Numerical Algorithms (2019) 82:155–181
https://doi.org/10.1007/s11075-018-0597-9

ORIGINAL PAPER

# Weighted and deflated global GMRES algorithms for solving large Sylvester matrix equations

Najmeh Azizi Zadeh[1] · Azita Tajaddini[1] · Gang Wu[2]

Received: 15 August 2017 / Accepted: 10 September 2018 / Published online: 18 September 2018

## Abstract

The solution of a large-scale Sylvester matrix equation plays an important role in control and large scientific computations. In this paper, we are interested in the large Sylvester matrix equation *with large dimension $A$ and small dimension $B$*, and a popular approach is to use the global Krylov subspace method. In this paper, we propose three new algorithms for this problem. We first consider the global GMRES algorithm with weighting strategy, which can be viewed as a precondition method. We present three new schemes to update the weighting matrix during iterations. Due to the growth of memory requirements and computational cost, it is necessary to restart the algorithm effectively. The deflation strategy is efficient for the solution of large linear systems and large eigenvalue problems; to the best of our knowledge, little work is done on applying deflation to the (weighted) global GMRES algorithm for large Sylvester matrix equations. We then consider how to combine the weighting strategy with deflated restarting, and propose a weighted global GMRES algorithm with deflation for solving large Sylvester matrix equations. In particular, we are interested in the global GMRES algorithm with deflation, which can be viewed as a special case when the weighted matrix is chosen as the identity. Theoretical analysis is given to show rationality of the new algorithms. Numerical experiments illustrate the numerical behavior of the proposed algorithms.

**Keywords** Large Sylvester matrix equation · Global GMRES · Weighting strategy · Deflated restarting · Krylov subspace

**Mathematics Subject Classification (2010)** 65F10 · 65F15 · 15A24 · 65F30

This author is supported by the National Science Foundation of China and the Natural Science Foundation of Jiangsu Province under grant BK20171185.

✉ Gang Wu
gangwu76@126.com; gangwu@cumt.edu.cn

Extended author information available on the last page of the article.

## 1 Introduction

The Sylvester matrix equation plays an important role in control and communications theory, model reduction, image restoration, signal processing, filtering, and decoupling techniques for ordinary partial differential equations, as well as the implementation of implicit numerical methods for ordinary differential equations (see [3, 5–7, 13, 15, 39], and the references therein). In this paper, we consider the large Sylvester matrix equation *with large dimension $A$ and small dimension $B$* [39, sec.4.3]

$$AX + XB = C, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{s\times s}$, $C \in \mathbb{R}^{n\times s}$ and $X \in \mathbb{R}^{n\times s}$, with $s \ll n$. This type of Sylvester matrix equation arises in many areas such as the solution of eigenvalue problems [42, sec.2.4], and in (separable) boundary value problems [39, 41].

If we define the operator $\mathcal{A}$ as

$$\begin{aligned} \mathcal{A} :\ & \mathbb{R}^{n\times s} \longrightarrow \mathbb{R}^{n\times s}, \\ X \to\ & \mathcal{A}X = AX + XB. \end{aligned} \tag{1.2}$$

Then, the Sylvester matrix (1.1) can be written as

$$\mathcal{A}X = C. \tag{1.3}$$

Also, (1.3) can be rewritten as the following large linear system

$$\left(I_s \otimes A + B^T \otimes I_n\right) vec(X) = vec(C), \tag{1.4}$$

where $\otimes$ denotes the Kronecker product operator, and $vec(X)$ denotes the vectorize operator defined as (in MATLAB notation)

$$vec(X) = [X(:,1); X(:,2); \ldots ; X(:,s)],$$

and $X(:,i)$ is the $i$th column of the matrix $X \in \mathbb{R}^{n\times s}$. The linear equation systems (1.4) have unique solution if and only if the matrix $(I_s \otimes A) + (B^T \otimes I_n)$ is nonsingular. Throughout this paper, we assume that the system (1.4) has a unique solution. However, the size of the linear equation systems (1.4) would be very huge. Therefore, we apply some iterative algorithms for solving (1.1) instead of (1.4).

If $B$ is small, say $s \le 1000$, then the solution of (1.1) reduces to that of some (complex) shifted linear systems [39]. However, this relies on the spectral decomposition or Schur decomposition of $B$, as well as directly or iteratively solving $s$ large linear systems with respect to $A$. Projection methods are popular techniques for solving the Sylvester matrix equations with large $A$ and small $B$ [39]. There are some algorithms based on the block or the matrix Krylov solvers for the solution of the Sylvester matrix equations (see, e.g. [1, 8, 13, 15, 17, 19, 21, 26, 32, 35, 38, 39], and the references therein). The main idea behind these algorithms is to exploit the global or block Arnoldi process to construct $F$-orthonormal or orthonormal bases for the matrix or block Krylov subspaces, respectively, and then apply some orthogonal or oblique projection techniques to extract approximations. Some preconditioned global Krylov subspace methods were investigated in [4]. An alternative choice that has recently shown great potential compared with the block Krylov subspace methods is the extended Krylov subspace methods [15, 37]. In this type of methods, both the block Krylov subspaces with respect to $A$ and $A^{-1}$ are included.

With various computationally effective refinements, the ADI iteration [3] has been one of the leading methods for solving large-scale Sylvester equations [39]. However, the ADI iteration often applies to large-scale Sylvester equations with *large dimension $A$ and large dimension $B$*, and the computation involves solving some shifted linear systems with respect to both $A$ and $B$. For general dense matrices $A$ and $B$, this will cost us $\mathcal{O}(n^3)$ flops for a shifted system with $A$ and $\mathcal{O}(s^3)$ flops for shifted system with $B$. Thus, it is preferable to use the Bartels-Stewart algorithm [2] or the Hessenberg-Schur algorithm [3] in the case that $A$ is large dimension and $B$ is small dimension.

In [11], Essai introduced a weighted Arnoldi process for solving large linear systems. The idea is to improve and accelerate the convergence rate of the standard algorithm by constructing a $D$-orthonormal basis for the Krylov subspace, where $D$ is called the weighting matrix and is generally a positive diagonal matrix. According to [10, 14], weighting strategy can improve the algorithm by deflating the eigenvalues that hinder the convergence. The weighting strategy has been successfully developed for solving linear systems [16, 36], matrix equations [31, 33], and large eigenvalue problems [45]. For example, Mohseni Moghadam et al. [31] presented a weighted global FOM method for solving nonsymmetric linear systems. They used the Schur complement formula and a new matrix product, and gave some theoretical results to show rationality of the proposed algorithm. In [33], Panjeh Ali Beik et al. proposed weighted global FOM and weighted global GMRES algorithms for solving the general coupled linear matrix equations.

For the sake of the growth of memory requirements and computational cost, the global Krylov subspace algorithms will become impractical as the step of the global Arnoldi process proceeds. One remedy is to use some restarting strategies [23, 24]. For Krylov subspace algorithms, a popular restarting strategy is the deflated restarting (also referred to as thick-restarting or deflation) strategy advocated in [20, 28–30, 43–45], in which the approximate eigenvectors are put firstly in the search subspace. Here, “deflated restarting” (or deflation) means computing some approximate eigenvectors corresponding to some eigenvalues, and using them to “deflate” these eigenvalues from the spectrum of the matrix, to speed up the convergence of the iterative algorithm. In [23], Lin proposed two minimal residual methods augmented with approximate eigenvectors for solving large Sylvester matrix equations and generalized Sylvester matrix equations. The search subspace for extracting the approximation is the Kronecker product subspace of two augmented Krylov subspaces. However, this method is not based on the global GMRES framework, and the approximate eigenvectors are appended to the tail of the search subspace. As a comparison, in our approach, the approximate eigenvectors are put firstly in the search subspace. So, our new algorithm is simpler and cheaper than the one given in [23]. Implicitly restarted global FOM and GMRES algorithms for Sylvester equations were introduced in [24], in which the implicitly restarting strategy was utilized [40].

The deflation strategy is popular for the solution of large linear systems [28, 29] and large eigenvalue problems [20, 30, 43–45]; to the best of our knowledge, however, little work is done on applying the deflated restarting strategy on the (weighted) global GMRES algorithm for large Sylvester matrix equations. Moreover, as was pointed out in [10, 11, 14], the optimal choice of the weighting matrix $D$ in the weighted approaches is still an open problem and needs further investigation. In this work, we first apply the weighting strategy to the global GMRES algorithm, and present three new schemes to update the weighting matrix at each restart. To accelerate the convergence of the weighted global GMRES algorithm, we consider how to knit the deflation strategy together with it, and the key is that the Sylvester matrix equation can be rewritten as a linear system of the form (1.3) theoretically. The new algorithm can be understood as applying the deflation technique to remove some small eigenvalues of the matrix $(I_s \otimes A + B^T \otimes I_n)$ at each restart.

This paper is organized as follows. After presenting the weighted global GMRES algorithm for the solution of Sylvester matrix equations in Section 2, the deflated version of this algorithm is proposed in Section 3. Some numerical experiments confirm the superiority of our new algorithm over the conventional ones are given in Section 4. Throughout this paper, we use the following notations. We define the inner product $\langle X, Y\rangle_F = trace(X^T Y)$. Let $A = [A_1, \ldots, A_p] \in \mathbb{R}^{n\times ps}$ and $B = [B_1, \ldots, B_l] \in \mathbb{R}^{n\times ls}$, where $A_j$ and $B_j$ are $n \times s$ matrices. The matrix $A^T \diamond B$ is defined by $(A^T \diamond B)_{ij} =< A_i, B_j >_F$. We denote by $(\cdot,\cdot)_D$ the $D$-inner product, and "$\perp_D$" means orthgonal with respect to the $D$-inner product. Denote by $|R_m|$ the absolute value of the matrix $R_m$, by $I_s$ the identity matrix of order $s$, and by $\mathbf{e}_m$ the $m$-th column of an identity matrix. Let $I$ and $\mathbf{0}$ be the identity matrix and the zero matrix whose order are clear from the context.

## 2 A weighted global GMRES algorithm for large Sylvester matrix equations

In this section, we recall some notations and definitions that will be used in this paper, and briefly introduce the weighted global Arnoldi process as well as the weighted global GMRES algorithm. We will propose three new schemes based on residual to update the weighting matrix during iterations.

The global generalized minimal residual (GLGMRES) algorithm is well-known for solving linear systems with multiple right-hand sides and for matrix equations [18, 32], which is an oblique projection method based on matrix Krylov subspace. Let us introduce the *weighted* global GMRES algorithm for Sylvester matrix equations. Let $D = diag(d_1, d_2, \ldots, d_n)$ be a diagonal matrix with $d_i > 0,\ i = 1, 2, \ldots, n$, and let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ be given, then the $D$-inner product with respect to two vectors is defined as [45]

$$(\mathbf{u}, \mathbf{v})_D = \mathbf{v}^T D\mathbf{u} = \textstyle\sum_{i=1}^n d_i \mathbf{u}_i \mathbf{v}_j,$$

and the associated $D$-norm of $\mathbf{u}$ is defined as

$$\|\mathbf{u}\|_D = \sqrt{(\mathbf{u}, \mathbf{u})_D}, \quad \forall \mathbf{u} \in \mathbb{R}^n.$$

For two matrices $Y, Z \in \mathbb{R}^{n\times s}$, the $D$-inner product is defined as [31]

$$(Y, Z)_D = trace(Z^T DY),$$

where $trace(\cdot)$ denotes the trace of a matrix, and $Z^T$ represents the transpose of the matrix $Z$. It can be verified that [31]

$$(Y, Z)_D = (vec(Y), vec(Z))_{I_s\otimes D}\,.$$

Also, the $D$-norm associated with this inner product is

$$\|Y\|_D = \sqrt{(Y, Y)_D}, \quad \forall\, Y \in \mathbb{R}^{n\times n}.$$

Next, we introduce a useful product that will be used latter:

**Definition 2.1** [31] Let $A = [A_1, A_2, \ldots, A_p]$ and $B = [B_1, B_2, \ldots, B_l]$, where $A_i, B_j \in \mathbb{R}^{n\times s}$, $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, l$. Then, elements of the matrix $A^T \diamond_D B$ are defined as

$$(A^T \diamond_D B)_{ij} = (A_i, B_j)_D, \quad i = 1, 2, \ldots, p, \; j = 1, 2, \ldots, l. \tag{2.1}$$

It was shown that $A^T \diamond_D B = A^T \diamond (DB)$ [31]. Let $V_1 \in \mathbb{R}^{n\times s}$ be an initial block vector that is $D$-orthogonal, that is, orthonormal with respect to the $D$-inner product. The following algorithm presents an $m$-step weighted global Arnoldi process:

**Algorithm 1** [33] The $m$-step weighted global Arnoldi process

```
1.  Given $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{s\times s}$, $V_1 \in \mathbb{R}^{n\times s}$, a positive diagonal matrix $D$ and an
    integer number $m > 0$.
2.  Compute $\beta = \|V_1\|_D$, $V_1 = V_1/\beta$.
3.  for $j = 1, 2, \ldots, m$
4.    Compute $W = \mathcal{A}V_j = AV_j + V_jB$.
5.    for $i = 1, 2, \ldots, j$
6.      $h_{ij} = (W, V_i)_D$.
7.      $W = W - h_{ij}V_i$.
8.    end
9.    Compute $h_{j+1,j} = \|W\|_D$. If $h_{j+1,j} = 0$ then stop.
10.   Compute $V_{j+1} = W/h_{j+1,j}$.
11.   end
```

The weighted global Arnoldi process constructs a $D$-orthonormal basis $\mathcal{V}_m = [V_1, V_2, \ldots, V_m] \in \mathbb{R}^{n\times ms}$, i.e.,

$$(V_i, V_j)_D = \delta_{ij}, \quad \forall i, j = 1, 2, \ldots, m,$$

for the matrix Krylov subspace

$$\begin{aligned}\mathcal{K}_m(\mathcal{A}, V_1) &:= span\{V_1, \mathcal{A}V_1, \ldots, \mathcal{A}^{m-1}V_1\}\\ &= \left\{\sum_{i=1}^{m-1} \alpha_i \mathcal{A}^{i-1}V_1 | \alpha_i \in \mathbb{R}, i = 1, 2, \ldots, m-1\right\}.\end{aligned}$$

Let $\bar{H}_m = \begin{bmatrix} H_m \\ h_{m+1,m}\mathbf{e}_m^T \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}$ be a quasi-upper Hessenberg matrix whose nonzero entries $h_{ij}$ are from Algorithm 1, and the matrix $H_m$ is obtained from the matrix $\bar{H}_m$ by deleting its last row. Note that the matrix $\mathcal{V}_m$ is $D$-orthogonal. With the help of Definition 2.1, we get the following relations:

$$\begin{aligned}[]
[\mathcal{A}V_1, \mathcal{A}V_2, \ldots, \mathcal{A}V_m] &= \mathcal{V}_m(H_m \otimes I_s) + h_{m+1,m}V_{m+1}(\mathbf{e}_m^T \otimes I_s) \qquad (2.2)\\
&= \mathcal{V}_{m+1}(\bar{H}_m \otimes I_s), \qquad (2.3)\\
\bar{H}_m &= \mathcal{V}_{m+1}^T \diamond_D \mathcal{A}\mathcal{V}_m, \qquad (2.4)
\end{aligned}$$

where $\mathcal{V}_{m+1} = [V_1, V_2, \ldots, V_{m+1}] \in \mathbb{R}^{n\times(m+1)s}$. Define

$$\mathcal{A}\mathcal{V}_m \equiv [\mathcal{A}V_1, \mathcal{A}V_2, \ldots, \mathcal{A}V_m], \qquad (2.5)$$

then (2.2) can be rewritten as

$$\mathcal{A}\mathcal{V}_m = \mathcal{V}_{m+1}(\bar{H}_m \otimes I_s). \qquad (2.6)$$

We are in a position to consider the weighted global GMRES algorithm for solving (1.1). Let $X_0 \in \mathbb{R}^{n\times s}$ be the initial guess, and the initial residual be $R_0 = C - AX_0 - X_0B$. In the weighted global GMRES algorithm, we construct an approximate solution of the form

$$X_m = X_0 + \mathcal{V}_m(\mathbf{y}_m^w \otimes I_s), \qquad (2.7)$$

where $\mathbf{y}_m^w \in \mathbb{R}^m$. The corresponding residual is

$$\begin{aligned}
R_m &= C - AX_m - X_mB\\
&= (C - AX_0 - X_0B) - \big(A\mathcal{V}_m(\mathbf{y}_m^w \otimes I_s) + \mathcal{V}_m(\mathbf{y}_m^w \otimes I_s)B\big)\\
&= R_0 - \mathcal{A}\mathcal{V}_m(\mathbf{y}_m^w \otimes I_s), \qquad (2.8)
\end{aligned}$$

here we used (1.2) and (2.5). Substituting (2.3) into (2.8), we arrive at

$$\begin{aligned}
R_m &= \beta V_1 - \mathcal{V}_{m+1}(\bar{H}_m \otimes I_s)(\mathbf{y}_m^w \otimes I_s)\\
&= \mathcal{V}_{m+1}\left((\beta\mathbf{e}_1 - \bar{H}_m\mathbf{y}_m^w) \otimes I_s\right), \qquad (2.9)
\end{aligned}$$

where $\mathbf{e}_1$ is the first canonical basis vector in $\mathbb{R}^{m+1}$. Note that the residual is $D$-orthogonal to $\mathcal{A}\mathcal{K}_m(\mathcal{A}, R_0)$, i.e.,

$$R_m = C - \mathcal{A}X_m \perp_D \mathcal{A}\mathcal{K}_m(\mathcal{A}, R_0), \qquad (2.10)$$

where $\mathcal{A}\mathcal{K}_m(\mathcal{A}, R_0) = span\{\mathcal{A}R_0, \ldots, \mathcal{A}^mR_0\}$, and "$\perp_D$" means orthogonal with respect to the "$\diamond_D$" inner product.

In order to compute $\mathbf{y}_m^w$, we have from (2.9) and (2.10) that

$$\begin{aligned}
\min_{\mathbf{y}} \|R_m\|_D &= \min_{\mathbf{y}} \|\mathcal{V}_{m+1}((\beta\mathbf{e}_1 - \bar{H}_m\mathbf{y}) \otimes I_s)\|_D\\
&= \min_{\mathbf{y}} \|(\beta\mathbf{e}_1 - \bar{H}_m\mathbf{y})^T(\mathcal{V}_{m+1})^TD(\mathcal{V}_{m+1})(\beta\mathbf{e}_1 - \bar{H}_m\mathbf{y})\|_2\\
&= \min_{\mathbf{y}} \|\beta\mathbf{e}_1 - \bar{H}_m\mathbf{y}\|_2,
\end{aligned}$$

where we used $\mathcal{V}_{m+1}^T \diamond_D \mathcal{V}_{m+1} = I_{m+1}$. Thus,

$$\mathbf{y}_m^w = \arg\min_{\mathbf{y}^w \in \mathbb{R}^m} \|\beta\mathbf{e}_1 - \bar{H}_m\mathbf{y}^w\|_2, \qquad (2.11)$$

or equivalently,

$$\bar{H}_m^T \bar{H}_m \mathbf{y}_m^w = \bar{H}_m^T \beta \mathbf{e}_1. \tag{2.12}$$

As was pointed out in [11, 16, 45], the optimal choice of $D$ in the weighted approaches is still an open problem and needs further investigation. Some choices for the weighting matrix have been considered in [16, 33, 36]. Also, to speed up the convergence rate, it was suggested to use a weighted inner product that changes at each restart [10, 14]. In this section, we propose three choices based on the residual $R_m$, which could be updated during iterations:

**Option 1:** Let $\|R(:,t_1)\| = \max_{1\le i\le s}\{\|R_m(:,i)\|_2\}$, where $R_m(:,t_1)$ is the $t_1$-th column of the residual matrix $R_m$ with the maximum norm. Then we define $D_1 = diag\left(\frac{|R(:,t_1)|}{\|R(:,t_1)\|_2}\right)$, where $|R(:,t_1)|$ stands for the absolute value of $R(:,t_1)$.

**Option 2:** Let $\|R(:,t_2)\|_2 = \min_{1\le i\le s}\{\|R_m(:,i)\|\}$, where $R_m(:,t_2)$ is the $t_2$-th column of the residual matrix $R_m$ with the minimum norm. Then we define $D_2 = diag\left(\frac{|R(:,t_2)|}{\|R(:,t)\|_2}\right)$, and $|R(:,t_2)|$ stands for the absolute value of $R(:,t_2)$.

**Option 3:** Use the mean of the absolute value of the residual matrix $R_m$: $D_3 = diag\left(\frac{\sum_{i=1}^{s}|R_m(:,i)|}{s}\right)$.

Combining these weighting strategies with Algorithm 1, we propose the first new algorithm in this work.

**Algorithm 2** A restarted weighted global GMRES algorithm for large Sylvester matrix equations (W-GLGMRES)

```
1. Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{s\times s}$, $C \in \mathbb{R}^{n\times s}$. Choose the initial guess $X_0 \in \mathbb{R}^{n\times s}$,
   an initial positive diagonal matrix $D$ and an integer $m > 0$, and the tolerance
   $tol > 0$.
2. Output: The approximation $X_m$.
3. Compute $R_0 = C - AX_0 - X_0B$ and $\beta = \|R_0\|_D$, $V_1 = R_0/\beta$.
4. Run Algorithm 1 with the initial block vector $V_1$ to build the matrices $\mathcal{V}_m$, $\bar{H}_m$.
5. Solving $\min_{\mathbf{y}^w\in\mathbb{R}^m} \|\beta e_1 - \bar{H}_m \mathbf{y}^w\|_2$ for $\mathbf{y}_m^w$.
6. Compute $X_m = X_0 + \mathcal{V}_m(\mathbf{y}_m^w \otimes I_s)$ and $R_m = C - AX_m - X_mB$. If $\|R_m\|_D < tol$,
   then stop, else continue.
7. Set $X_0 = X_m$ and update the weighting matrix $D$ according to Options 1–3, and
   go to Step 3.
```

Theoretically, the Sylvester matrix (1.1) can be reformulated as the linear system (1.4). It is known that the convergence of the restarted GMRES method can be significantly improved, for some problems, by using a weighted inner product that changes at each restart [10, 11, 14]. Specifically, it was shown that weighted inner products can help in two distinct ways [10]: when the coefficient matrix has localized eigenvectors, weighting can allow restarted GMRES to focus on eigenvalues that otherwise slow convergence; for general problems, weighting can break the cyclic convergence pattern into which restarted GMRES often settles. On the other hand, performing computations in a weighted inner product can be viewed as a (diagonally) preconditioned method (with $D$ or the $I \otimes D$ Kronecker product as preconditioner) in the standard Frobenius inner product [22]. It is well known that preconditioning strategy is very efficient for Krylov solvers, so it is necessary to improve performance of this preconditioned technique for large Sylvester equations.

A popular choice is to use weights determined by the residual vector obtained from the last cycle [10, 11, 14, 45]. However, one can only use the weighting matrix $D$ of size $n$ rather than of size $n \times s$ in the weighted global GMRES algorithm for Sylvester matrix equations. This is the reason why we just use a column vector from $\text{span}\{|R_m|\}$ instead of $R_m$ itself to construct the weighting matrix. Indeed, the three choices of $D_j$ $(j = 1, 2, 3)$ for the Sylvester matrix equation can be understood as the weighted GMRES algorithm with the weighting matrices

$$\widehat{D}_j = I_s \otimes D_j, \;\; j = 1, 2, 3,$$

respectively, for the linear system (1.4). We expect $D_1$ (or $D_2$) will work better as the residual vector in absolute value with largest (or smallest) norm is dominant. In general, we expect $D_3$ will work better as it can be viewed as the mean of all the residual vectors in absolute value, which contains the information in all the columns of $|R_m|$; see the numerical experiments made in Example 1.

## 3 Weighted global GMRES algorithm with deflation for large Sylvester matrix equations

In this section, we speed up the weighted global GMRES algorithm by using the deflated restarting strategy that is popular for large linear systems and large eigenvalue problems [20, 28–30, 44, 45]. In the first cycle of the weighted global GMRES algorithm with deflation, the standard weighted global GMRES algorithm is run. To apply the deflated restarting strategy, we need to compute $k$ $(1 \le k \le m)$ weighted harmonic Ritz pairs. Let $\mathcal{V}_m$ be the $D$-orthonormal basis obtained from the "previous" cycle, we seek $k$ weighted harmonic Ritz pairs $(\theta_i, Y_i)$ that satisfy

$$\mathcal{A}\mathcal{V}_m Y_i - \theta_i \mathcal{V}_m Y_i \perp_D (\mathcal{A} - \sigma I)\mathcal{K}_m(\mathcal{A}, V), \quad i = 1, \ldots, k, \tag{3.1}$$

where

$$\mathcal{A}\mathcal{V}_m = [\mathcal{A}V_1, \mathcal{A}V_2, \ldots, \mathcal{A}V_m] = [AV_1 + V_1 B, \ldots, AV_m + V_m B],$$

and $Y_i = \mathbf{g}_i \otimes I_s \in \mathbb{C}^{ms \times s}$ with $\mathbf{g}_i \in \mathbb{C}^{m \times 1}$. In this work, we want to deflate some smallest eigenvalues in magnitude, and a shift $\sigma = 0$ is used throughout this paper.

From (2.6), we have

$$\begin{aligned} \mathcal{A}\mathcal{V}_m(\mathbf{g}_i \otimes I_s) - \theta_i \mathcal{V}_m(\mathbf{g}_i \otimes I_s) &= (\mathcal{A}\mathcal{V}_m - \theta_i \mathcal{V}_m)(\mathbf{g}_i \otimes I_s) \\ &= \mathcal{V}_{m+1}\left((\bar{H}_m - \theta_i \bar{I}_m) \otimes I_s\right)(\mathbf{g}_i \otimes I_s), \end{aligned}$$

where $\bar{I}_m = \begin{bmatrix} I_m \\ 0 \end{bmatrix}$. By (3.1) and Definition 2.1, we compute $(\theta_i, \mathbf{g}_i)$ via solving the following (small-sized) generalized eigenvalue problem

$$\theta_i \left( (\mathcal{A}\mathcal{V}_m)^T \diamond_D \mathcal{V}_m \right) \mathbf{g}_i = (\mathcal{A}\mathcal{V}_m)^T \diamond_D (\mathcal{A}\mathcal{V}_m)\mathbf{g}_i. \tag{3.2}$$

From (2.6) and the fact that $\mathcal{V}_{m+1}^T \diamond_D \mathcal{V}_{m+1} = I_{m+1}$, we rewrite (3.2) as

$$\theta_i H_m^T \mathbf{g}_i = \bar{H}_m^T \bar{H}_m \mathbf{g}_i. \tag{3.3}$$

If $H_m$ is nonsingular, (3.3) is equivalent to

$$(H_m + h_{m+1,m}^2 H_m^{-T} \mathbf{e}_m \mathbf{e}_m^T)\mathbf{g}_i = \theta_i \mathbf{g}_i. \tag{3.4}$$

Then, we define the "weighted harmonic Ritz vector" as $Y_i = \mathbf{g}_i \otimes I_s$, and the corresponding harmonic residual is

$$\begin{aligned} \widetilde{R}_i = \mathcal{A}\mathcal{V}_m Y_i - \theta_i \mathcal{V}_m Y_i &= \mathcal{V}_{m+1} \left( (\bar{H}_m - \theta_i \bar{I}_m)\mathbf{g}_i \otimes I_s \right) \\ &= \mathcal{V}_{m+1}(\widetilde{\boldsymbol{r}}_i \otimes I_s), \quad i = 1, \ldots, k, \end{aligned}$$

where $\widetilde{\boldsymbol{r}}_i = (\bar{H}_m - \theta_i \bar{I}_m)\mathbf{g}_i, \ i = 1, \ldots, k$.

*Remark 3.1* In [9], a global harmonic Arnoldi method was proposed for computing harmonic Ritz pairs of large matrices. The difference is that our approach is based on the weighted projection techniques, i.e., our work is an extension of the work in [9] to the case of $D$ other than the identity matrix $I$.

In the following, we characterize the relationship between the weighted harmonic residuals and the residual from the weighted global GMRES algorithm. Let $X_0 \in \mathbb{R}^{n\times s}$ be the initial guess and $R_0 = C - AX_0 - X_0 B$ be the initial residual. After one cycle of the weighted global GMRES algorithm, we have the approximate solution $X_m = X_0 + \mathcal{V}_m(\mathbf{y}_m^w \otimes I_s)$, where $\mathbf{y}_m^w$ is defined in (2.11). The associated residual with respect to $X_m$ is

$$R_m = \mathcal{V}_{m+1} \left( (\beta \mathbf{e}_1 - \bar{H}_m \mathbf{y}_m^w) \otimes I_s \right) = \mathcal{V}_{m+1}(\mathbf{r}_m \otimes I_s),$$

where $\mathbf{r}_m \equiv \beta \mathbf{e}_1 - \bar{H}_m \mathbf{y}_m^w$. The following result shows that $\mathbf{r}_m$ and $\{\widetilde{\boldsymbol{r}}_i\}_{i=1}^k$ are collinear with each other.

**Theorem 3.1** *Let $\widetilde{R}_i = \mathcal{V}_{m+1}(\widetilde{\boldsymbol{r}}_i \otimes I_s), \ i = 1, \ldots, k$, be the weighted harmonic residuals and $R_m = \mathcal{V}_{m+1}(\boldsymbol{r}_m \otimes I_s)$ be the weighted global GMRES residual, respectively, where $\widetilde{\boldsymbol{r}}_i = (\bar{H}_m - \theta_i \bar{I}_m)\mathbf{g}_i$ and $\boldsymbol{r}_m = \beta \boldsymbol{e}_1 - \bar{H}_m \mathbf{y}_m^w$. Then $\{\widetilde{\boldsymbol{r}}_i\}_{i=1}^k$ and $\boldsymbol{r}_m$ are collinear with each other.*

*Proof* Note that both the weighted harmonic residuals and the residual of the weighted global GMRES algorithm are in $range(\mathcal{V}_{m+1})$, and they are both $D$-orthogonal to $\mathcal{A}\mathcal{V}_m$. Thus, there is an $s \times s$ matrix $T$, such that

$$\mathcal{V}_{m+1}(\widetilde{\boldsymbol{r}}_i \otimes I_s) = \mathcal{V}_{m+1}(\mathbf{r}_m \otimes I_s)T, \tag{3.5}$$

so we have $(\widetilde{\boldsymbol{r}}_i \otimes I_s) = (\mathbf{r}_m \otimes I_s)T$. Let $\mathbf{r}_m = (\eta_1, \eta_2, \ldots, \eta_{m+1})^T$ and $\widetilde{\boldsymbol{r}}_i = (\tau_1, \tau_2, \ldots, \tau_{m+1})^T$, we obtain from (3.5) that

$$(\mathbf{r}_m \otimes I_s)T = \begin{pmatrix} \eta_1 I_s \\ \eta_2 I_s \\ \vdots \\ \eta_{m+1} I_s \end{pmatrix} T = \begin{pmatrix} \eta_1 T \\ \eta_2 T \\ \vdots \\ \eta_{m+1} T \end{pmatrix} = \begin{pmatrix} \tau_1 I_s \\ \tau_2 I_s \\ \vdots \\ \tau_{m+1} I_s \end{pmatrix},$$

implying that $\{\widetilde{\boldsymbol{r}}_i\}_{i=1}^k$ and $\mathbf{r}_m$ are collinear with each other. □

We are ready to consider how to apply the deflated restarting strategy to the weighted global GMRES algorithm, and show that a weighted global Arnoldi-like relation still holds after restarting. Let $G_k = [\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k]$, and let $G_k = Q_k \Gamma_k$ be the reduced QR factorization. We stress that both forming $G_k$ and computing the QR decomposition can be implemented in real arithmetics; see Step 9 of Algorithm 3. Then, we orthonormalize $\mathbf{r}_m$ against $\begin{bmatrix} Q_k \\ 0_{1\times k} \end{bmatrix}$ to get $\mathbf{q}_{k+1}$, and let $Q_{k+1} = \left[\begin{array}{c|c} Q_k & \mathbf{q}_{k+1} \\ 0_{1\times k} & \end{array}\right]$.

Notice that both (2.2) and (2.3) hold in the first cycle, i.e.,

$$\mathcal{A}\mathcal{V}_m = \mathcal{V}_m(H_m \otimes I_s) + h_{m+1,m} V_{m+1}(\mathbf{e}_m^T \otimes I_s),$$

and thus

$$\mathcal{A}\mathcal{V}_m(\mathbf{g}_i \otimes I_s) = \mathcal{V}_m(H_m \otimes I_s)(\mathbf{g}_i \otimes I_s) + h_{m+1,m} V_{m+1}(\mathbf{e}_m^T \mathbf{g}_i \otimes I_s),\ i = 1, \ldots, k. \quad (3.6)$$

That is,

$$\mathcal{A}\mathcal{V}_m(\mathbf{g}_i \otimes I_s) = [\mathcal{V}_m\ V_{m+1}] \begin{bmatrix} H_m \mathbf{g}_i \\ h_{m+1,m} \mathbf{e}_m^T \mathbf{g}_i \end{bmatrix} \otimes I_s.$$

We note that

$$\widetilde{\boldsymbol{r}}_i = (\bar{H}_m - \theta_i \bar{I}_m)\mathbf{g}_i = \begin{bmatrix} (H_m - \theta_i I)\mathbf{g}_i \\ h_{m+1,m} \mathbf{e}_m^T \mathbf{g}_i \end{bmatrix},$$

and

$$\begin{aligned} \begin{bmatrix} H_m \mathbf{g}_i \\ h_{m+1,m} \mathbf{e}_m^T \mathbf{g}_i \end{bmatrix} &= \begin{bmatrix} H_m \mathbf{g}_i - \theta_i \mathbf{g}_i + \theta_i \mathbf{g}_i \\ h_{m+1,m} \mathbf{e}_m^T \mathbf{g}_i \end{bmatrix} \\ &= \theta_i \begin{bmatrix} \mathbf{g}_i \\ 0 \end{bmatrix} + \widetilde{\boldsymbol{r}}_i, \quad i = 1, \ldots, k. \end{aligned}$$

As a result,

$$\begin{bmatrix} H_m \mathbf{g}_i \\ h_{m+1,m} \mathbf{e}_m^T \mathbf{g}_i \end{bmatrix} \in span\left\{ \begin{bmatrix} G_k \\ 0 \end{bmatrix}, \mathbf{r}_m \right\} = span\{Q_{k+1}\},\ i = 1, \ldots, k,$$

where we used the fact that $\{\widetilde{\boldsymbol{r}}_i\}_{i=1}^k$ and $\mathbf{r}_m$ are collinear with each other; see Theorem 3.1. Therefore,

$$\mathcal{A}\mathcal{V}_m(\mathbf{g}_i \otimes I_s) \in span\{\mathcal{V}_{m+1}(Q_{k+1} \otimes I_s)\},\ \ i = 1, \ldots, k,$$

and

$$\mathcal{A}\mathcal{V}_m(Q_k \otimes I_s) \subseteq span\{\mathcal{V}_{m+1}(Q_{k+1} \otimes I_s)\}. \tag{3.7}$$

Define $\mathcal{V}_k^{new} \equiv \mathcal{V}_m(Q_k \otimes I_s) = [V_1^{new}, \dots, V_k^{new}]$, where $V_i^{new} \in \mathbb{R}^{n\times s},\ 1 \le i \le k$, so we have

$$\mathcal{A}\mathcal{V}_m(Q_k \otimes I_s) = \mathcal{A}\mathcal{V}_k^{new} = \mathcal{A}[V_1^{new}, \dots, V_k^{new}].$$

If we denote

$$\mathcal{V}_{k+1}^{new} = \mathcal{V}_{m+1}(Q_{k+1} \otimes I_s),$$

then, we have from (3.7) that

$$\mathcal{A}\mathcal{V}_m(Q_k \otimes I_s) \subseteq span\{\mathcal{V}_{k+1}^{new}\},$$

and there is a $(k+1)s \times ks$ matrix $P$ such that

$$\mathcal{A}\mathcal{V}_m(Q_k \otimes I_s) = \mathcal{V}_{m+1}(\bar{H}_m \otimes I_s)(Q_k \otimes I_s) = \mathcal{V}_{m+1}(Q_{k+1} \otimes I_s)P = \mathcal{V}_{k+1}^{new}P.$$

The condition $\mathcal{V}_{k+1}^{new^T} \diamond_D \mathcal{V}_{k+1}^{new} = I_{k+1}$ yields

$$\begin{aligned} P &= (Q_{k+1} \otimes I_s)^T \mathcal{V}_{m+1}^T \diamond_D \mathcal{A}\mathcal{V}_m(Q_k \otimes I_s) \\ &= (Q_{k+1} \otimes I_s)^T \mathcal{V}_{m+1}^T \diamond_D \mathcal{V}_{m+1}(\bar{H}_m \otimes I_s)(Q_k \otimes I_s) \\ &= Q_{k+1}^T \bar{H}_m Q_k \otimes I_s = \bar{H}_k^{new} \otimes I_s, \end{aligned}$$

where $\bar{H}_k^{new} \equiv Q_{k+1}^T \bar{H}_m Q_k \in \mathbb{R}^{(k+1)\times k}$ is generally a dense matrix. In conclusion, we obtain

$$\mathcal{A}\mathcal{V}_k^{new} = \mathcal{V}_{k+1}^{new}(\bar{H}_k^{new} \otimes I_s). \tag{3.8}$$

Next, setting $\mathcal{V}_k = \mathcal{V}_k^{new}$, $\mathcal{V}_{k+1} = \mathcal{V}_{k+1}^{new}$ and $\bar{H}_k = \bar{H}_k^{new}$, respectively, then (3.8) can be written in the following relation

$$\mathcal{A}\mathcal{V}_k = \mathcal{V}_{k+1}(\bar{H}_k^{new} \otimes I_s). \tag{3.9}$$

Then, update $D$ according to options 1–3, where it is denoted by $D^{new}$. In the weighted GMRES-DR presented in [10], Embree et al. showed how to restart the weighted GMRES-DR algorithm with a change of inner product by using the Cholesky factorization, which is based on Proposition 1 of [11]. Based on this idea, we want to transform (3.9) to a new relation such that $\mathcal{V}_{k+1}$ is $D^{new}$-orthogonal. Decompose

$\mathcal{V}_{k+1}^{new^T} \diamond_{D^{new}} \mathcal{V}_{k+1}^{new} = L^T L,$

say, by using the Cholesky factorization, where $L$ is the lower triangular matrix. Next, we set

$$\begin{aligned} \bar{H}_k^{new} &:= L\bar{H}_k L_k^{-1}, \\ \mathcal{V}_k^{new} &:= \mathcal{V}_k(L^{-1} \otimes I_s), \\ \mathcal{V}_{k+1}^{new} &:= \mathcal{V}_{k+1}(L^{-1} \otimes I_s), \end{aligned}$$

where $L_k$ is the $k \times k$ leading submatrix of $L$. Thus, $\mathcal{V}_{k+1}^{new^T} \diamond_{D^{new}} \mathcal{V}_{k+1}^{new} = I_{k+1}$.

In the sequel, we run the new weighted global Arnoldi algorithm from index $k+1$ to $m$ with the last s columns of $\mathcal{V}_{k+1}^{new}$ as the starting matrix to build a new m-step global Arnoldi relation, i.e.,

$$\mathcal{A}\mathcal{V}_m^{new} = \mathcal{V}_{m+1}^{new}(\bar{H}_m^{new} \otimes I_s), \tag{3.10}$$

where $(\mathcal{V}_{m+1}^{new})^T \diamond_{D^{new}} \mathcal{V}_{m+1}^{new} = I_{m+1}$. Denote $\mathcal{V}_{m+1} = \mathcal{V}_{m+1}^{new}$, $\mathcal{V}_m = \mathcal{V}_m^{new}$ and $\bar{H}_m = \bar{H}_m^{new}$. In summary, we have the following theorem:

**Theorem 3.2** *A global Arnoldi-like relation holds for the weighted global Arnoldi algorithm with deflation*

$$\begin{aligned} \mathcal{A}\mathcal{V}_m &= \mathcal{V}_m(H_m \otimes I_s) + h_{m+1,m}V_{m+1}(\boldsymbol{e}_m^T \otimes I_s) && (3.11)\\ &= \mathcal{V}_{m+1}(\bar{H}_m \otimes I_s). && (3.12)\end{aligned}$$

We are ready to present the second new algorithm in this work:

In particular, we pay special attention to the case of $D = I$, where the above algorithm reduces to the global GMRES algorithm with deflation. Indeed, after (3.8), we run the global Arnoldi algorithm [32] to update $\mathcal{V}_{k+1}^{new}$ and $\bar{H}_k^{new}$, and Theorem 3.2 still holds. Thus, we propose the third new algorithm in this work:

*Remark 3.2* Compared with Algorithm 3, the advantages of the above algorithm are that there is no need to perform $D$-inner products nor do Cholesky factorization during iterations. So, Algorithm 4 is much cheaper than Algorithm 3. Refer to Section 4 for numerical performance of this new algorithm.

## 4 Numerical experiments

In this section, we perform some numerical experiments to show the potential of our three new algorithms for solving large Sylvester matrix equations. All the numerical examples were performed using MATLAB R2013b on PC-Pentium(R) with CPU 2.66 GHz and 4.00 of RAM. In all the algorithms, we set $X_0 = 0_{n\times s}$ to be the initial guess. Except for Examples 3, 4, and 6, we choose $C = sprand(n, s, s)$ as the right-hand side matrix, where $sprand(n, s, s)$ is the MATLAB command generating a random, $n$-by-$s$, sparse matrix with approximately $s \times n \times s$ uniformly distributed nonzero entries. Moreover, we use the stopping criterion

$$\frac{\|\mathcal{V}_{m+1}((\beta \mathbf{e}_1 - \bar{H}_m \mathbf{y}_m^w) \otimes I_s)\|_D}{\|C\|_D} \le tol = 10^{-6}, \tag{4.1}$$

and all the algorithms will be stopped as soon as the maximal iteration number $maxit = 2500$ is reached. For all the algorithms, we consider comparisons in three aspects: the number of iterations (referred to `iter`), the runtime in terms of seconds (referred to `CPU`), and the "real relative" residual in terms of Frobenius norm (referred to `res.norm`) defined as

$$res.norm = \frac{\|C - AX_m - X_mB\|_F}{\|C\|_F}, \tag{4.2}$$

**Algorithm 3** A weighted global GMRES with deflation for large Sylvester matrix equations (W-GLGMRES-D)

```
1.  **Input:** $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{s\times s}$, $C \in \mathbb{R}^{n\times s}$. *Choose an initial guess* $X_0 \in \mathbb{R}^{n\times s}$, *a positive diagonal matrix* $D$, *the positive integer number* $m$, *and a convergence tolerance* $tol > 0$.
2.  **Output:** *The approximation* $X_m$.
3.  *Compute* $R_0 = C - AX_0 - X_0B$, $\beta = \|R_0\|_D$ and $V_1 = R_0/\beta$.
4.  *Set* $\mathbf{c} = [\beta; \mathbf{0}_{m\times 1}]$.
5.  *Run the weighted global Arnoldi algorithm to obtain* $\mathcal{V}_{m+1}$ and $\bar{H}_m$.
6.  *Solve* $\min_{\mathbf{y}^w} \|\mathbf{c} - \bar{H}_m\mathbf{y}^w\|_2$ for $\mathbf{y}^w_m$.
7.  *Compute* $X_m = X_0 + \mathcal{V}_m(\mathbf{y}^w_m \otimes I_s)$, and $R_m = C - AX_m - X_mB$.
8.  *If* $\|R_m\|_D < tol$, *then stop, else continue.*
9.  *Compute the* $k$ *eigenpairs* $(\theta_i, \boldsymbol{g}_i)$ *with the smallest magnitude from (3.3) or (3.4). Set* $G_k = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k]$: *We first separate the* $\{\boldsymbol{g}_i\}'s$ *into real and imaginary parts if they are complex, to form the columns of* $G_k \in \mathbb{R}^{m\times k}$. *Both the real and the imaginary parts need to be included. Then we compute the reduced QR factorization of* $G_k : G_k = Q_k\Gamma_k$, *where* $Q_k = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k]$. *Notice that both* $G_k$ *and* $Q_k$ *are real.*
10. *Extend the vectors* $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k$ *to length* $m+1$ *with zero entries, then orthonormalize the vector* $\boldsymbol{r}_m = \boldsymbol{c} - \bar{H}_m\boldsymbol{y}^w_m$ *against the columns of* $\begin{bmatrix} Q_k \\ \boldsymbol{0} \end{bmatrix}$ *to form* $\boldsymbol{q}_{k+1}$. *Set* $Q_{k+1} = \left[\begin{array}{c|c} Q_k & \mathbf{q}_{k+1} \\ \mathbf{0} & \end{array}\right]$.
11. *Compute* $\mathcal{V}^{new}_{k+1} = \mathcal{V}_{m+1}(Q_{k+1} \otimes I_s)$ *and* $\bar{H}^{new}_k = Q^T_{k+1}\bar{H}_mQ_k$, *note that* $\bar{H}^{new}_k$ *is generally a full matrix. Set* $\mathcal{V}_{k+1} = \mathcal{V}^{new}_{k+1}$ *and* $\bar{H}_k = \bar{H}^{new}_k$, *and update* $D$ *according to options 1–3.*
12. *Compute* $P = \mathcal{V}^{new^T}_{k+1} \diamond_D \mathcal{V}^{new}_{k+1}$ *and* $P = L^TL$, *where* $L$ *is* $(k+1)\times(k+1)$ *lower triangular matrix. Let* $L_k$ *denote the leading* $k \times k$ *submatrix of* $L$, *then update* $\mathcal{V}^{new}_{k+1} := \mathcal{V}_{k+1}(L^{-1} \otimes I_s)$, $\bar{H}^{new}_k = L\bar{H}_kL^{-1}_k$, *where* $L_k$ *is the k-by-k principal submatrix of L.*
13. *Run the weighted global Arnoldi algorithm from index* $k+1$ *to* $m$ *to obtain* $\mathcal{V}_{m+1}$ *and* $\bar{H}_m$, *where the* $(k+1)$*th block is the last* $s$ *columns of* $\mathcal{V}^{new}_{k+1}$.
14. *Set* $X_0 = X_m$, $R_0 = R_m$ *and* $\boldsymbol{c} = \mathcal{V}^T_{m+1} \diamond_D R_0$, *and go to Step 6.*
```

where $X_m$ are the computed solutions from the algorithms. In all the tables below, we denote by $m, k$ the number of global Arnoldi process and the number of harmonic Ritz block vectors added to the search subspace, respectively. And, we denote by GLGMRES the standard restarted global GMRES algorithm proposed in [32]. In all figures, the $X$-label shows the iteration number and the $Y$-label indicates the real relative residual.

*Example 1* In this example, we illustrate the numerical behavior of Algorithm 2 (W-GLGMRES) for different choices of $D$, and show efficiency of our three new

**Algorithm 4** A global GMRES with deflation for large Sylvester matrix equations (GLGMRES-D)

```
1. Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{s\times s}$, $C \in \mathbb{R}^{n\times s}$. Choose an initial guess $X_0 \in \mathbb{R}^{n\times s}$, the positive integer number $m$ and a convergence tolerance $tol > 0$.
2. Output: The approximation $X_m$.
3. Compute $R_0 = C - AX_0 - X_0B$, $\beta = \|R_0\|_F$ and $V_1 = R_0/\beta$.
4. Set $\mathbf{c} = [\beta; \mathbf{0}_{m\times 1}]$.
5. Run the global Arnoldi algorithm to obtain $\mathcal{V}_{m+1}$ and $\bar{H}_m$.
6. Solve $\min_{\mathbf{y}} \|\mathbf{c} - \bar{H}_m\mathbf{y}\|_2$ for $\mathbf{y}_m$.
7. Compute $X_m = X_0 + \mathcal{V}_m(\mathbf{y}_m \otimes I_s)$, and $R_m = C - AX_m - X_mB$.
8. If $\|R_m\|_F < tol$, then stop, else continue.
9. Compute the $k$ eigenpairs $(\theta_i, \boldsymbol{g}_i)$ with the smallest magnitude from (3.3) or (3.4). Set $G_k = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k]$: We first separate the $\{\boldsymbol{g}_i\}'s$ into real and imaginary parts if they are complex, to form the columns of $G_k \in \mathbb{R}^{m\times k}$. Both the real and the imaginary parts need to be included. Then, we compute the reduced QR factorization of $G_k$ : $G_k = Q_k\Gamma_k$, where $Q_k = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k]$. Notice that both $G_k$ and $Q_k$ are real.
10. Extend the vectors $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_k$ to length $m+1$ with zero entries, then orthonormalize the vector $\boldsymbol{r}_m = \boldsymbol{c} - \bar{H}_m\boldsymbol{y}_m$ against the columns of $\begin{bmatrix} Q_k \\ \boldsymbol{0} \end{bmatrix}$ to form $\boldsymbol{q}_{k+1}$. Set $Q_{k+1} = \left[\begin{array}{c|c} Q_k & \boldsymbol{q}_{k+1} \\ \boldsymbol{0} & \end{array}\right]$.
11. Compute $\mathcal{V}_{k+1}^{new} = \mathcal{V}_{m+1}(Q_{k+1} \otimes I_s)$ and $\bar{H}_k^{new} = Q_{k+1}^T\bar{H}_mQ_k$, note that $\bar{H}_k^{new}$ is generally a full matrix. Set $\mathcal{V}_{k+1} = \mathcal{V}_{k+1}^{new}$ and $\bar{H}_k = \bar{H}_k^{new}$.
12. Compute $P = \mathcal{V}_{k+1}^{new^T} \diamond \mathcal{V}_{k+1}^{new}$ and $P = L^TL$, where $L$ is $(k+1)\times(k+1)$ lower triangular matrix. Let $L_k$ denote the leading $k \times k$ submatrix of $L$, then update $\mathcal{V}_{k+1}^{new} := \mathcal{V}_{k+1}(L^{-1} \otimes I_s)$, $\bar{H}_k^{new} = L\bar{H}_kL_k^{-1}$, where $L_k$ is the principal submatrix of $L$.
13. Run the global Arnoldi algorithm from index $k+1$ to $m$ to obtain $\mathcal{V}_{m+1}$ and $\bar{H}_m$, where the $(k+1)$th block is the last $s$ columns of $\mathcal{V}_{k+1}^{new}$.
14. Set $X_0 = X_m$, $R_0 = R_m$ and $\mathbf{c} = \mathcal{V}_{m+1}^T \diamond R_0$, and go to Step 6.
```

weighting strategies proposed in Options 1–3. To this aim, we compare the performance of W-GLGMRES with the standard global GMRES algorithm (GLGMRES) [32].

The matrices $A$ and $B$ are obtained from the discretization of the operators [1]

$$L_i(u) = \Delta u - f_{1,i}(x,y)\frac{\partial u}{\partial x} - f_{2,i}(x,y)\frac{\partial u}{\partial y} - f_{3,i}(x,y)u, \quad i = 1, 2,$$

on the unit square $[0,1] \times [0,1]$ with homogeneous Dirichlet boundary conditions. In these operators, we have $f_{1,1}(x,y) = e^{x^2+y}$, $f_{1,2}(x,y) = 2xy$, $f_{2,1}(x,y) = \sin(x+2y)$, $f_{2,2}(x,y) = e^{xy}$, $f_{3,1}(x,y) = \cos(xy)$, and $f_{3,2}(x,y) = xy$. The dimensions of matrices $A$ and $B$ are $n = n_0^2$ and $s = s_0^2$, respectively. By using the command $fdm\_2d\_matrix$ from LYAPACK [34], we can extract the matrices $A = fdm(f_{1,1}, f_{2,1}, f_{3,1})$ and $B = fdm(f_{1,2}, f_{2,2}, f_{3,2})$.

We make use of three cases for $D$, i.e., $D_1$, $D_2$, and $D_3$, which are proposed in Options 1–3. Note that they could be updated during the cycles. Table 1 lists the numerical results for different choices of $m$, $s$, and $n$; and Fig. 1 plots the convergence curves of the algorithms for $n = 22,500$ and $40,000$ as $s = 16, m = 15$.

From Table 1 and Fig. 1, we observe that the three weighted GLGMRES algorithms need much fewer iterations and much less CPU time than the standard GLGMRES algorithm, and they reach about the same accuracy in terms of the "real" residual norm. More precisely, W-GLGMRES performs better than the standard GLGMRES algorithm, using $D_1$, $D_2$, or $D_3$ as the weighting matrix; and it works the best with $D_3$. So, we benefit from these preconditioning strategies. All these demonstrate that the WGLGMRES algorithm has the potential to improve the convergence, and also it is more robust and efficient than the standard global GMRES algorithm.

*Example 2* In this example, we compare our new weighting strategies with the ones proposed in [16, 36], and show the effectiveness of our new strategies. In [16], Heyouni et al. considered the linear equation $AX = C$, and proposed a weighted matrix $D$ with elements $D_{i,j} = \sqrt{ns}|C|/\|C\|_F$, where $|C|$ is the matrix with absolute values of $C$. They then introduced a weighted strategy as $(X, Y)_D = tr(X^T(D \circ Y))$, where $D \circ Y$ denotes the Hadamard product of $D$ and $Y$. In [36], Saberi Najafi et al. suggested choosing the weighting matrix $D$ as a diagonal random matrix whose diagonal elements are uniformly and randomly chosen from $(0, 2)$. In all the numerical examples from now on, we use $D_3$ as the representative for the three new weighting strategies in our new algorithms.

**Table 1** Example 1: Performance of W-GLGMRES with different choices of $D$

| | | | $n = 22500$ | | | $n = 40000$ | | |
|---|---|---|---|---|---|---|---|---|
| $s$ | $m$ | $D$ | iter | res.norm | CPU | iter | res.norm | CPU |
| | | $I$ | 271 | 9.6980e−07 | 1.7603e+03 | 502 | 9.8712e−07 | 3.6526e+03 |
| | | $D_1$ | 225 | 8.9007e−07 | 1.4532e+03 | 361 | 9.0881e−07 | 2.6656e+03 |
| 25 | 10 | $D_2$ | 196 | 8.9191e−07 | 1.2564e+03 | 375 | 9.9170e−07 | 2.7972e+03 |
| | | $D_3$ | 149 | 8.9245e−07 | 964.4450 | 247 | 9.2564e−07 | 1.8723e+03 |
| | | $I$ | 287 | 9.6370e−07 | 1.0593e+03 | 511 | 9.9187e−07 | 2.6007e+03 |
| | | $D_1$ | 224 | 8.6859e−07 | 819.1613 | 350 | 8.0319e−07 | 1.7281e+03 |
| 16 | 10 | $D_2$ | 221 | 9.5079e−07 | 828.4745 | 278 | 9.0354e−07 | 1.2168e+03 |
| | | $D_3$ | 147 | 8.9043e−07 | 540.1904 | 253 | 9.1782e−07 | 1.1863e+03 |
| | | $I$ | 135 | 9.9273e−07 | 1.0023e+03 | 229 | 9.4550e−07 | 2.5741e+03 |
| | | $D_1$ | 93 | 9.2973e−07 | 690.4734 | 164 | 9.6980e−07 | 1.8403e+03 |
| 16 | 15 | $D_2$ | 85 | 8.8796e−07 | 612.2571 | 138 | 9.1576e−07 | 1.5791e+03 |
| | | $D_3$ | 77 | 8.9375e−07 | 558.9338 | 125 | 8.8805e−07 | 1.4288e+03 |

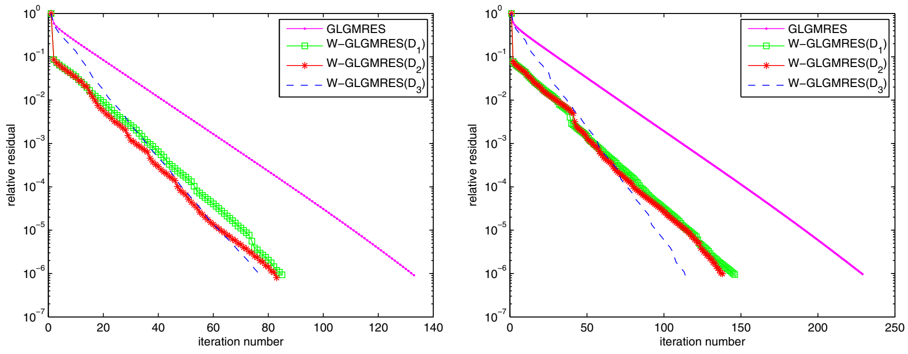


**Fig. 1** Example 1: Convergence curves of the W-GLGMRES algorithm and those of the standard GLGMRES algorithm. Left: $n = 22500, s = 16, m = 15$; Right: $n = 40,000, s = 16, m = 15$

The test matrices for Example 2–Example 5 are available from *the Matrix Market Collection* [27]. The names of these matrices, the size, the nonzero elements, and the type of the matrices are shown in Table 2. Table 3 lists the iteration numbers, CPU time, and residual norms of the approximations, obtained from running W-GLGMRES with three different weighting strategies. The results demonstrate that by using our weighting strategy $D_3$, the weighted global GMRES converges faster, and it needs fewer number of iterations and less CPU time than the other two strategies given in [16, 36]. More precisely, the "Hadamard product" strategy [16] is better than the "randomized" strategy [36] according to iteration numbers and CPU time, while our new strategy based on the residual works the best. However, we find that the "real residual" norm *res.norm* computed from the "Hadamard product" strategy [16] may be larger than the desired tolerance $tol = 10^{-6}$ in some cases, and it is obvious to see that our new strategy can cure this drawback very well. Indeed, the stopping criterion used is (4.1) in practical calculations, rather than (4.2). Figures 2 and 3 plot the convergence curves of the three algorithms. Again, they illustrate the effectiveness and efficiency of our new choice of weighting matrix.

**Table 2** Summary of the test matrices for Example 2–Example 5

| Matrix | $n$ | nnz | Type | Application area |
|---|---|---|---|---|
| saylr4 | 3564 | 22316 | Real unsymmetric | Oil reservoir modeling |
| saylr3 | 1000 | 375 | Real unsymmetric | Oil reservoir modeling |
| add32 | 4960 | 19848 | Real unsymmetric | Electronic circuit design |
| sherman1 | 1000 | 3750 | Real symmetric | Oil reservoir modeling |
| sherman4 | 1104 | 3786 | Real unsymmetric | Oil reservior modeling |
| sherman5 | 3312 | 20793 | Real unsymmetric | Oil reservoir modeling |
| cavity15 | 2597 | 76367 | Real unsymmetric | Finite element modeling |
| rdb1250l | 1250 | 7300 | Real unsymmetric | Chemical engineering |
| pde2961 | 2961 | 14585 | Real unsymmetric | Partial differential equation |
| bloweybq | 10,001 | 49999 | Real symmetric | Materials problem |

**Table 3** Example 2: Numerical results of W-GLGMRES with different weighting stratiges

| | | $m = 10$ | | | $m = 15$ | | |
|---|---|---|---|---|---|---|---|
| Problem | Weighting Strategy | iter | res.norm | CPU | iter | res.norm | CPU |
| $A = sherman4$ | [16] | 6 | 1.5149e-06 | 43.6959 | 5 | 1.2774e-07 | 69.3429 |
| $B = fdm(cos(xy), e^{y^2x}, 100)$ | [36] | 7 | 3.2531e-07 | 44.1180 | 5 | 1.2228e-07 | 63.1971 |
| $n = 1104, s = 400$ | $D_3$ | 5 | 3.0576e−07 | 24.2452 | 4 | 3.8178e−08 | 41.2886 |
| $A = add32$ | [16] | 11 | 1.6391e−06 | 882.7363 | 7 | 1.6882e−07 | 472.5842 |
| $B = fdm(sin(xy), e^{xy}, 10)$ | [36] | 13 | 3.7399e−07 | 1.4418e+03 | 7 | 2.5876e−07 | 1.5228e+03 |
| $n = 4960, s = 400$ | $D_3$ | 9 | 5.9532e−07 | 395.5188 | 5 | 1.3011e−07 | 795.2263 |
| $A = saylr4$ | [16] | 34 | 1.3957e−06 | 1.7681e+03 | 16 | 8.7617e−07 | 1.7514e+03 |
| $B = fdm(sin(xy), e^{xy}, 10)$ | [36] | 42 | 7.1649e−07 | 2.8118e+03 | 21 | 8.7432e−07 | 2.6701e+03 |
| $n = 3564, s = 400$ | $D_3$ | 28 | 2.2365e−07 | 928.8154 | 15 | 5.1916e−07 | 882.1374 |
| $A = sherman1$ | [16] | 11 | 1.6761e−06 | 77.7592 | 7 | 1.8580e−07 | 93.6094 |
| $B = fdm(sin(xy), e^{xy}, 10)$ | [36] | 13 | 3.5584e−07 | 86.6745 | 7 | 2.7730e−07 | 86.2779 |
| $n = 1000, s = 400$ | $D_3$ | 9 | 3.4312e−07 | 57.0749 | 5 | 2.1024e−07 | 50.5531 |
| $A = pde2961$ | [16] | 12 | 1.1415e−06 | 355.3121 | 8 | 3.5254e−07 | 408.9183 |
| $B = fdm(sin(xy), e^{xy}, 10)$ | [36] | 13 | 9.0506e−07 | 590.8999 | 8 | 4.1388e−07 | 698.5396 |
| $n = 2961, s = 400$ | $D_3$ | 9 | 4.2336e−07 | 257.7321 | 6 | 2.8330e−07 | 332.4973 |

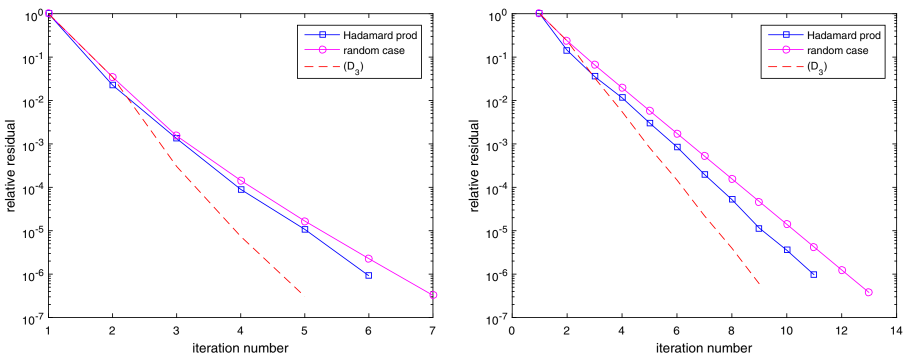


**Fig. 2** Example 2: Convergence curves of W-GLGMRES with different choices of $D$: the "Hadamard product" strategy [16], the "randomized" strategy [36], and $D_3$. Left: $A = sherman4$, Right: $A = add32$, $m = 10$

*Example 3* In this example, we try to show the efficiency of our three new algorithms. We compare the weighted global GMRES algorithm (W-GLGMRES, Algorithm 2), the weighted global GMRES with deflation (WGLGMRES-D, Algorithm 3), and the global GMRES algorithm with deflation (GLGMRES-D, Algorithm 4) with the global GMRES algorithm (GLGMRES) advocated in [32]. The matrix $A$ in all the problems is considered from Table 2. In the first test problem, we use $B = fdm(\cos(xy), e^{y^2 x}, 45)$, and for test problems 2–4, we use $B =$

$$\begin{pmatrix} 0.01 & 0.1 & & & & \\ & 0.2 & 0.1 & & & \\ & & 100 & 0.1 & & \\ & & & 200 & \ddots & \\ & & & & \ddots & 0.1 \\ & & & & & 9800 \end{pmatrix} \in \mathbb{R}^{100\times 100}.$$

For all these problems, the right-hand side is chosen as $C = A * eye(n, s) * B$. Table 4 reports the results of the four test problems, where we use $m = 20$ and $k = 10$ in W-GLGMRES-D and GLGMRES-D, and "†" denotes the algorithm did not converge within 2500 iterations or within 30 h.

It is seen that W-GLGMRES outperforms GLGMRES in all cases, which illustrates the effectiveness of our weighting strategy. Furthermore, W-GLGMRES-D has similar numerical behavior as GLGMRES-D in terms of iteration numbers and accuracy. In terms of CPU time, this table reports that W-GLGMRES and GLGMRES-D are faster than the other two, while GLGMRES-D performs the best and GLGMRES performs the poorest. This is reasonable because of the computation of $D$-inner product in the weighted algorithms. Moreover, it is interesting to see that W-GLGMRES runs faster than W-GLGMRES-D if they use about the same number of iterations. This is because the latter costs more than W-GLGMRES per iteration, where one needs to compute a Cholesky factorization in each iteration; see Step 12 of Algorithm

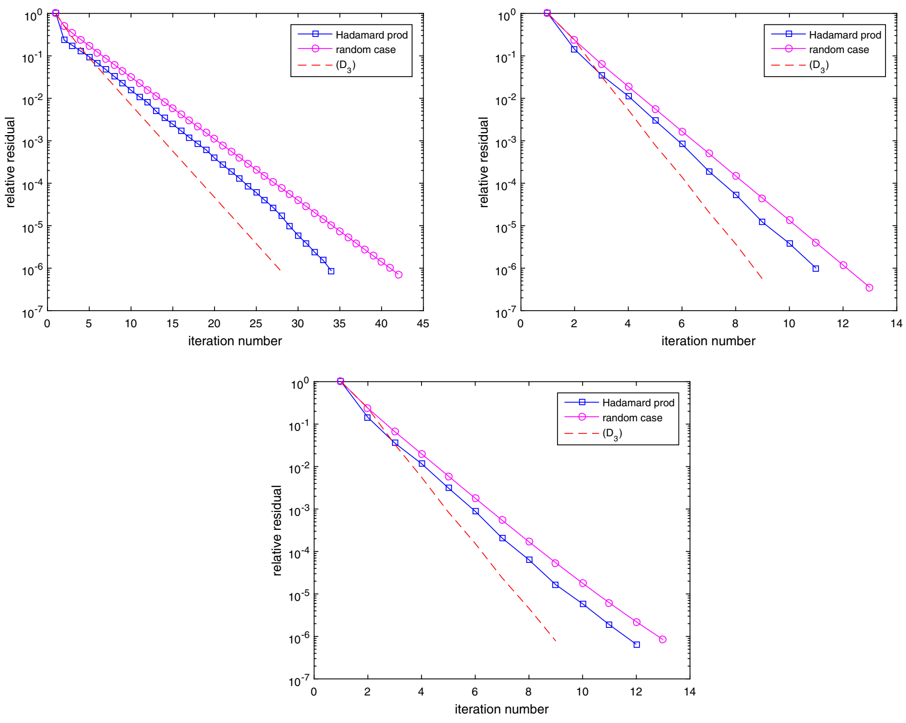


**Fig. 3** Example 2: Convergence curves of W-GLGMRES with different choices of $D$: the "Hadamard product" strategy [16], the "randomized" strategy [36], and $D_3$. Left: $A = saylr4$, middle: $A = sherman1$, Right: $A = pde2961$, $m = 10$

3. In Figs. 4 and 5, we plot the convergence curves of GLGMRES, WGLGMRES, GLGMRES-D, and W-GLGMRES-D, with $m = 20, k = 10$. From Table 4 and Figs. 4 and 5, we conclude that applying deflation strategy or with weighting technique leads to much better numerical performance than their original counterpart.

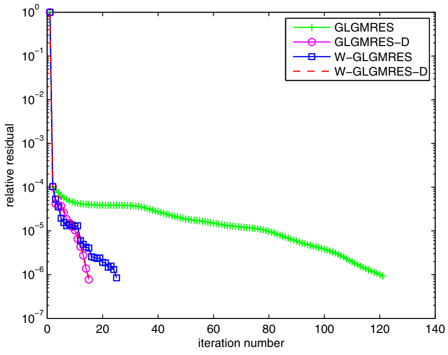
*Example 4* In this example, we compare our global GMRES algorithm with deflation (GLGMRES-D), the weighted GLGMRES algorithm (W-GLGMRES), and the weighted global GMRES algorithm with deflation (W-GLGMRES-D) with the implicitly restarted global GMRES algorithm (GLGMRES-IR) [24]. Table 5 lists the numerical results for the comparison of the above algorithms as $m = 10$, $k = 3$, and $s = 100$. The matrix $B$ for the first two problems is $B =$

$$\begin{pmatrix} 0.01 & 0.001 & & & & \\ & 0.2 & 0.001 & & & \\ & & 0.001 & 0.001 & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & 0.001 \\ & & & & & 0.0098 \end{pmatrix} \in \mathbb{R}^{100\times 100},$$

**Table 4** Example 3: A comparison of the four algorithms

| | Problem | $(m,k)$ | Algorithm | iter | res.norm | CPU |
|---|---|---|---|---|---|---|
| | | (20,-) | GLGMRES | 121 | 9.3700e−07 | 7.5800e+03 |
| 1 | $A = sherman4$ | (20,-) | W-GLGMRES | 25 | 7.6326e−07 | 180.4981 |
| | $n = 1104, s = 225$ | (20,10) | GLGMRES-D | 15 | 6.1557e−06 | 55.9331 |
| | | (20,10) | W-GLGMRES-D | 15 | 7.9004e−07 | 150.9944 |
| | | (20,-) | GLGMRES | 23 | 9.6257e−07 | 135.3658 |
| 2 | $A = cavity15$ | (20,-) | W-GLGMRES | 4 | 6.3192e−07 | 50.2583 |
| | $n = 2597, s = 100$ | (20,10) | GLGMRES-D | 5 | 6.4379e−07 | 10.1583 |
| | | (20,10) | W-GLGMRES-D | 4 | 1.4565e−06 | 61.1481 |
| | | (20,-) | GLGMRES | 62 | 9.5311e−07 | 280.7433 |
| 3 | $A = saylr3$ | (20,-) | W-GLGMRES | 10 | 3.8104e−07 | 25.2682 |
| | $n = 1000, s = 100$ | (20,10) | GLGMRES-D | 9 | 3.0705e−07 | 8.8104 |
| | | (20,10) | W-GLGMRES-D | 9 | 3.1185e−07 | 30.7022 |
| | | (20,-) | GLGMRES | 40 | 6.7746e−07 | 471.7118 |
| 4 | $A = sherman5$ | (20,-) | W-GLGMRES | 6 | 2.6253e−07 | 120.9306 |
| | $n = 3312, s = 100$ | (20,10) | GLGMRES-D | 6 | 6.6650e−07 | 15.3136 |
| | | (20,10) | W-GLGMRES-D | 6 | 9.8813e−07 | 153.5709 |

and it for the third problem is $B = \begin{pmatrix} 0.01 & 0.01 & & & & \\ & 0.2 & 0.01 & & & \\ & & 100 & 0.01 & & \\ & & & 200 & \ddots & \\ & & & & \ddots & 0.01 \\ & & & & & 9800 \end{pmatrix} \in \mathbb{R}^{100\times 100}$.



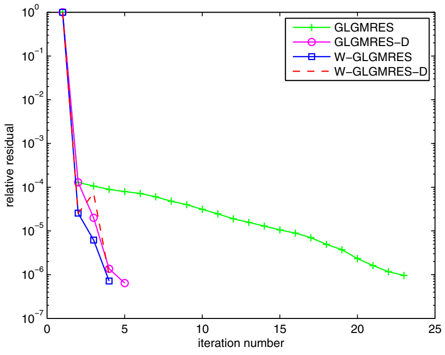


**Fig. 4** Example 3: Convergence curves of the four algorithms. Left: $A = sherman4$, Right: $A = cavity15$

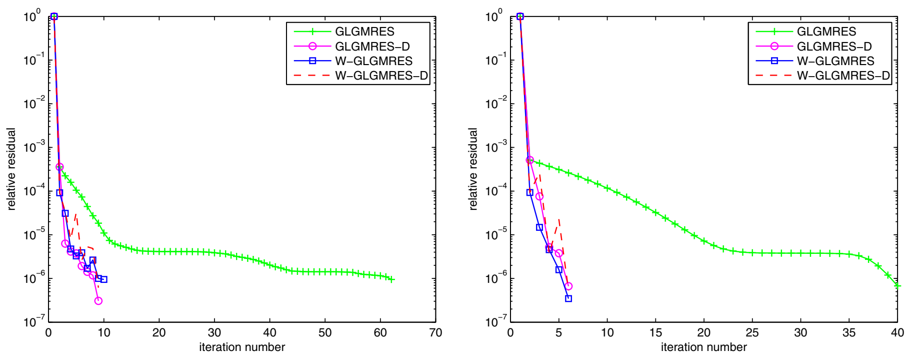


**Fig. 5** Example 3: Convergence curves of the four algorithms. Left: $A = saylr3$, Right: $A = sherman5$

For the fourth problem, we use $B = fdm(10, 0, 0, -100)$. The matrix $C$ for all the four problems is chosen as $C = A * eye(n, s) * B$.

It is seen from Table 5 that the three proposed algorithms converge faster than GLGMRES-IR in most cases, and GLGMRES-DR outperforms the others in terms of CPU time. Interestingly, we find that W-GLGMRES uses the fewest iterations for all the problems, except for the `saylr 4` matrix. This demonstrates the efficiency of our weighting strategy. On the other hand, it also reveals that a combination of the weighting strategy with the deflation strategy may not yield better numerical performance than using them separately (see also [10]). Figures 6 and 7 show

**Table 5** Example 4: A comparison of the four algorithms, $m = 10$, $k = 3$, and $s = 100$

| *Problem* | Methods | iter | res.norm | CPU time |
|---|---|---|---|---|
| | W-GLGMRES | 14 | 8.7349e−07 | 44.9998 |
| $A = sherman4$ | W-GLGMRES-D | 16 | 7.2553e−07 | 67.1982 |
| | GLGMRES-D | 16 | 8.5039e−07 | 20.4577 |
| $n = 1104$ | GLGMRES-IR | 23 | 8.4414e−07 | 138.1739 |
| | W-GLGMRES | 97 | 1.0330e−07 | 1.2012e+03 |
| $A = saylr4$ | W-GLGMRES-D | 87 | 1.0988e−07 | 2.1156e+03 |
| | GLGMRES-D | 251 | 9.0001e−07 | 252.8007 |
| $n = 3564$ | GLGMRES-IR | 251 | 9.9943e−07 | 1.1345e+03 |
| | W-GLGMRES | 22 | 6.5422e−07 | 27.8391 |
| $A = rdb1250l$ | W-GLGMRES-D | 34 | 5.1828e−07 | 55.3448 |
| | GLGMRES-D | 82 | 9.9967e−07 | 19.3840 |
| $n = 1250$ | GLGMRES-IR | 359 | 9.9996e−07 | 375.2365 |
| | W-GLGMRES | 33 | 8.5983e−07 | 279.4737 |
| $A = pde2961$ | W-GLGMRES-D | 48 | 8.2086e−07 | 565.1410 |
| | GLGMRES-D | 131 | 9.9776e−07 | 119.7611 |
| $n = 2961$ | GLGMRES-IR | 65 | 8.2135e−07 | 242.2317 |

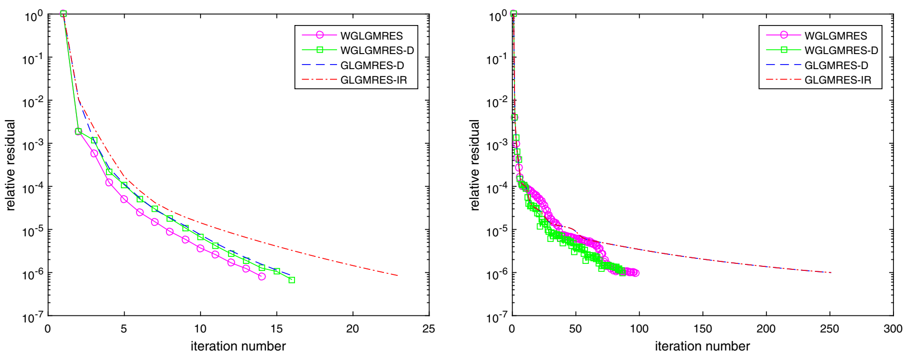


**Fig. 6** Example 4: Convergence curves of the four algorithms. Left: $A = sherman4$, Right: $A = saylr4$

the convergence curves of these four algorithms for the four test problems. It is observed that W-GLGMRES, W-GLGMRES-D, and GLGMRES-D work better than GLGMRES-IR algorithm.

*Example 5* In this example, we compare our new algorithms with other global GMRES algorithms for solving (1.1). To this aim, we compare Algorithm 2 (W-GLGMRES), Algorithm 3 (W-GLGMRES-D), and Algorithm 4 (GLGMRES-D) with the extended Sylvester global Arnoldi algorithm (LRS-EGA) proposed in [15] and an extended Krylov subspace-based method (EKS-SYL) [25], as well as two preconditioned algorithms ILU-GLGMRES and SSOR-GLGMRES advocated in [4].

There are four test matrices that are available from the *Matrix Market* [27] and *the university of Florida sparse matrix collection* [12]; see Table 2. The matrix $B$ for Table 6 is considered as $B = fdm(\cos(xy), e^{y^2x}, 100)$, and for Table 7 the matrices $A$ and $B$ are generated by the command $fdm\_2d\_matrix$ from LYAPACK [34] (see Example 1):

$$A = fdm(e^{x^2+y}, 2xy, \cos(xy)), \quad B = fdm(\sin(x+2y), e^{xy}, xy).$$

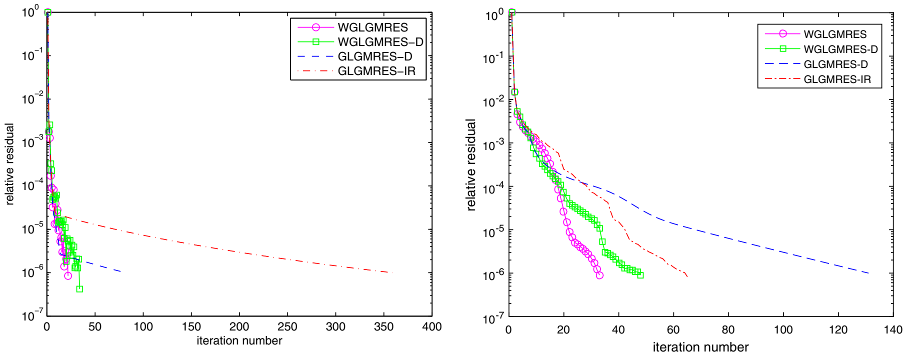


**Fig. 7** Example 4: Convergence curves of the four algorithms. Left: $A = rdb1250l$, Right: $A = pde2961$

**Table 6** Example 5: A comparison of the seven algorithms, where "†" denotes the algorithm did not converge within 2500 iterations or within 30 h

| Matrix | Algorithm | iter | res.norm | CPU-time |
|---|---|---|---|---|
| | ILU-GLGMRES | † | † | † |
| | SSOR-GLGMRES | 7 | 6.5081e−07 | 70430.9290 |
| bloweybq | LRS-EGA | † | † | † |
| $n = 10001$ | EKS-SYL | 10 | 2.1862e−07 | 1.7975e+03 |
| $s = 100$ | W-GlGMRES | 6 | 9.6782e−07 | 312.6073 |
| | W-GLGMRES-D | 10 | 2.1164e−07 | 672.2138 |
| | GLGMRES-D | 11 | 5.2675e-07 | 20.6623 |
| | ILU-GLGMRES | 49 | 1.2646e−06 | 387.7105 |
| | SSOR-GLGMRES | 6 | 6.3859e−07 | 3.3256e+03 |
| add32 | LRS-EGA | 2 | 6.5846e−06 | 1.2424 |
| $n = 4960$ | EKS-SYL | 228 | 9.8553e−07 | 51.1381 |
| $s = 100$ | W-GlGMRES | 3 | 6.5122e−07 | 31.7400 |
| | W-GLGMRES-D | 4 | 1.2847e−07 | 60.0364 |
| | GLGMRES-D | 4 | 1.2888e−07 | 2.9291 |
| | ILU-GLGMRES | † | † | † |
| | SSOR-GLGMRES | 7 | 1.3557e−07 | 3.3448e+03 |
| pde2961 | LRS-EGA | 4 | 8.3711e−06 | 7.0141 |
| $n = 2961$ | EKS-SYL | 242 | 9.7226e−07 | 22.1783 |
| $s = 100$ | W-GlGMRES | 3 | 8.5883e−07 | 11.7007 |
| | W-GLGMRES-D | 4 | 8.9621e−07 | 23.8213 |
| | GLGMRES-D | 4 | 7.6716e−07 | 1.9304 |
| | ILU-GLGMRES | † | † | † |
| | SSOR-GLGMRES | 26 | 1.0931e−06 | 1.8292e+03 |
| sherman4 | LRS-EGA | 18 | 1.3305e−05 | 95.9448 |
| $n = 1104$ | EKS-SYL | 422 | 9.8821e−07 | 478.7562 |
| $s = 400$ | W-GlGMRES | 87 | 3.9116e−07 | 366.9429 |
| | W-GLGMRES-D | 141 | 6.1300e−06 | 984.3839 |
| | GLGMRES-D | 297 | 9.8154e−06 | 1.0240e+03 |

The extended Sylvester global Arnoldi algorithm (LRS-EGA) applies to the continuous time Sylvester matrix equations of the form [15]

$$AX + XB + EF^T = \mathbf{0}, \tag{4.3}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{s\times s}$ are nonsingular matrices, and $E \in \mathbb{R}^{n\times r}$, $F \in R^{s\times r}$ have full rank $r$ with $r \ll n, s$. Note that (4.3) is a special case of (1.1), with $C = -EF^T$. Given $n, r$, and $s$, we use $E = rand(n, r)$ and $F = rand(s, r)$ that are generated randomly by using the MATLAB command `rand`. Note that LRS-EGA is a non-restarted algorithm, similar to [15], we use the `LU` command in MATLAB for $A^{-1}$, and the CPU time includes that for both LU decomposition and for solving the

**Table 7** Example 5: A comparison of the seven algorithms, where "†" denotes the algorithm did not converge within 2500 iterations or within 30 h; $A = fdm(e^{x^2+y}, 2xy, \cos(xy))$, $B = fdm(\sin(x + 2y), e^{xy}, xy)$

| Matrix | Algorithm | iter | res.norm | CPU-time |
|---|---|---|---|---|
| | ILU-GLGMRES | 5 | 4.7637e−07 | 0.1378 |
| | SSOR-GLGMRES | 27 | 7.5161e−07 | 4.8566 |
| $n = 225$ | LRS-EGA | 8 | 1.3779e−06 | 0.8719 |
| | EKS-SYL | 83 | 9.0816e−07 | 63.8299 |
| $s = 25$ | W-GlGMRES | 8 | 1.9179e−07 | 0.1457 |
| | W-GLGMRES-D | 10 | 3.1800e−07 | 0.3317 |
| | GLGMRES-D | 10 | 3.4966e−07 | 0.0870 |
| | ILU-GLGMRES | 5 | 9.6511e−07 | 0.3294 |
| | SSOR-GLGMRES | 43 | 5.6154e−07 | 24.7481 |
| $n = 400$ | LRS-EGA | 9 | 2.7382e−06 | 1.6641 |
| | EKS-SYL | 81 | 9.0013e−07 | 245.2098 |
| $s = 25$ | W-GlGMRES | 10 | 7.0130e−07 | 0.3994 |
| | W-GLGMRES-D | 13 | 6.5937e−07 | 0.7873 |
| | GLGMRES-D | 13 | 7.4603e−07 | 0.1115 |
| | ILU-GLGMRES | 6 | 1.1559e−06 | 17.1003 |
| | SSOR-GLGMRES | 317 | 9.8604e−07 | 2.7836e+04 |
| $n = 2500$ | LRS-EGA | 11 | 2.0017e−06 | 37.7832 |
| | EKS-SYL | 179 | 9.6058e−07 | 301.1042 |
| $s = 25$ | W-GlGMRES | 29 | 9.9150e−07 | 69.6093 |
| | W-GLGMRES-D | 33 | 5.6430e−07 | 109.5230 |
| | GLGMRES-D | 44 | 7.9598e−07 | 2.1742 |
| | ILU-GLGMRES | 7 | 3.7371e−06 | 347.1237 |
| | SSOR-GLGMRES | † | † | † |
| $n = 10000$ | LRS-EGA | 12 | 2.5437e−06 | 721.6443 |
| | EKS-SYL | 450 | 9.9780e−07 | 930.8535 |
| $s = 25$ | W-GlGMRES | 104 | 7.8562e−07 | 3.7433e+03 |
| | W-GLGMRES-D | 98 | 7.5191e−07 | 4.3108e+03 |
| | GLGMRES-D | 138 | 9.5315e−07 | 40.7113 |

Sylvester equation. Moreover, we set $r = 5$, the step-size parameter to be 1, and the tolerance $\tau = 0.01$ for the truncated SVD in this algorithm; for more details, refer to [15, Algorithm 4]. In the ILU-GLGMRES algorithm, we use the pivot threshold $thresh = 0.01$ for the ILU decomposition, and the relaxation parameter is chosen as $\omega = 0.9$ in the SSOR-GLGMRES algorithm. In this example, we make use of $m = 10$, $k = 5$ in W-GLGMRES-D and GLGMRES-D.

Tables 6 and 7 report the numerical results, where "†" denotes the algorithm did not converge within 2500 iterations or within 30 h. Specially, as LRS-EGA is a non-restarted algorithm, here "iter" means the number of Krylov steps. It is seen that our three new algorithms W-GLGMRES, W-GLGMRES-D, and GLGMRES-D outperform the other four algorithms significantly. For instance, we see from Table 6 that ILU-GLGMRES fails to converge for many problems, while SSOR-GLGMRES is very costly per iteration. The EKS-SYL algorithm [25] is an extended Krylov subspace-based method for solving Sylvester matrix equations, as is seen from Tables 6 and 7, it needs more iterations than the other methods. The LRS-EGA algorithm performs better than ILU-GLGMRES, SSOR-GLGMRES, and EKS-SYL, but it fails to converge for the `bloweybq` matrix. Therefore, our new algorithms are superior to these algorithms and are suitable to Sylvester matrix equations with large $A$ and small $B$. On the other hand, we note from Tables 6 and 7 that the relative "real" residual `res.norm` reported may be larger than the tolerance $tol = 10^{-6}$. This is due to the fact that we use the "computed" residual as the stopping criterion (see (4.1)).

## 5 Conclusion

The global GMRES algorithm is popular for large Sylvester matrix equations with large $A$ and small $B$. The weighting strategy can be viewed as a preconditioned technique and thus improve the algorithm by deflating the eigenvalues that hinder the convergence. However, the optimal choice of the weighting matrix is still an open problem and needs further investigation. Furthermore, due to the growth of memory requirements and computational cost, it is necessary to seek efficient restarting strategies for the global GMRES algorithm.

In this work, we propose three new algorithms for solving large Sylvester matrix equations. We first consider the weighted global GMRES algorithm (W-GLGMRES) and present three new schemes based on residuals to update the weighting matrix during iterations. Numerical experiments show that the third one $D_3$ is a good choice for the weighting matrix of W-GLGMRES. We then apply the deflated restarting strategy to the weighted algorithm, and give a weighted global GMRES algorithm with deflation (W-GLGMRES-D). Specifically, when $D = I$, it reduces to the global GMRES algorithm with deflation (GLGMRES-D), and we find that the identity $I$ is appropriate for the weighting matrix of W-GLGMRES-D. Extensive numerical experiments illustrate the superiority of our new algorithms for Sylvester matrix equations with large $A$ and small $B$.

**Acknowledgments** We would like to thank the reviewers and our editor Prof. Claude Brezinski for invaluable comments and suggestions that improve the representation of this paper significantly. Moreover, we are grateful to Prof. Yiqin Lin for providing us with the MATLAB codes of the EKS-SYL algorithm.

## References

1. Agoujil, S., Bentbib, A.H., Jabilou, K., Sadek, E.M.: A minimal residual norm method for large-scale Sylvester matrix equations. Electron. Tran. Numer. Anal. **43**, 45–59 (2014)
2. Bartels, R., Stewart, G.W.: Algorithm 432: The solution of the matrix equation $AX - XB = c$. Commun. ACM **8**, 820–826 (1972)

3. Benner, P., Li, R., Truhar, N.: On the ADI method for Sylvester equations. J. Comput. Appl. Math. **223**, 1035–1045 (2009)
4. Bouhamidi, A., Jbilou, K.: A note on the numerical approximate solutions for generalized Sylvester matrix equations with applications. Appl. Math. Comput. **206**, 687–694 (2008)
5. Calvetti, D.: Application of ADI iterative methods to the restoration of noisy images. SIAM J. Matrix Anal. Appl. **17**, 165–186 (1996)
6. Datta, B.N.: Numerical Methods for Linear Control Systems Design and Analysis. Elsevier Press (2003)
7. Datta, B.N., Datta, K.: Theoretical and computational aspects of some linear algebra problems in control theory. In: Byrnes, C.I., Lindquist, A. (eds.) Computational and Combinatorial Methods in Systems Theory, vol. 177, pp. 201–212. Elsevier, Amsterdam (1986)
8. Dehghan, M., Hajarian, M.: Two algorithms for finding the Hermitian reflexive and skew-Hermitian solutions of Sylvester matrix equations. Appl. Math. Lett. **24**, 444–449 (2011)
9. Duan, C., Jia, Z.: A global harmonic Arnoldi method for large non-Hermitian eigenproblems with an application to multiple eigenvalue problems. J. Comput. Appl. Math. **234**, 845–860 (2010)
10. Embree, M., Morgan, R.B., Nguyen, H.V.: Weighted inner products for GMRES and GMRES-DR. SIAM J. Sci. Comp. **39**, S610–S632 (2017)
11. Essai, A.: Weighted, FOM and GMRES for solving nonsymmetric linear systems. Numer. Alg. **18**, 277–292 (1998)
12. The University of Florida Sparse Matrix Collection, https://www.cise.ufl.edu/research/sparse/matrices
13. El Guennouni, A., Jbilou, K., Riquet, A.J.: Block Krylov subspace methods for solving large Sylvester equations. Numer. Alg. **29**, 75–96 (2002)
14. Güttel, S., Pestana, J.: Some observations on weighted GMRES. Numer. Alg. **67**, 733–752 (2014)
15. Heyouni, M.: Extended Arnoldi methods for large low-rank Sylvester matrix equations. Appl. Numer. Math. **60**, 1171–1182 (2010)
16. Heyouni, M., Essai, A.: Matrix Krylov subspace methods for linear systems with multiple right-hand sides. Numer. Alg. **40**, 137–156 (2005)
17. Jaimoukha, I.M., Kasenally, E.M.: Krylov subspace methods for solving large Lyapunov equations. SIAM J. Numer. Anal. **31**, 227–251 (1994)
18. Jbilou, K., Messaoudi, A., Sadok, H., Global, F.OM.: GMRES algorithms for matrix equations. Appl. Numer Math. **31**, 49–63 (1999)
19. Jbilou, K., Riquet, A.J.: Projection methods for large Lyapunov matrix equations. Linear Alg. Appl. **415**, 344–358 (2006)
20. Jiang, W., Wu, G.: A thick-restarted block Arnoldi algorithm with modified Ritz vectors for large eigenproblems. Comput. Math. Appl. **60**, 873–889 (2010)
21. Khorsand Zak, M., Toutounian, F.: Nested splitting CG-like iterative method for solving the continuous Sylvester equation and preconditioning. Adv. Comput. Math. **40**, 865–880 (2013)
22. Liesen, J., Strakos, Z.: Krylov Subspace Methods, Principles and Analysis. Oxford University Press (2013)
23. Lin, Y.: Minimal residual methods augmented with eigenvectors for solving Sylvester equations and generalized Sylvester equations. Appl. Math. Comput. **181**, 487–499 (2006)
24. Lin, Y.: Implicitly restarted global, FOM and GMRES for nonsymmetric matrix equations and Sylvester equations. Appl. Math. Comput. **167**, 1004–1025 (2005)
25. Lin, Y., Bao, L., Wei, Y.: A projection method based on extended krylov subspaces for solving Sylvester equations. Int. J. Math. Comput. Phys. Elec. Comput. Eng. **5**(7), 1064–1070 (2011)
26. Lin, Y., Simoncini, V.: Minimal residual methods for large scale Lyapunov equations. Appl. Numer. Math. **72**, 52–71 (2013)
27. Matrix Market, http://math.nist.gov/matrixMarket/
28. Morgan, R.: GMRES with deflated restarting. SIAM J. Sci. Comput. **24**(1), 20–37 (2002)
29. Morgan, R.: Restarted block, GMRES with deflation of eigenvalues. Appl. Numer. Math. **54**, 222–236 (2005)
30. Morgan, R., Zeng, M.: A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity. Linear Alg. Appl. **415**, 96–113 (2006)
31. Mohseni Mohgadam, M., Panjeh Ali Beik, F.: A new weighted global full orthogonalization method for solving nonsymmetric linear systems with multiple right-hand sides. Int. Electron. J. Pure Appl. Math. **2**, 47–67 (2010)
32. Panjeh Ali Beik, F., Mohseni Mohgadam, M.: Global generalized minimum residual method for solving Sylvester equation. Aust. J. Basic Appl. Sci. **5**, 1128–1134 (2011)

33. Panjeh Ali Beik, F., Khojasteh Salkuyeh, D.: Weighted versions of gl-FOM and gl-GMRES for solving general coupled linear matrix equations. Comput. Math. Math. Phys. **55**, 1606–1618 (2015)
34. Penzel, T.: LYAPACK: A MATLAB toolbox for large Lyapunov and Riccati equations, model reduction problems, and linear-quadratic optimal control problems, software available at https://www.tu-chemnitz.de/sfb393/lyapack/
35. Robbé, M., Sadkane, M.: A convergence analysis of GMRES and FOM methods for Sylvester equations. Numer. Alg. **30**, 71–89 (2002)
36. Saberi Najafi, H., Zareamoghaddam, H.: A new computational, GMRES Method. Appl. Math. Comput. **199**, 527–534 (2008)
37. Simoncini, V.: The extended Krylov subspace for parameter dependent systems. Appl. Numer. Math. **60**, 550–560 (2010)
38. Simoncini, V.: A new iterative method for solving large-scale Lyapunov matrix equations. SIAM J. Sci. Comput. **29**, 1268–1288 (2007)
39. Simoncini, V.: Computational methods for linear matrix equations. SIAM Rev. **58**, 377–441 (2016)
40. Sorensen, D.: Implicit application of polynomial filters in a k-step Arnoldi method. SIAM J. Matrix Anal. Appl. **13**, 357–385 (1992)
41. Wan, F.: An in-core finite difference method for separable boundary value problems on a rectangle. Stud. Appl. Math. **52**, 103–113 (1973)
42. Watkins, D.: The Matrix Eigenvalue Problem. GR and Krylov Subspace Methods. SIAM, Philadelphia (2007)
43. Wu, G., Wei, Y.: A Power-Arnoldi algorithm for computing PageRank. Numer. Linear Alg. Appl. **14**, 521–546 (2007)
44. Wu, K., Simon, H.: Thick-restart Lanczos method for sysmmetric eigenvalue problems. SIAM J. Matrix Anal. Appl. **22**, 602–616 (2000)
45. Zhong, H., Wu, G.: Thick restarting the weighted harmonic Arnoldi algorithm for large interior eigenproblems. Int. J. Comput. Math. **88**, 994–1012 (2011)

## Affiliations

**Najmeh Azizi Zadeh[1] · Azita Tajaddini[1] · Gang Wu[2]**

Najmeh Azizi Zadeh
nazizizadeh@math.uk.ac.ir

Azita Tajaddini
atajadini@uk.ac.ir

[1] Department of Applied Mathematics, Faculty of Mathematics & Computer Sciences, Shahid Bahonar University of Kerman, Kerman, Iran

[2] School of Mathematics, China University of Mining and Technology, Xuzhou, 221116, Jiangsu, People's Republic of China