-p	0	0	0

Algorithm 3 (*A Flexible and Adaptive Simpler GMRES with Deflated Restarting for Shifted Linear Systems (Fad-SGMRES-DR-Sh)*)**.**

1. Start: Given the initial guess $x_0(\alpha_j)$, an integer e, a tolerance tol, a threshold parameter $\nu \in [0, 1]$, let m the maximal dimension of the solving subspace, $r_0(\alpha_j) = b - x_0(\alpha_j)$;
2. Select seed system: At the first iteration (after the second iteration), for all systems (for non-converged systems), find $ss \in \{1, \ldots, s\}$, where s is adjusted by the number of non-converged systems, such that

$$\|r_0(\alpha_{ss})\|_2 = \max_{1\le j\le s} \|r_0(\alpha_j)\|_2.$$

   Re-order $r_0(\alpha_1), \ldots, r_0(\alpha_s)$, so that the residual of the seed system is placed in the first place. Thus, after re-ordering, $ss = 1$;
3. Apply one cycle of FAd-SGMRES-Sh to the seed system $Ax = b$, generate W_m, V_m, U_m, x_m, and r_m;
4. Compute the eigenvalues and eigenvectors of the generalized eigenvalue problem (2.8) by using the QZ algorithm. Let $g_1, \ldots, g_e$ be the eigenvectors corresponding to the e smallest eigenvalues of (2.8). Set $G_e = [g_1, \ldots, g_e]$, and compute the QR decompositions of G_e and U_mP_e: $G_e = P_eL_e$, $U_mP_e = \widehat{P}_eU_e^{new}$. Let $W_e^{new} = W_mP_e$ and $V_e^{new} = V_m\widehat{P}_e$.
5. Let $W_e = W_e^{new}$, $V_e = V_e^{new}$, $U_e = U_e^{new}$, and $x_0 = x_m$, $r_0 = r_m$, $r_e = r_0$;
6. Iterate: for $k = e + 1, \ldots, m$, do

   (1) $z_k = \begin{cases} r_e/\|r_e\|_2, & \text{if } k = e+1, \\ r_{k-1}/\|r_{k-1}\|_2, & \text{if } k > e+1, \text{ and } \|r_{k-1}\|_2 \le \nu\|r_{k-2}\|_2, \\ v_{k-1}, & \text{otherwise.} \end{cases}$
   (2) $w_k = M_k^{-1}z_k$,
   (3) $v_k = Aw_k$,
   (4) for $i = 1, \ldots, k-1$
   $u_{ik} = v_i^Hv_k$, $v_k = v_k - u_{ik}v_k$.
   end
   (5) $u_{kk} = \|v_k\|_2$, $v_k = v_k/\|v_k\|_2$.
   (6) $\xi_k = v_k^Hr_{k-1}$, $r_k = r_{k-1} - v_k\xi_k$, if $\|r_k\|_2 \le tol$, then go to **Step 7**.
   end
7. Let k be the final iteration number of **Step 6**.
   For seed system, solve (2.10);
   For add systems, $j = 2, \ldots, s$, solve (2.6), and update $r_k(\alpha_j)$ using (2.7);
8. Set $x_k(\alpha_j) = x_0(\alpha_j) + W_ky_k(\alpha_j)$, $j = 1, \ldots, s$. For the non-converged systems, reset $r_0(\alpha_j) = r_k(\alpha_j)$, $x_0(\alpha_j) = x_k(\alpha_j)$, $j = 1, \ldots, s$, go to **step 2**.

In the end of this section, it is meaningful to evaluate the computational costs in a generic cycle of GMRES-Sh, Ad-SGMRES-Sh, FAd-SGMRES-Sh and FAd-SGMRES-DR-Sh, where the detailed pseudo-codes of GMRES-Sh and Ad-SGMRES-Sh are to be found in [28]. The comparisons are presented in Tables 1 and 2. Here, we denote "mv" the number of matrix–vector products. "op_{M_k}" denotes the number of the preconditioning process $M_k^{-1}z_k$ in one cycle, "vector updates" denotes the number of vectors that need to be updated in one cycle. We also write down the number of generalized eigenvalue problems by "G-p" in one cycle.

3. Numerical results

In this section, numerical comparisons are made for GMRES-Sh [17], Ad-SGMRES-Sh [28], FGMRES-Sh [13], GMRES-DR-Sh [43], FAd-SGMRES-Sh and FAd-SGMRES-Dr-Sh according to the number of outer matrix–vector products (referred to as mv), and the elapsed CPU time in seconds (referred to as cpu). We set the stopping criterion as

$$\frac{\|b - (A + \alpha_jI)x_k(\alpha_j)\|_2}{\|b\|_2} < 1\mathrm{e}-6, \quad j = 1, 2, \ldots, s.$$

Table 2
Main computational costs per cycle for the 1st cycle and the other cycle of FAd-SGMRES-DR-Sh.

	FAd-SGMRES-DR-Sh (1st cycle)	FAd-SGMRES-DR-Sh (other cycle)
mv	m	$m-e$
Dot products	$m(\sum_{k=1}^{m}(k-1)+1+s)$	$(m-e)(\sum_{k=1}^{m}(k-1)+1+s)$
saxpy	$m(\sum_{k=1}^{m}(k-1)+1)+2s$	$(m-e)(\sum_{k=1}^{m}(k-1)+1)+2s$
op_{M_k}	m	$m-e$
Vector updates	$2m+2s$	$2m+2s$
G-p	1	1

Table 3
The test matrices used in Example 3.1.

Matrix ID	Matrix name	Size	Nonzeros	Problem domain
1	add20	2,395	13,151	Circuit simulation
2	bidiag1	1,000	1,999	Academic
3	bidiag2	1,000	1,999	Academic
4	cdde1	961	4,681	Computational fluid dynamics
5	epb1	14,734	95,053	Thermal
6	sherman4	1,104	3,786	Computational fluid dynamics
7	wang1	2,903	19,093	Semiconductor device
8	wang4	26,068	177,196	Semiconductor device
9	young1c	841	4,089	Acoustics
10	young2c	841	4,089	Acoustics

The bold values in the following tables indicate the fastest in the terms of *cpu*. The numerical results are obtained by using MATLAB R2014a (64 bit) on an PC-Intel Core i5-6200U, CPU 2.4 GHz, 8 GB RAM with machine epsilon 10^{-16} in double precision floating point arithmetic.

Example 3.1. We consider the same matrices used in [28]. These matrices are from the University of Florida Sparse Matrix Collection and Example 1 in [45]. Table 3 lists the matrices with their information. Here bidiag1 and bidiag2 are bidiagonal matrices with super-diagonal entries being all one. The diagonal elements of bidiag1 are $0.1, 1, 2, 3, \ldots, 999$, and the ones of bidiag2 are $1, 2, 3, \ldots, 1000$. All the initial vectors are zero in all examples. The right-hand side b is generated by the MATLAB code *randn*$(n, 1)$, where n is the dimension of A. The shift parameters are $\alpha = 0, 0.4, 2$. For FAd-SGMRES-Sh and FAd-SGMRES-DR-Sh, the flexible preconditioner is chosen as running 10 steps of the un-restarted GMRES algorithm [8]. The same strategy is used in Example 3.2. For FGMRES-Sh, we use LU decomposition to exactly solve $(A+\sigma_i)w = v$ in the preconditioning process. Similar to the work [13], we select the same $\sigma_1 = 0.5$ in the first $m/2$ steps, in the last $m/2$ steps for the same $\sigma_2 = 1$. Thus, the LU decomposition of $A+\sigma_i I$ need to save for using in the first and last $m/2$ steps of each cycle. The same strategy is also used in Example 3.3.

In Table 4, we reported the $mv(cpu)$ of each algorithm for listed matrices with size smaller than 1000, and the dimension of the approximate subspace in each cycle is set as $m = 10$, $\mu = 0.9$. For FAd-SGMRES-Dr-Sh, e is the number of harmonic eigenvectors retained from the previous cycle. We compare two cases, i.e., $e = 3, 6$. In Table 5, for comparison, we set $m = 20$ and $e = 5, 10, 15$, with $\mu = 0.9$, and the matrices size are all larger than 1000. In all tables, "†" denotes that the algorithm fails to converge even after using 10 000 outer matrix–vector products.

As seen from Tables 4 and 5, for smaller matrices except for cdde1, FGMRES-Sh is the best solver among these algorithms, which is inseparable from the exact solution of $(A+\sigma_k)w = v$ during the preconditioning process. But for the larger matrices, especially for wang4 whose size is 26 068, the exact solving process of FGMRES-Sh obviously became a time-consuming obstacle, while FAd-SGMRES-DR-Sh performs best. It also can see for FAd-SGMRES-DR-Sh with different values e, in some examples, e.g., epb1 in Table 5, even the number mv is smaller, but the elapsed CPU time is larger, this is because when using the harmonic Ritz value information, we need to compute a generalized eigenvalue problem (2.8) and sort these eigenvalues; thus if the eigenvectors number e is larger, the elapsed CPU time for the previous procedure may be larger too. Thus, it is important to choose appropriate m and e. For some matrices, such as bidiag2, cdde1, add20 and sherman4, we can see the number mv of FAd-SGMRES-DR-Sh is not much less than FAd-SGMRES-Sh, even equal to each other, this is because after preconditioning, the small eigenvalue problems of these matrices are well controlled, thus the effect of deflated restarting is not obvious, whereas the other matrices still need the deflated restarting. Consequently, for large and difficult problems, FAd-SGMRES-DR-Sh still performs better than the other mentioned algorithms.

Example 3.2. In this example, we apply our algorithms to solve quantum chromodynamics (QCD) problems with multiple shifts, which is one of the most time-consuming supercomputer applications. D_i, $1 \le i \le 14$ are denoted the complex

Table 4
Convergence behaviors of GMRES-Sh, Ad-SGMRES-Sh, FGMRES-Sh, FAd-SGMRES-Sh, GMRES-DR-Sh and FAd-SGMRES-DR-Sh with `tol` $= 1e-6$, $m = 10$ and $\mu = 0.9$.

Method	$mv(cpu)$, $m = 10$, $\mu = 0.9$				
	bidiag1	bidiag2	cdde1	young1c	young2c
GMRES-Sh	4678(0.36)	513(0.06)	†	†	†
Ad-SGMRES-Sh	4678(0.34)	513(0.06)	9569(1.00)	†	†
FGMRES-Sh	7(**0.02**)	7(**0.01**)	118(0.07)	12(**0.13**)	11(**0.12**)
FAd-SGMRES-Sh	54(0.04)	35(0.03)	21(0.06)	627(0.73)	615(0.72)
GMRES-DR-Sh $e = 3$	351(0.81)	258(0.05)	174(0.10)	†	†
GMRES-DR-Sh $e = 6$	373(0.10)	240(0.06)	169(0.06)	†	†
FAd-SGMRES-DR-Sh $e = 3$	39(0.02)	32(0.02)	19(0.06)	231(0.34)	230(0.34)
FAd-SGMRES-DR-Sh $e = 6$	41(0.03)	32(0.02)	19(**0.04**)	193(0.25)	178(0.24)

Table 5
Convergence behaviors of GMRES-Sh, Ad-SGMRES-Sh, FGMRES-Sh, FAd-SGMRES-Sh, GMRES-DR-Sh and FAd-SGMRES-DR-Sh with `tol` $= 1e-6$, $m = 20$ and $\mu = 0.9$.

Method	$mv(cpu)$, $m = 20$, $\mu = 0.9$				
	add20	epb1	sherman4	wang1	wang2
GMRES-Sh	1231(0.32)	1300(1.17)	548(0.17)	1049(0.33)	†
Ad-SGMRES-Sh	1231(0.27)	1310(1.70)	548(0.10)	894(0.24)	†
FGMRES-Sh	635(11.39)	1099(9.74)	14(0.07)	295(1.38)	3161(268.17)
FAd-SGMRES-Sh	55(0.10)	72(0.59)	23(0.05)	51(0.12)	148(2.30)
GMRES-DR-Sh $e = 5$	629(0.27)	601(1.89)	134(0.09)	473(0.25)	1162(7.05)
GMRES-DR-Sh $e = 10$	†	591(2.30)	130(0.04)	496(0.23)	†
GMRES-DR-Sh $e = 15$	†	†	136(0.06)	†	†
FAd-SGMRES-DR-Sh $e = 5$	56(0.11)	63(**0.58**)	23(0.06)	44(0.11)	80(**1.25**)
FAd-SGMRES-DR-Sh $e = 10$	55(**0.08**)	63(0.59)	23(0.02)	**44(0.07)**	76(1.26)
FAd-SGMRES-DR-Sh $e = 15$	56(0.09)	63(0.70)	**23(0.01)**	44(0.08)	79(1.50)

matrices downloaded from Matrix Market.[3] These D_i are discretizations by the Dirac operator used in numerical simulation of quark behavior at different physical temperatures [3,18]. For each D_i, we take $A_i = (\frac{1}{k_c} + 10^{-3})I - D_i$ as the base matrix, where k_c is the critical value such that for $\frac{1}{k_c} < \frac{1}{k} < \infty$, the matrix $\frac{1}{k}I - D_i$ is *real-positive*. Table 6 lists the matrices D_i with their information. Moreover, the right-hand side $b =$ `ones(length(A), 1)`, and the initial guess in each example is zero vector. We take $[0.0001, 0.0002, \ldots, 0.0004, 0.001, 0.002, \ldots, 0.004, 0.01, 0.02, \ldots, 0.04]$ as the set of shifted values α_j. It is shown from Fig. 1 that the eigenvalues of base matrix A_1 are in the right-half of the complex plane, but partially surround the origin [12].

For seed matrices $A_1 - A_7$, we set $m = 10$, $\mu = 0.9$, and $e = 3, 6$. Table 7 gives the results of the considered algorithms. From Table 7, it can be seen that GMRES-DR-Sh does not converge for each matrix, and FGMRES-Sh costs too much time, whereas FAd-SGMRES-DR-Sh performs best; this implies that after adding inexact preconditioning and then deflating the small eigenvalues can accelerate the convergence. In Table 8, we compare the other algorithms besides FGMRES-Sh and GMRES-DR-Sh, and we set $m = 20$, $\mu = 0.9$, $e = 5, 10, 15$ for seed matrices $A_8 - A_{14}$. As seen from Tables 7 and 8, FAd-SGMRES-DR-Sh performs better than the other algorithms for most examples with deflating the small eigenvalues (in modulas). It is also known that the appropriate choice of m and e is important for FAd-SGMRES-DR-Sh, which will be subject to further investigations in the future.

Example 3.3. As we know, preconditioning is the critical point that effects the convergence of iteration methods directly [39]. However, different preconditioners will make different effects. In this example, some numerical results of FAd-SGMRES-Sh with different preconditioners are reported. We select ILU and IGMRES [39], and then denote the two algorithms by FAd-SGMRES-Sh(ILU) and FAd-SGMRES-Sh(IGMRES), respectively. At the same time, we also execute the flexible preconditioned GMRES with LU decomposition (FGMRES-Sh(LU)) [13] for comparison. All the matrices used in

[3] Refer to the website: http://math.nist.gov/MatrixMarket/.

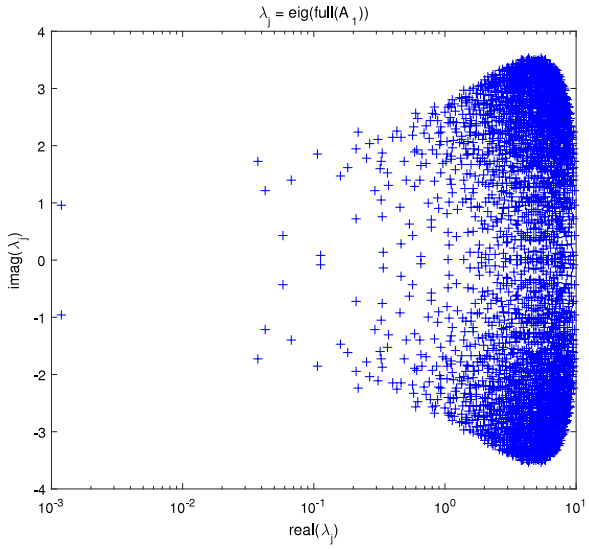

Fig. 1. The eigenvalues distribution of A_1.

Table 6
The matrices D_i used in Example 3.2.

Matrix ID	Denotation	Matrix name	Size	Nonzeros	k_c
1	D_1	CONF5.0-00L4X4-1000	3,072	119,808	0.20611
2	D_2	CONF5.0-00L4X4-1400	3,072	119,808	0.20328
3	D_3	CONF5.0-00L4X4-1800	3,072	119,808	0.20265
4	D_4	CONF5.0-00L4X4-2200	3,072	119,808	0.20235
5	D_5	CONF5.0-00L4X4-2600	3,072	119,808	0.21070
6	D_6	CONF6.0-00L4X4-2000	3,072	119,808	0.17968
7	D_7	CONF6.0-00L4X4-3000	3,072	119,808	0.16453
8	D_8	CONF5.4-00L8X8-0500	49,152	1,916,928	0.17865
9	D_9	CONF5.4-00L8X8-1000	49,152	1,916,928	0.17843
10	D_{10}	CONF5.4-00L8X8-1500	49,152	1,916,928	0.17689
11	D_{11}	CONF5.4-00L8X8-2000	49,152	1,916,928	0.17835
12	D_{12}	CONF6.0-00L8X8-2000	49,152	1,916,928	0.15717
13	D_{13}	CONF6.0-00L8X8-3000	49,152	1,916,928	0.15649
14	D_{14}	CONF6.0-00L8X8-8000	49,152	1,916,928	0.15623

Table 7
Convergence behaviors of Ad-SGMRES-Sh, FAd-SGMRES-Sh and FAd-SGMRES-DR-Sh with $n = 3072$, $\texttt{tol} = 1\texttt{e} - 6$, $m = 10$ and $\mu = 0.9$.

Method	$mv(cpu)$, $m = 10$, $\mu = 0.9$						
	A1	A2	A3	A4	A5	A6	A7
GMRES-Sh	812(0.56)	315(0.36)	634(0.57)	384(0.40)	564(0.53)	2357(1.78)	176(0.26)
Ad-SGMRES-Sh	812(0.86)	315(0.36)	634(0.70)	384(0.44)	564(0.62)	2357(2.53)	176(0.22)
FGMRES-Sh	6(21.67)	6(32.21)	5(21.91)	6(32.19)	6(21.60)	4(21.30)	4(21.56)
FAd-SGMRES-Sh	105(0.71)	60(0.46)	64(0.50)	63(0.46)	84(0.62)	70(0.49)	23(0.21)
GMRES-DR-Sh $e = 3$	†	†	†	†	†	†	†
GMRES-DR-Sh $e = 6$	†	†	†	†	†	†	†
FAd-SGMRES-DR-Sh $e = 3$	80(0.57)	54(0.42)	56(0.42)	57(0.42)	71(0.51)	52(0.39)	23(0.21)
FAd-SGMRES-DR-Sh $e = 6$	74(**0.50**)	51(**0.35**)	53(**0.35**)	56(**0.37**)	70(**0.49**)	48(**0.31**)	23(**0.16**)

the above two examples are considered in our experiments, and record the typical results in Table 9. Here *iter* denotes the iteration number of Arnoldi process.

As seen from Table 9, FGMRES-Sh(LU) and FAd-SGMRES-Sh(ILU) show almost the same performance for each matrices. Especially for smaller size matrices, both are performing better than FAd-SGMRES-Sh(IGMRES). However, for large-scale matrices, FAd-SGMRES-Sh(IGMRES) will be the best solver. This is because the inner loop of FGMRES-Sh(LU) becomes

Table 8
Convergence behaviors of Ad-SGMRES-Sh, FAd-SGMRES-Sh and FAd-SGMRES-DR-Sh with $n = 49152$, $\texttt{tol} = 1\texttt{e} - 6$, $m = 20$ and $\mu = 0.9$.

Matrix	$mv(cpu)$, $m = 20$, $\mu = 0.9$				
	Ad-SGMRES-Sh	FAd-SGMRES-Sh	FAd-SGMRES-DR-Sh		
			$e = 5$	$e = 10$	$e = 15$
A_8	872(18.71)	105(12.11)	95(**11.27**)	94(11.70)	92(12.90)
A_9	584(12.90)	79(9.47)	77(**9.10**)	76(9.48)	76(10.93)
A_{10}	471(10.34)	72(8.46)	71(8.50)	69(**8.41**)	69(9.38)
A_{11}	431(9.61)	71(**8.32**)	72(8.66)	71(8.80)	71(9.75)
A_{12}	659(15.11)	53(6.63)	50(6.06)	50(**5.94**)	50(6.31)
A_{13}	1010(21.72)	54(6.27)	51(**5.94**)	52(6.18)	51(6.61)
A_{14}	648(13.69)	54(6.13)	49(**5.63**)	49(5.83)	49(6.19)

Table 9
Convergence behaviors of FGMRES-Sh(LU), FAd-SGMRES-Sh(ILU) and FAd-SGMRES-Sh(IGMRES) with $\texttt{tol} = 1\texttt{e} - 6$, $m = 20$, $\mu = 0.9$, $\alpha = [0, 0.4, 2]$, and $\sigma_1 = 0.5$, $\sigma_2 = 1$.

Matrix	$iter(cpu)$, $m = 20$, $\mu = 0.9$		
	FGMRES-Sh(LU)	FAd-SGMRES-Sh(ILU)	FAd-SGMRES-Sh(IGMRES)
bidiag1	8(0.33)	3(**0.06**)	42(0.09)
sherman4	14(**0.07**)	17(0.16)	24(0.10)
wang4	1181(199.68)	1241(2433.90)	117(**1.88**)
young1c	11(**0.15**)	13(0.16)	299(0.41)
young2c	10(**0.03**)	13(0.11)	265(0.31)

time-consuming to exactly solve a linear system with the coefficient matrix $A + \sigma_i I$ using the LU decomposition, and the saving of the LU decomposition is another big cost. For FAd-SGMRES-Sh(ILU), although there is no storage about the LU decomposition, but in each cycle, it needs to calculate the incomplete LU decomposition of A and solving two sparse triangular linear systems, these are still both flaws. While for FAd-SGMRES-Sh(IGMRES), 10 steps of the inexact GMRES will not cost too much time. Consequently, for smaller size matrices, it is better to use FGMRES-Sh(LU) and FAd-SGMRES-Sh(ILU) to solve shifted systems, and it is best to use FAd-SGMRES-Sh(IGMRES) for solving some large-scale shifted systems.

4. Conclusions

In the present paper, we established two iterative algorithms based on the Simpler GMRES for solving shifted linear systems simultaneously, namely FAd-SGMRES-Sh and FAd-SGMRES-DR-Sh. Moreover, these variants can be regarded as two improvements of Ad-SGMRES-Sh, which is recently proposed by Jing, Yuan and Huang in [28]. The resultant algorithms converge in less matrix–vector products than the other related solvers (GMRES-Sh, Ad-SGMRES-Sh, FAd-GMRES-Sh, and GMRES-DR-Sh), especially for large problems. Furthermore, although the cost per iteration of FAd-SGMRES-Sh and FAd-SGMRES-DR-Sh is higher, in our numerical experiences, the overall execution time is still lower. In addition, the FAd-SGMRES-DR-Sh performs better than FAd-SGMRES-Sh when the coefficient matrix of the seed system has many eigenvalues close to the origin as verified by numerical experiments. In conclusion, the proposed algorithms can be recommended as two efficient tools for solving shifted linear systems.

As an outlook for the future, the advanced development of preconditioning strategies (such as the polynomial preconditioning [6,21], the nested iterative technique [46] and other preconditioning strategies [22,47]) for solving shifted linear systems remains a meaningful topic for further research.

Acknowledgments

The authors would like to thank the anonymous referees and Editor-in-Chief Prof. Leszek Feliks Demkowicz for their constructive comments that helped to improve the quality of the paper.

References

[1] A. Feriani, F. Perotti, V. Simoncini, Iterative system solvers for the frequency analysis of linear mechanical systems, Comput. Methods Appl. Mech. Eng. 190 (13–14) (2000) 1719–1739.
[2] V. Simoncini, F. Perotti, On the numerical solution of $(\lambda^2 A + \lambda B + C)x = b$ and application to structual dynamics, SIAM J. Sci. Comput. 23 (6) (2002) 1876–1898.
[3] D. Darnell, R.B. Morgan, W. Wilcox, Deflation of eigenvalues for iterative methods in lattice QCD, Nucl. Phys. B 129-130 (2004) 856–858.
[4] A.N. Langville, C.D. Meyer, Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, Princeton, NJ, 2006.

[5] B.N. Datta, Y. Saad, Arnoldi methods for large sylvester-like observer matrix equations and an associated algorithm for partial spectrum assignment, Linear Algebra Appl. 154–156 (1991) 447–454.
[6] M.I. Ahmad, D.B. Szyld, M.B. van Gijzen, Preconditioned multishift BiCG for $\mathcal{H}_2$-optimal model reduction, SIAM. J. Matrix Anal. Appl. 38 (2) (2017) 401–424.
[7] B. Jegerlehner, Krylov space solvers for shifted linear systems, Report IUHET-353, Department of Physics, Indiana University, 1996, 16 pages. Also available online at https://arxiv.org/pdf/hep-lat/9612014.pdf.
[8] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear system, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.
[9] A. Frommer, U. Glässner, Restarted GMRES for shifted linear systems, SIAM J. Sci. Comput. 19 (1) (1998) 15–28.
[10] G.-D. Gu, J.-J. Zhang, Z.-W. Li, Restarted GMRES augmented with eigenvectors for shifted linear systems, Int. J. Comput. Math. 80 (8) (2003) 1037–1047.
[11] G.-D. Gu, Restarted GMRES augmented with harmonic Ritz vectors for shifted linear systems, Int. J. Comput. Math. 82 (7) (2005) 837–849.
[12] D. Darnell, R.B. Morgan, W. Wilcox, Deflated GMRES for systems with multiple shifts and multiple right-hand sides, Linear Algebra Appl. 429 (10) (2008) 2415–2434.
[13] G.-D. Gu, X.-L. Zhou, L. Lin, A flexible preconditioned Arnoldi method for shifted linear systems, J. Comput. Math. 25 (5) (2007) 522–530.
[14] A.K. Saibaba, T. Bakhos, P.K. Kitanidis, A flexible Krylov solver for shifted systems with application to oscillatory hydraulic tomography, SIAM J. Sci. Comput. 35 (6) (2013) A3001–A3023.
[15] D.-L. Sun, T.-Z. Huang, Y.-F. Jing, B. Carpentieri, A block GMRES method with deflated restarting for sloving linear systems with multiple shifts and multiple right-hand sides, Numer. Linear Algebra Appl. 25 (5) (2018) e2148, http://dx.doi.org/10.1002/nla.2148, 1–21.
[16] D.-L. Sun, T.-Z. Huang, B. Carpentieri, Y.-F. Jing, A new shifted block GMRES method with inexact breakdowns for solving multi-shifted and multiple right-hand sides linear systems, J. Sci. Comput. 78 (2) (2019) 746–769.
[17] A. Frommer, BiCGStab(ℓ) For families of shifted linear systems, Computing 70 (2) (2003) 87–109.
[18] K.M. Soodhalter, D.B. Szyld, F. Xue, Krylov subspace recycling for sequences of shifted linear systems, Appl. Numer. Math. 81 (2014) 105–118.
[19] X.-M. Gu, T.-Z. Huang, J. Meng, T. Sogabe, H.-B. Li, L. Li, BiCR-type Methods for families of shifted linear systems, Comput. Math. Appl. 68 (7) (2014) 746–758.
[20] Y.-F. Jing, T.-Z. Huang, Restarted weighted full orthogonalization method for shifted linear systems, Comput. Math. Appl. 57 (9) (2009) 1583–1591.
[21] G. Wu, Y.-C. Wang, X.-Q. Jin, A preconditioned and shifted GMRES algorithm for the pagerank problem with multiple damping factors, SIAM J. Sci. Comput. 34 (5) (2012) A2558–A2575.
[22] V. Simoncini, Extended Krylov subspace for parameter dependent systems, Appl. Numer. Math. 60 (5) (2010) 550–560.
[23] L. Du, T. Sogabe, S.-L. Zhang, IDR(s) For solving shifted nonsymmetric linear systems, J. Comput. Appl. Math. 274 (2015) 35–43.
[24] 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, Numer. Linear Algebra Appl. 22 (1) (2015) 1–25.
[25] 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: Proceedings of the Conference in Numerical Linear Algebra and Scientific Computation, Kent (Ohio) USA, March (1992) 13-14, Walter de Gruyter, Berlin, 1993, pp. 101–121, http://dx.doi.org/10.1515/9783110857658.
[26] X.-M. Gu, T.-Z. Huang, G. Yin, B. Carpentieri, C. Wen, L. Du, Restarted hessenberg method for solving shifted nonsymmetric linear systems, J. Comput. Appl. Math. 331 (2018) 166–177.
[27] H. Walker, L. Zhou, A simpler GMRES, Numer. Linear Algebra Appl. 1 (6) (1994) 571–581.
[28] Y.-F. Jing, P. Yuan, T.-Z. Huang, A simpler GMRES and its adaptive variant for shifted linear systems, Numer. Linear Algebra Appl. 24 (1) (2017) e2076, http://dx.doi.org/10.1002/nla.2076, 1–7.
[29] Y.-Q. Lin, L. Bao, Q.-H. Wu, Simpler GMRES with deflated restarting, Math. Comput. Simulation 82 (11) (2012) 2238–2252.
[30] R. Boojhawon, M. Bhuruth, Restarted simpler GMRES augmented with harmonic Ritz vectors, Future Gener. Comput. Syst. 20 (3) (2004) 389–397.
[31] R. Boojhawon, D.Y. Tangman, K. Dookhitram, M. Bhuruth, Analysis of an implicitly restarted simpler GMRES variant of augmented GMRES, in: D. Taniar, O. Gervasi, B. Murgante, E. Pardede, B.O. Apduhan (Eds.), Computational Science and Its Applications C ICCSA 2010, in: ICCSA 2010. Lecture Notes in Computer Science, vol. 6017, Springer, Berlin, Heidelberg, 2010, pp. 570–585.
[32] H.-X. Zhong, G. Wu, G.-L. Chen, A flexible and adaptive simpler block GMRES with deflated restarting for linear system with multiple right-hand side, J. Comput. Appl. Math. 282 (2015) 139–156.
[33] D.B. Szyld, J.A. Vogel, FQMR: A flexible quasi-minimal residual method with inexact preconditioning, SIAM J. Sci. Comput. 23 (2) (2001) 363–380.
[34] V. Simoncini, D.B. Szyld, Theory of inexact krylov subspace methods and applications to scientific computing, SIAM J. Sci. Comput. 25 (2) (2003) 454–477.
[35] J. Meng, P.-Y. Zhu, H.-B. Li, X.-M. Gu, A deflated block flexible GMRES-DR method for linear systems with multiple right-hand sides, Electron. Trans. Numer. Anal. 41 (2014) 478–496.
[36] P. Jiránek, M. Rozložník, Adaptive version of simpler GMRES, Numer. Algorithms 53 (1) (2010) 93–112.
[37] Y. Sadd, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (2) (1993) 461–469.
[38] C. Li, H. Liu, A new seed projection method for solving shifted systems with multiple right-hand sides, Int. J. Math. Comput. Phys. Quantum Eng. 7 (4) (2013) 386–388.
[39] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, PA, 2003.
[40] Z.-Z. Bai, G.H. Golub, M.K. Ng, Hermitian and skew-hermitian splitting methods for non-hermitian positive definite linear systems, SIAM J. Matrix Anal. Appl. 24 (3) (2003) 603–626.
[41] Z.-Z. Bai, I.S. Duff, J.-F. Yin, Numerical study on incomplete orthogonal factorization preconditioners, J. Comput. Appl. Math. 226 (1) (2009) 22–41.
[42] A. Gaul, M.H. Gutknecht, J. Liesen, R. Nabben, A framework for deflated and augmented Krylov subspace methods, SIAM J. Matrix Anal. Appl. 34 (2) (2013) 495–518.
[43] R.B. Morgan, GMRES With deflated restarting, SIAM J. Sci. Comput. 24 (1) (2002) 20–37.
[44] C.C. Paige, B.N. Parlett, H.A. van der Vorst, Approximate solutions and eigenvalue bounds from Krylov subspaces, Numer. Linear Algebra Appl. 2 (2) (1995) 115–133.
[45] R.B. Morgan, Restarted block GMRES with deflation of eigenvalues, Appl. Numer. Math. 54 (2) (2005) 222–236.
[46] M. Baumann, M.B. van Gijzen, Nested Krylov methods for shifted linear systems, SIAM J. Sci. Comput. 37 (5) (2015) S90–S112.
[47] K.M. Soodhalter, Two recursive GMRES-type methods for shifted linear systems with general preconditioning, Electron. Trans. Numer. Anal. 45 (2016) 499–523.