Restarted weighted full orthogonalization method for shifted linear systems

Yan-Fei Jing*, Ting-Zhu Huang
School of Applied Mathematics, University of Electronic Science and Technology of China, Chengdu, Sichuan, 610054, PR China

A R T I C L E I N F O

Article history:
Received 19 September 2008
Accepted 17 October 2008

Keywords:
Shifted algebraic linear system
Arnoldi process
Restarted FOM
Accelerating convergence rate

A B S T R A C T

It is known that the restarted full orthogonalization method (FOM) outperforms the restarted generalized minimum residual method (GMRES) in several circumstances for solving shifted linear systems when the shifts are handled simultaneously. On the basis of the Weighted Arnoldi process, a weighted version of the Restarted Shifted FOM is proposed, which can provide accelerating convergence rate with respect to the number of restarts. In the cases where our hybrid algorithm needs less enough number of restarts to converge than the Restarted Shifted FOM, the associated CPU consuming time is also reduced, as shown by the numerical experiments. Moreover, our algorithm is able to solve certain shifted systems which the Restarted Shifted FOM cannot handle sometimes.

© 2008 Elsevier Ltd. All rights reserved.

1. Introduction

Given a real large sparse $n \times n$ nonsymmetric matrix $A \in \mathbb{R}^{n \times n}$ and the right-hand side $b \in \mathbb{R}^n$, it is of much interest to simultaneously solve the shifted nonsingular linear systems

$$ (A - \sigma I)x = b, \quad (1) $$

for several (say a few hundreds; see, e.g., [1,2]) tabulated values of the parameter $\sigma \in \mathbb{R}$, where $I$ denotes the identity matrix of proper dimension from the context. Such shifted systems arise in a variety of practical applications such as control theory, structural dynamics, higher-order implicit methods for solving time-dependent partial differential equations and quantum chromodynamics; see [1,3–8] and the references therein. It is well known that shifted matrices, which differ by a multiple of the identity only, generate the same Krylov subspaces associated with any fixed generating vector $v$, which is termed the shift-invariance property. As a matter of fact, in such circumstances, the Krylov subspace

$$ K_m(A, v) = \text{span}\{v, Av, \ldots, A^{m-1}v\} $$

is isomorphic to

$$ K_m(A - \sigma I, \hat{v}) = \text{span}\{\hat{v}, (A - \sigma I)\hat{v}, \ldots, (A - \sigma I)^{m-1}\hat{v}\} $$

with $\hat{v} = \beta v$, $0 \neq \beta \in \mathbb{R}$. Therefore, if we apply a Krylov subspace method to solve (1) simultaneously, a certain amount of computational efficiency can be maintained if the Krylov subspace is the same for all the shifted systems each time. This happens when the generating vectors are collinear, for the orthonormal basis and the square Hessenberg matrix are required to be calculated only once; see [8].

* Supported by NCET of Chinese ministry of education (NCET-04-0893), scientific and technological key project of Chinese ministry of education (107098).
* Corresponding author.

E-mail addresses: 00jyfvictory@163.com, yanfeijing@uestc.edu.cn (Y.-F. Jing), tingzhuhuang@126.com, tzhuang@uestc.edu.cn (T.-Z. Huang).
From the restarting point of view, there are two types of Krylov subspace methods for solving the shifted linear systems. The first type is based on Conjugate Gradient (CG) and Lanczos recurrences without restarting, including the shifted TFQMR method [2] and the Shifted two-sided Lanczos method [9]. Memory requirements however may limit their applicability to general shifted problems, since additional long vectors need to be stored for each shifted system simultaneously handled. The other type involves variants of restarted Generalized Minimum Residual Method (GMRES) [1,5,10–16]. Although those restarted GMRES-type methods are widely known and appreciated to be effective on (1), see [3], the computed GMRES shifted residuals are not collinear in general after the first restart so that it loses the computational efficiency mentioned above. Consequently, certain enforcement has to be made to guarantee the computed GMRES shifted residuals collinear to each other in order to maintain the computational efficiency; see, e.g., [1,5].

Compared to restarted GMRES, it is more natural and more effective for restarted Full Orthogonalization Method (FOM) to be applied to shifted linear systems simultaneously handled, for all residuals are naturally collinear. As a result, the computational efficiency can be maintained because the orthonormal basis and the square Hessenberg matrix are required to be calculated only once each time. For more details, refer to [8].

The motivation of this note is to improve the convergence behavior of the Restarted Shifted FOM [8] in respect of restarting number on the basis of the Weighted Arnoldi process proposed by Essai in [17]. Recently, the applications of the weighting techniques for the accelerating purpose in some aspects have been a subject of growing interest. For example, Saberi Najafi and Ghazvini [18] developed a weighted restarting algorithm to quickly obtain highly accurate eigenvalues of a nonsymmetric matrix. In addition, Cao and Yu [19] discussed the performance of the preconditioned weighted FOM and GMRES, which have not so good performance compared to preconditioned FOM(m) and preconditioned GMRES(m) though.

This paper presents a hybrid algorithm, termed the Restarted Weighted Shifted FOM, which in effect, brings together the best of the Restarted Shifted FOM [8] on the one hand and the Weighted Arnoldi process [17] on the other. Our method indeed may provide accelerating convergence rate with respect to the number of restarts at a little extra expense, which will be shown by the numerical experiments in Section 4. In some circumstances where our hybrid algorithm needs less enough number of restarts to converge than the Restarted Shifted FOM, it can amortize the extra cost in the Weighted Arnoldi process and consequently it can reduce the CPU computing time. Moreover, our algorithm is able to solve certain shifted systems which the Restarted Shifted FOM cannot handle sometimes.

The rest of the paper is organized as follows. Section 2 introduces the Weighted Arnoldi process, as well as some relations for the generated D-orthonormal basis with the square Hessenberg matrix. Section 3 presents the weighted version of the Restarted Shifted FOM, and some details on the strategy for the choice of the weights. Numerical experiments are shown in Section 4 and some conclusions are drawn in Section 5.

2. Weighted Arnoldi process

The Weighted Arnoldi process has been developed by Essai in [17]. We first briefly recall some knowledge of the Arnoldi process.

Arnoldi process is a procedure for building an Euclidean orthonormal basis of the Krylov subspace $K_m(A, v) = \text{span}\{v, Av, \ldots, A^{m-1}v\}$ with the generating vector $v$. In exact arithmetic, the following algorithm describes one variant of Arnoldi process using Modified-Gram–Schmidt algorithm with the starting vector $v_1 = v/\|v\|_2$; see, e.g., [20]:

**Algorithm 1 (Arnoldi Process–Modified-Gram–Schmidt).**

1. Choose a vector $v_1$ of 2-norm unity
2. For $j = 1, 2, \ldots, m$, Do:
3. Compute $\omega = Av_j$
4. For $i = 1, 2, \ldots, j$, Do:
5. $h_{ij} = (\omega, v_i)_2$
6. $\omega = \omega - h_{ij}v_i$
7. EndDo
8. $h_{j+1,j} = \|\omega\|_2$. If $h_{j+1,j} = 0$, stop
9. $v_{j+1} = \omega / h_{j+1,j}$
10. EndDo.

Denote by $V_m$, the $n \times m$ orthonormal matrix with column vectors $v_1, v_2, \ldots, v_m$, by $H_m$, the $(m + 1) \times m$ Hessenberg matrix whose non-zero entries $h_{ij}$ are constructed by the above Arnoldi process, and by $H_m$ the matrix obtained from $H_m$ by deleting its last row. Then the following well-known relations hold

$$AV_m = V_mH_m + h_{m+1,m}v_{m+1}^T = V_{m+1}H_m,$$

where $e_m^T$ indicates the real transpose of the $m$th canonical unit vector.
A shifted relation for the shifted system (1) is transformed into
\[(A - \sigma I)V_m = V_m(H_m - \sigma I_m) + h_{m+1,m}v_{m+1}e_m^T,\]  
where \(I_m\) is the identity matrix of dimension \(m\).

As addressed by Essai [17], the Weighted Arnoldi process uses, instead of the Euclidean scalar product, another one, denoted by \((\cdot, \cdot)_D\), where \(D\) is a chosen diagonal matrix. The main idea associated with non-Euclidean inner products is to have convergence of the components of the residual which are far away from zero. In order to achieve this purpose, an appropriate weight is associated to each term of the inner product. A natural choice of these weights is the absolute values of the entries of the first residual. Furthermore, at each restart, a different inner product is chosen. More details on the strategy for the choice of the weights will be provided in Section 3. Before representing the Weighted Arnoldi process, the definition of the \(D\)-scalar product needs to be given. Let \(D = \text{diag}(d_1, d_2, \ldots, d_n)\) be a chosen diagonal matrix with \(d_i > 0, \forall i \in \{1, 2, \ldots, n\}\), then the \(D\)-scalar product is defined as
\[(u, v)_D = u^TDu = \sum_{i=1}^{n} (d_i u_i v_i), \quad \forall u, v \in \mathbb{R}^n,
\]
and the \(D\)-norm \(\| \cdot \|_D\) associated with this inner product is
\[\|u\|_D = \sqrt{(u, u)_D}, \quad \forall u \in \mathbb{R}^n.\]

Now the Weighted Arnoldi process, which builds a \(D\)-orthonormal basis of the Krylov subspace \(\mathcal{K}_m(A, v) = \text{span}\{v, Av, \ldots, A^{m-1}v\}\) with the generating vector \(v\), can be described as follows with the starting vector \(\tilde{v}_1 = v/\|v\|_D^1\):

Algorithm 2 (Weighted Arnoldi Process [17]).

1. Choose a vector \(\tilde{v}_1\) of \(D\)-norm unity.
2. For \(j = 1, 2, \ldots, m\), Do:
3. \hspace{1cm} Compute \(\tilde{\omega} = A\tilde{v}_j\).
4. \hspace{1cm} For \(i = 1, 2, \ldots, j\), Do:
5. \hspace{2cm} \(\tilde{h}_{ij} = (\tilde{\omega}, \tilde{v}_i)_D\).
6. \hspace{1cm} \(\tilde{\omega} = \tilde{\omega} - \tilde{h}_{ij}\tilde{v}_i\).
7. \hspace{1cm} EndDo.
8. \(\tilde{h}_{j+1,j} = \|\tilde{\omega}\|_D\). If \(\tilde{h}_{j+1,j} = 0\), stop.
9. \(\tilde{v}_{j+1} = \tilde{\omega}/\tilde{h}_{j+1,j}\).
10. EndDo.

Vectors \(\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_m\) generated by Algorithm 2 form a \(D\)-orthonormal basis, in other words, if \(\tilde{V}_m = [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_m]\), then \(\tilde{V}_m^T\tilde{D}\tilde{V}_m = I_m\). Denote by \(\tilde{H}_m\), the \((m + 1) \times m\) Hessenberg matrix whose nonzero entries \(\tilde{h}_{ij}\) are constructed by Algorithm 2, and by \(\tilde{H}_m\) the matrix obtained from \(\tilde{H}_m\) by deleting its last row. Then an analogous relation to (2) still holds
\[A\tilde{V}_m = \tilde{V}_m\tilde{H}_m + h_{m+1,m}v_{m+1}e_m^T = \tilde{V}_{m+1}\tilde{H}_m,\]  
where \(e_m^T\) indicates the real transpose of the \(m\)th canonical unit vector. See the proof of (4) in [18]. For more details on the relations between the matrices generated in Algorithms 1 and 2, refer to [17].

Analogously to (3), the weighted shifted relation for the shifted system (1) is transformed into
\[(A - \sigma I)\tilde{V}_m = \tilde{V}_m(\tilde{H}_m - \sigma I_m) + h_{m+1,m}\tilde{v}_{m+1}e_m^T,\]  
where \(I_m\) is the identity matrix of dimension \(m\).

At the end of this section, let us have an investigation into the complexity of both the Arnoldi and the Weighted Arnoldi process.

Denote by \(\text{Nnz}\) the number of nonzero entries of the matrix \(A\). Both the Arnoldi and Weighted process require \(m\) steps and, at each step, a matrix-vector product is computed, then the cost is \(2m\text{Nnz}\) operations. Keep the difference between the Euclidean inner product and the \(D\)-scalar product in mind, we can approximately obtain the total number of operations required for both the Arnoldi and Weighted Arnoldi process listed in Table 1. For more analysis, see [17].

### 3. Restarted weighted shifted FOM

Given an initial approximate solution \(x_0\) to (1), the initial residual vector \(r_0\) is computed as \(r_0 = b - (A - \sigma I)x_0\). In the following, assume that all shifted linear systems have the same right-hand side and \(x_0 = 0\) so that the collinearity of the generating vectors for the Krylov subspaces are guaranteed for all the shifted linear systems at the beginning.

1. To avoid indexing overwhelming we shall adopt the tilde symbol \(\tilde{\cdot}\) for the weighted quantities.
Thanks to the similarity of (3) and (5), in our hybrid algorithm, the only difference in FOM is that \( \hat{y}_m \) is computed by solving the reduced shifted system \( (H_m - \sigma I_m) \hat{y} = \hat{\beta}_0 e_1 \) rather than \( (H_m - \sigma I_m) y = \beta_0 e_1 \) compared to the Restarted Shifted FOM proposed in [8], where \( \hat{\beta}_0 = \|r_0\|_D \), \( \beta_0 = \|r_0\|_2 \), and \( e_1 \) is the first canonical unit vector of length \( m \). The following proposition shows that collinearity for all the new residuals still holds in the weighted shifted case when FOM is applied.

**Proposition 1.** For each \( k = 1, 2, \ldots, s \), let \( \tilde{x}^{(k)}_m = \tilde{V}_m \tilde{y}^{(k)}_m \) be an FOM approximate solution to \( (A - \alpha_1 I_l) = b \) in \( \mathcal{K}_m(A - \alpha_1 I, b) \), with \( \tilde{V}_m \) satisfying (5) with \( \sigma = \alpha_1 \). Then there exists \( \beta_m \in \mathbb{R} \) such that \( \tilde{r}^{(k)}_m = b - (A - \alpha_1 I_l) \tilde{x}^{(k)}_m = \beta_m \tilde{v}^{(k)}_{m+1} \).

**Proof.** It is straightforward that \( \tilde{r}^{(k)}_m = \tilde{r}^{(k)}_{m+1} \) by setting \( \beta_m = -\tilde{r}^{(k)}_{m+1} \tilde{r}^{(k)}_m \), \( m = 1, 2, \ldots, s \), where \( \tilde{r}^{(k)}_m \) is the \( m \)th component of the vector \( \tilde{y}^{(k)}_m \) according to the proof of Proposition 2.1 in [8] by the comparison of (3) and (5). \( \Box \)

Now, the weighted version of the Restarted Shifted FOM in [8], named the Restarted Weighted Shifted FOM, is developed in general as follows

**Algorithm 3** (Restarted Weighted Shifted FOM).

Given \( A, b, m, \{\sigma_1, \sigma_2, \ldots, \sigma_s\} \), \( \text{Set}_{\text{index}} = \{1, 2, \ldots, s\} \):

1. Start: choose the initial approximate solutions to all shifted systems \( \tilde{x}^{(k)}_0 \), \( k \in \text{Set}_{\text{index}} \), and a tolerance \( \varepsilon \): compute \( \tilde{r}^{(k)}_0 = b - (A - \alpha_1 I) \tilde{x}^{(k)}_0 \), \( k \in \text{Set}_{\text{index}} \).
2. Choose diagonal matrices \( D^{(k)} = \text{diag}(d_1^{(k)}, d_2^{(k)}, \ldots, d_n^{(k)}) \), compute \( \tilde{\rho}_m^{(k)} = \|\tilde{r}^{(k)}_0\|_{D^{(k)}} \) and \( \tilde{v}_1^{(k)} = \tilde{r}^{(k)}_0 / \tilde{\rho}_m^{(k)} \), \( k \in \text{Set}_{\text{index}} \).
3. Construct the \( D \)-orthonormal bases \( \tilde{V}_m^{(k)} \) and the Hessenberg matrices \( \tilde{H}_m^{(k)} \), by the Weighted Arnoldi process, starting with the vectors \( \tilde{v}_1^{(k)} \), \( k \in \text{Set}_{\text{index}} \).
4. For each \( k \in \text{Set}_{\text{index}} \):
   \[ \tilde{y}^{(k)}_m = (\tilde{H}_m^{(k)} - \alpha_1 I_m)^{-1} e_1 \tilde{\rho}_m^{(k)} \]
   \[ \text{Set} \tilde{x}^{(k)}_m = \tilde{x}^{(k)}_0 + \tilde{V}_m^{(k)} \tilde{y}^{(k)}_m, \quad \tilde{r}^{(k)}_m = b - (A - \alpha_1 I_l) \tilde{x}^{(k)}_m. \]
5. Eliminate converged systems. Update Set_{index}. If Set_{index} = \emptyset, exist.
6. Restart: if \( \tilde{r}^{(k)}_m \parallel_{D^{(k)}} \tilde{r}^{(k)}_0 \parallel_{D^{(k)}} < \varepsilon \), stop;
   else set \( \tilde{x}^{(k)}_0 = \tilde{x}^{(k)}_m, \quad \tilde{r}^{(k)}_0 = \tilde{r}^{(k)}_m, \quad k \in \text{Set}_{\text{index}}, \) and goto 2.

Before illustrating the practical implementations of Algorithm 3, we first show a scaling-invariant property of Algorithm 3.

**Proposition 2.** If the diagonal matrices \( D^{(k)} \) are replaced by \( \alpha_k D^{(k)} \), where \( \alpha_k \) is a positive real scalar, \( k = 1, 2, \ldots, s \), then the results of Algorithm 3 will not change.

**Proof.** Denote \( \tilde{D}^{(k)} = \alpha_k D^{(k)} \). \( k = 1, 2, \ldots, s \), then according to Algorithm 2, for \( j = 1, 2, \ldots, m \), \( i = 1, 2, \ldots, j + 1 \), we have

\[
\tilde{v}^{(k)}_i = \frac{1}{\sqrt{\alpha_k}} \hat{v}^{(k)}_i, \quad \tilde{\omega}^{(k)} = \frac{1}{\sqrt{\alpha_k}} \hat{\omega}^{(k)},
\]

and

\[
\tilde{h}^{(k)}_{ij} = (\tilde{\omega}^{(k)}, \tilde{v}^{(k)}_i)\tilde{D}^{(k)} = \frac{1}{\alpha_k} (\hat{\omega}^{(k)}, \hat{v}^{(k)}_i)\hat{D}^{(k)} = (\hat{\omega}^{(k)}, \hat{v}^{(k)}_i)\hat{D}^{(k)} = \tilde{h}^{(k)}_{ij},
\]

i.e., we have

\[
\tilde{V}_m^{(k)} = \frac{1}{\sqrt{\alpha_k}} \hat{V}_m^{(k)}, \quad \tilde{H}^{(k)} = \hat{H}^{(k)}, \quad \tilde{H}_m^{(k)} = \hat{H}_m^{(k)}.
\]

(6)

\[ ^2 \text{We shall adopt the hat symbol \^{} for the scaling quantities.} \]
It is observed that
\begin{equation}
\hat{\beta}^{(k)}_m = \| \tilde{\varepsilon}^{(k)}_0 \| = \sqrt{\alpha_k \| \tilde{r}^{(k)}_0 \|_D} = \sqrt{\alpha_k \hat{\beta}^{(k)}_m} .
\end{equation}

In Algorithm 3 for the scaling solutions, \( \tilde{x}^{(k)}_m = \tilde{x}^{(k)}_0 + \tilde{V}_m \tilde{y}^{(k)}_m \), where \( \tilde{x}^{(k)}_0 = \tilde{x}^{(k)}_0 \) and \( \tilde{y}^{(k)}_m \) satisfies \((H_m - \sigma_k I_m) \tilde{y}^{(k)}_m = \hat{\beta}^{(k)}_m e_1\).

By (6) and (7), we have \( \hat{y}^{(k)}_m = \sqrt{\alpha_k} y^{(k)}_m \) and
\begin{equation}
\hat{\beta}^{(k)}_m = \sqrt{\alpha_k} \hat{\beta}^{(k)}_m
\end{equation}
which completes the proof.

We remark that, due to the invariance of Krylov subspaces with respect to shifting and scaling, namely, the generated space is the same for the nonshifted case. In view of this, Proposition 2 appears to be straightforward. From Proposition 2, it can be found that it is unnecessary to scale the weighting matrices \( D^{(k)} \), \( k = 1, 2, \ldots, s \) with positive scalars because the scaling can do nothing about the generated Krylov subspace as well as the solutions to (1).

As was assumed before, \( \tilde{x}^{(k)}_0 = 0 \), \( k = 1, 2, \ldots, s \) for all the shifted linear systems so that all of them have the same initial residual vectors, that is, \( \tilde{r}^{(k)}_0 = b \), \( k = 1, 2, \ldots, s \). Various implementations of Algorithm 3 may be obtained by varying the weighting strategy in line 2. However, it seems to make no difference on the convergence behavior with respect to the number of restarts associated with different weighting strategies, as will be revealed by the following three strategies to some extent. For convenience, \( D^{(k)} \), \( k = 1, 2, \ldots, s \) is denoted as \( D \) in the first two strategies, since all the shifted linear systems have the same weights at each restart. \( u_i \), \( i = 1, 2, \ldots, n \), denotes the \( i \)-th component of the vector \( u \in \mathbb{R}^n \), with \( |u_i| \) denoting the corresponding absolute value.

Strategy 1. In the initialization, the weights for all the shifted linear systems are the same, which are set to be
\begin{equation}
\hat{d}^{(k)}_i = \sqrt{n} \left| \tilde{r}^{(k)}_0 \right|_2 = \sqrt{n} \left| \tilde{b} \right|_2 , \quad i = 1, 2, \ldots, n, \quad k = 1, 2, \ldots, s .
\end{equation}

During the restarting procedure, in virtue of the results in Proposition 1 as well as the fact that \( \tilde{V}_m \) is \( D \)-orthonormal, we get
\begin{equation}
\hat{d}^{(k)}_i = \sqrt{n} \left| \tilde{r}^{(k)}_0 \right|_2 = \sqrt{n} \left| \tilde{r}^{(k)}_m \tilde{V}_m \right|_2 = \sqrt{n} \left| \tilde{V}_m \right|_2 = \sqrt{n} | \tilde{V}_m | = \sqrt{n} | \tilde{v}_m |.
\end{equation}

Strategy 2. Through the whole computation, the weight vector
\begin{equation}
d = [d^{(k)}_1, d^{(k)}_2, \ldots, d^{(k)}_s]^T , \quad k = 1, 2, \ldots, s
\end{equation}
is chosen to satisfy \( \| d \|_D = \sqrt{n} \) so that the \( D \)-norm recovers the Euclidean norm if all the elements of \( d \) are equal to unity. At the beginning,
\begin{equation}
\hat{d}^{(k)}_i = \sqrt{n} \left| \tilde{r}^{(k)}_0 \right|_2 = \sqrt{n} \left| \tilde{b} \right|_2 , \quad i = 1, 2, \ldots, n, \quad k = 1, 2, \ldots, s .
\end{equation}

Therefore, we have
\begin{equation}
\| b \|_D^2 = (Db, b) \geq \frac{n}{\| b \|_D} \sum_{i=1}^{n} |(b)_i|^2 ,
\end{equation}
resulting in
\begin{equation}
\| b \|_D = \left( \sqrt{n} \sum_{i=1}^{n} |(b)_i|^2 \right)^{\frac{1}{2}},
\end{equation}
so we can fix the weights in (8) with (9) at the beginning. The successive choice for the weights in each restart is the same as that in Strategy 1.

Strategy 3. The weights are always chosen to be
\begin{equation}
\hat{d}^{(k)}_i = \sqrt{n} \left| \tilde{r}^{(k)}_m \right|_2 , \quad i = 1, 2, \ldots, n, \quad k = 1, 2, \ldots, s .
\end{equation}
Comparison among the three strategies with Examples 1 and 2 in terms of both number of restarts and CPU consuming time in seconds (CPU for short).

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Example 1</th>
<th>Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_1 = -1 )</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( \sigma_2 = 1 )</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>CPU</td>
<td>0.063</td>
<td>0.063</td>
</tr>
</tbody>
</table>

Table 3
Example 1. Comparison in terms of both number of restarts and CPU consuming time in seconds (CPU for short).

<table>
<thead>
<tr>
<th>Approach</th>
<th>RWS-FOM</th>
<th>RS-FOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_1 = -1 )</td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>( \sigma_2 = 1 )</td>
<td>24</td>
<td>33</td>
</tr>
<tr>
<td>CPU</td>
<td>0.063</td>
<td>0.031</td>
</tr>
</tbody>
</table>

It is easy to observe that the computed weighted shifted residuals with Strategy 3 are not collinear after the first restart in general. So Algorithm 3 with such strategy can only be applied to each shifted system separately, suffering the same problem confronting the restarted GMRES-type methods; see [8]. The aim for supplying this strategy is to investigate the effect of different weighting choices in Algorithm 3. Concerning the optimal weights as mentioned in [17], we might get the conclusion that the optimal weights may not exist for Algorithm 3, because the convergence behavior in respect of number of restarts remains the same even for different choices of \( D^{(k)} \), \( k = 1, 2, \ldots, s \) at each restart, as will be revealed by Table 2 in the next section.

4. Numerical experiments

In order to show the accelerating convergence behavior with respect to number of restarts using Algorithm 3 to solve \((1)\) simultaneously, some numerical experiments have been carried out in this section far from being exhaustive. We compare the Restarted Weighted Shifted FOM with Strategy 1 (referred to as RWS-FOM) and the Restarted Shifted FOM in [8] (referred to as RS-FOM). The reason for employing Strategy 1 is that all of the three strategies supplied in the previous section perform the same convergence behavior in respect of number of restarts while Strategy 1 and 2 require almost the same operations, much less than that of Strategy 3; see Table 2 for the comparison among the three strategies taking Examples 1 and 2 for instance in terms of both number of restarts and CPU consuming time in seconds (CPU for short). Therefore, without loss of generality, we adopt Strategy 1 in Algorithm 3 for the purpose of comparison with RS-FOM.

All experiments were performed on a PC-Pentium(R) 4, CPU 3.06 GHz, 512M of RAM using MATLAB 6.5 with machine epsilon \(10^{-10}\). Exact arithmetic is assumed throughout the paper. The right-hand side is the vector of all ones, normalized to have 2-norm unity, and two values for the shift parameter are considered, \( \sigma_1 = -1 \), \( \sigma_2 = 1 \). The stopping criterion is that the current relative residual satisfies \( \| r^{(m)} \|_2 < \varepsilon = 10^{-8} \). All tests here are started with an initial guess equal to zero.

The dimension of the Krylov subspace is chosen to be \( m = 10 \).

Now, we give the numerical comparison results for RWS-FOM and RS-FOM by two means. The relations between the number of restarts as \( x \)-axis and the relative residual’s logarithm based on 10 as \( y \)-axis are depicted in the form of figures, separately. The comparison results in terms of both number of restarts and CPU consuming time in seconds are reported in Tables 3–6 for the corresponding experiments.

Example 1 ([8,14]). The first example considers a \( 100 \times 100 \) upper bidiagonal matrix \( A \) with diagonal the vector \( d = [0.001, 0.002, 0.003, 0.004, 0.1, 0.11, \ldots, 1.05] \) and super-diagonal the vector of all ones. From Table 3, RWS-FOM saves a little number of restarts compared to RS-FOM at an expense of extra operations. Obviously, there still exists oscillation for the relative residuals in RWS-FOM as occurred in RS-FOM reported in the right plot of Fig. 1.

Example 2 ([18,21]). Let the matrix \( A \) be \( 200 \times 200 \) used in the corresponding literature as follows

\[
A = \begin{bmatrix}
1 & 0.21 & 1.2 & 0.13 & 1.42 \\
0.45 & 2 & 0.21 & 1.2 & 0.13 & 1.42 \\
0 & 0.45 & 3 & 0.21 & 1.2 & 0.13 \\
0.12 & 0 & 0.45 & 4 & 0.21 & 1.2 \\
0.11 & 0.12 & 0 & 0.45 & 1.2 & 0.13 \\
0.11 & 0.12 & 0 & 0.45 & 197 & 0.12 \\
0 & 0.45 & 198 & 0.21 & 1.2 & 0 \\
0.11 & 0.12 & 0 & 0.45 & 199 & 0.21 \\
0.11 & 0.12 & 0 & 0.45 & 200 & 0.21
\end{bmatrix}
\]
Table 4
Example 2. Comparison in terms of both number of restarts and CPU consuming time in seconds (CPU for short).

<table>
<thead>
<tr>
<th>Approach</th>
<th>RWS-FOM</th>
<th>RS-FOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1 = -1$</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>$\sigma_2 = 1$</td>
<td>32</td>
<td>123</td>
</tr>
<tr>
<td>CPU</td>
<td>0.125</td>
<td>0.187</td>
</tr>
</tbody>
</table>

Table 5
Example 3. Comparison in terms of both number of restarts and CPU consuming time in seconds (CPU for short).

<table>
<thead>
<tr>
<th>Approach</th>
<th>RWS-FOM</th>
<th>RS-FOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1 = -1$</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>$\sigma_2 = 1$</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>CPU</td>
<td>6.344</td>
<td>4.110</td>
</tr>
</tbody>
</table>

Table 6
Example 4. Comparison in terms of both number of restarts and CPU consuming time in seconds (CPU for short).

<table>
<thead>
<tr>
<th>Approach</th>
<th>RWS-FOM</th>
<th>RS-FOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1 = -1$</td>
<td>35</td>
<td>100</td>
</tr>
<tr>
<td>$\sigma_2 = 1$</td>
<td>37</td>
<td>100</td>
</tr>
<tr>
<td>CPU</td>
<td>25.578</td>
<td>61.547</td>
</tr>
</tbody>
</table>

As observed in Table 4 and Fig. 2, significant progress has been made by RWS-FOM not only in reduced number of restarts but also in decreased CPU computing time. Under such circumstances where RS-FOM needs much more number of restarts to converge than RWS-FOM, RWS-FOM is much preferred.

Example 3 ([22]). This example concerns the three-dimensional convection–diffusion equation

$$-(u_{xx} + u_{yy} + u_{zz}) + q(u_x + u_y + u_z) = f(x, y, z)$$

on the unit cube $\Omega = [0, 1] \times [0, 1] \times [0, 1]$, with constant $q$ and Dirichlet-type boundary conditions.

Discretizing the above equation with seven-point finite difference upwind scheme and assuming that the grid points in all the three directions have the same number ($n$), a positive definite system with coefficient matrix $A(n^3 \times n^3)$ can be obtained; for details, see [23–25], Different $q$ and $n$ give rise to different matrices $A$. The mesh spacing is defined as $h = \frac{1}{n+1}$, and $r = \frac{qh^2}{2}$ is the mesh Reynolds number. Here, we set $n = 32$, $q = 100$ so that a matrix $A$ of order $n \times n \times n = 32 \times 32 \times 32 = 32768$ is formed. We reported both the number of restarts and the CPU consuming time in seconds in Table 5. From Fig. 3, we get a glimpse of a sharp decrease of the relative residual for RWS-FOM with $\sigma_2 = 1$ as that in the right plot of Fig. 2.

Example 4 ([26]). The final example is the matrix stems from the 2-D parallel plate waveguide problem taken as the first numerical example in [26], which generates a quite ill-conditioned matrix $A$ of order 7120. We set the maximal number of restarts to be 100 for both RWS-FOM and RS-FOM. The remarkable comparison results are listed in Table 6. Observed from Fig. 4, while RS-FOM stagnates on both shifted systems, RWS-FOM is able to converge. In such cases, RWS-FOM outperforms RS-FOM deeply not only in both computing time and number of restarts, but also in the solvability for the shifted linear systems.
5. Concluding remarks

Based on the results of numerical experiments we conclude that our hybrid algorithm – the Restarted Weighted Shifted FOM (referred to as RWS-FOM) – indeed can lead to accelerating convergence rate with respect to the number of restarts compared to the Restarted Shifted FOM (referred to as RS-FOM) proposed in [8]. Under certain conditions where RWS-FOM requires less enough number of restarts to converge, this algorithm can also reduce the CPU consuming time. Moreover, RWS-FOM is able to handle certain shifted linear systems while RS-FOM cannot.
From the weighting point of view, many of the Krylov subspace methods can be retrofitted to the simultaneous solution of shifted linear systems with certain enforcement. Future work may focus on simultaneously solving the shifted linear systems with the weighted versions of other Krylov subspace methods.

However, the body of theoretical evidence is not available recently for the fact that RWS-FOM has advantage over RS-FOM. The numerical and computational efficiency of RWS-FOM in respect of restarting number is just illustrated on a set of problems arising both from academic and from industrial applications. Furthermore, convergence analysis is also under consideration.

References