

Some observations on weighted GMRES

Stefan Güttel · Jennifer Pestana

Received: date / Accepted: date

Abstract We investigate the convergence of the weighted GMRES method for solving linear systems. Two different weighting variants are compared with unweighted GMRES for three model problems, giving a phenomenological explanation of cases where weighting improves convergence, and a case where weighting has no effect on the convergence. We also present new alternative implementations of the weighted Arnoldi algorithm which may be favorable in terms of computational complexity, and examine stability issues connected with these implementations. Two implementations of weighted GMRES are compared for a large number of examples. We find that weighted GMRES may outperform unweighted GMRES for some problems, but more often this method is not competitive with other Krylov subspace methods like GMRES with deflated restarting or BICGSTAB, in particular when a preconditioner is used.

Keywords weighted GMRES · linear systems · Krylov subspace method · harmonic Ritz values

1 Introduction

The GMRES method of Saad and Schultz [24] is one of the most popular Krylov subspace methods for solving a non-Hermitian system of linear equa-

S. G. was supported by Deutsche Forschungsgemeinschaft Fellowship No. GU 1244/1-1. This publication was based on work supported in part by Award No. KUK-C1-013-04, made by King Abdullah University of Science and Technology (KAUST).

S. Güttel

Mathematical Institute, University of Oxford, 24–29 St Giles’, Oxford OX1 3LB, UK

J. Pestana

Mathematical Institute, University of Oxford, 24–29 St Giles’, Oxford OX1 3LB, UK

E-mail: pestana@maths.ox.ac.uk

tions $A\mathbf{x} = \mathbf{b}$, where $A \in \mathbb{C}^{N \times N}$ is invertible and $\mathbf{b} \in \mathbb{C}^N$. Given an initial guess $\mathbf{x}^{(0)}$, GMRES computes successive iterates $\mathbf{x}^{(k)}$, $k = 1, 2, \dots$, so that

$$\|\mathbf{r}^{(k)}\|_2 = \min_{\substack{p \in \mathcal{P}_k \\ p(0)=1}} \|p(A)\mathbf{r}^{(0)}\|_2,$$

where \mathcal{P}_k denotes the linear space of polynomials of degree at most k , and $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ is the k -th residual.

Since GMRES uses the Arnoldi algorithm, its computational cost increases with each iteration. An alternative is to restart GMRES after m iterations [24], taking the last computed residual as the next initial residual. We call the original method *full* GMRES and the latter *restarted* GMRES or GMRES(m). The set of m Arnoldi iterations between successive restarts will be called a *cycle*.

Although full GMRES is guaranteed to terminate with the exact solution in at most N steps, the restarted version may stagnate [9, 24, 31]. Even if stagnation does not occur, convergence can be extremely slow [3, 32, 33]. The behaviour of restarted GMRES has been well studied and a number of remedies for slow convergence have been proposed [8, 17, 23, 27–29].

One such remedy is the *weighted* GMRES method of Essai [10], shortly denoted as WGMRES(m), that aims to improve the convergence of GMRES(m) by using a weighted inner product that changes at each cycle. This weighted inner product will be called a D -inner product, and is defined for any Hermitian positive definite matrix $D \in \mathbb{C}^{N \times N}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{C}^N$ as

$$\langle \mathbf{x}, \mathbf{y} \rangle_D = \mathbf{y}^H D \mathbf{x},$$

where \mathbf{y}^H represents the Hermitian conjugate of \mathbf{y} . The associated D -norm of a vector $\mathbf{x} \in \mathbb{C}^N$ is

$$\|\mathbf{x}\|_D = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_D}.$$

The WGMRES(m) method also starts from an initial guess $\mathbf{x}^{(0)}$, and then computes successive approximations $\mathbf{x}^{(k)}$, $k = 1, 2, \dots$, such that at the end of the k -th cycle

$$\|\mathbf{r}^{(k)}\|_D = \min_{\substack{p \in \mathcal{P}_m \\ p(0)=1}} \|p(A)\mathbf{r}^{(k-1)}\|_D. \quad (1)$$

For further details we refer to Essai [10].

The essential ingredient of weighted GMRES is the weighted Arnoldi algorithm [10] which, after m iterations, generates basis vectors $\mathbf{v}_1, \dots, \mathbf{v}_m$ of the Krylov space

$$\mathcal{K}_m(A, \mathbf{r}) = \text{span}\{\mathbf{r}, A\mathbf{r}, \dots, A^{m-1}\mathbf{r}\}.$$

If one collects the Krylov basis vectors in a matrix $V_m = [\mathbf{v}_1, \dots, \mathbf{v}_m] \in \mathbb{C}^{N \times m}$, one can write down an Arnoldi decomposition

$$AV_m = V_{m+1} \underline{H}_m = V_m H_m + \mathbf{v}_{m+1} h_{m+1,m} \mathbf{e}_m^T, \quad (2)$$

where $\underline{H}_m \in \mathbb{C}^{(m+1) \times m}$ is the upper Hessenberg matrix

$$\underline{H}_m = \begin{bmatrix} H_m \\ h_{m+1,m} \mathbf{e}_m^T \end{bmatrix},$$

and \mathbf{e}_m is the m -th canonical unit vector. The matrix $V_{m+1} = [V_m, \mathbf{v}_{m+1}]$ is D -orthonormal, i.e., $V_{m+1}^H D V_{m+1} = I_{m+1}$, the identity matrix of dimension $m+1$. The weighted Arnoldi algorithm requires more computation per iteration than standard Arnoldi in the Euclidean inner product and, consequently, one cycle of WGMRES(m) is computationally more expensive than one of GMRES(m). However, convergence may occur more quickly.

Essai [10] considered the particular weight matrix

$$D = \frac{1}{\sqrt{N} \|\mathbf{r}^{(k-1)}\|_2} \text{diag}(|r_1^{(k-1)}|, |r_2^{(k-1)}|, \dots, |r_N^{(k-1)}|), \quad (3)$$

where the $r_j^{(k-1)}$ are the entries of the residual vector $\mathbf{r}^{(k-1)}$, so that greater emphasis is given to large components of the residual at each cycle. Note that $D = D^{(k)}$ changes at each cycle, but to keep notation simple, we typically omit the superindex k . Note that D may be poorly conditioned if the diagonal entries vary too much in magnitude. In such cases, adding a small multiple of the identity will improve the conditioning of D .

For a number of test problems, WGMRES(m) with the weight matrix (3) required fewer cycles and less CPU time than the standard GMRES(m) method [10]. Application of WGMRES(m) to systems left-preconditioned by ILU(0) [16] also resulted in a slight reduction in the number of cycles required for convergence when compared with GMRES(m) [4]. However, the CPU time for WGMRES(m) was greater, with the additional CPU time a consequence of the computation of nonstandard inner products and norms. The weighted GMRES method has also been used to solve shifted linear systems [15], and systems with multiple right-hand sides [14]. We remark that Niu *et al.* [20] showed that WGMRES(m) can be accelerated by augmenting the Krylov subspace with ℓ error approximations $\mathbf{z}^{(i)}$, $i = 1 \dots, \ell$, where $\mathbf{z}^{(i)} = \mathbf{x}^{(i)} - \mathbf{x}^{(i-1)}$ when $i > 0$, and $\mathbf{z}^{(i)} = \mathbf{0}$ otherwise.

Although intuitively it seems sensible to emphasise those entries of the residual vector that are large in magnitude, the convergence behaviour of WGMRES(m) is not well understood. We attempt to remedy this here by examining the harmonic Ritz values associated with WGMRES(m). In view of the limited understanding even of the convergence of full GMRES, it seems unlikely at this stage that a simple and complete convergence theory for WGMRES can be developed. However, some insight can be gained by studying several model problems. We also propose three alternative implementations of the weighted Arnoldi algorithm and compare their cost and stability.

The outline of this paper is as follows. An analysis of the harmonic Ritz values associated with GMRES(m) and WGMRES(m) is given in Section 2. In Section 3 we describe Essai's implementation of the weighted Arnoldi algorithm and propose alternative implementations. We additionally discuss the

merits of each. Finally, in Section 4, the different implementations are tested on a number of problems and compared with standard GMRES(m), GMRES(m) with deflated restarting, and BICGSTAB.

2 Harmonic Ritz values and the convergence of weighted GMRES

It is known, see [12, 30, 34], that the weighted harmonic Ritz values θ_j with corresponding Ritz vectors \mathbf{u}_j satisfy

$$\mathbf{u}_j \in \mathcal{K}_m(A, \mathbf{r}), \quad (A^{-1}\mathbf{u}_j - \theta_j\mathbf{u}_j) \perp_D A\mathcal{K}_m(A, \mathbf{r}), \quad (4)$$

where $\mathbf{u} \perp_D \mathcal{V}$ means that $\langle \mathbf{u}, \mathbf{v} \rangle_D = 0$ for every $\mathbf{v} \in \mathcal{V}$. The condition $\mathbf{u}_j \in \mathcal{K}_m(A, \mathbf{r})$ is equivalent to $\mathbf{u}_j = V_m \mathbf{z}_j$, where V_m has as its columns the D -orthonormal basis vectors of $\mathcal{K}_m(A, \mathbf{r})$ in (2), and $\mathbf{z}_j \in \mathbb{C}^m$. It follows from (2) and (4) that

$$(H_m + |h_{m+1,m}|^2 \mathbf{f}_m \mathbf{e}_m^T) \mathbf{z}_j = \theta_j \mathbf{z}_j,$$

where $\mathbf{f}_m = H_m^{-H} \mathbf{e}_m$. It is also well known that the harmonic Ritz values $\theta_1^{(k)}, \dots, \theta_m^{(k)}$ associated with cycle k are the zeros of the residual polynomial $p_m^{(k)} \in P_m$, $p_m^{(k)}(0) = 1$, which is uniquely determined by the condition

$$\begin{aligned} \|\mathbf{r}^{(k)}\|_{D^{(k)}} &= \|p_m^{(k)}(A) p_m^{(k-1)}(A) \cdots p_m^{(1)}(A) \mathbf{r}^{(0)}\|_{D^{(k)}} \\ &= \min_{\substack{p \in P_m \\ p(0)=1}} \|p(A) p_m^{(k-1)}(A) \cdots p_m^{(1)}(A) \mathbf{r}^{(0)}\|_{D^{(k)}}. \end{aligned}$$

If A is normal, then with $\tilde{p}^{(k)} := p_m^{(k)} p_m^{(k-1)} \cdots p_m^{(1)}$ we have

$$\|\mathbf{r}^{(k)}\|_{D^{(k)}} \leq \|\mathbf{r}^{(0)}\|_{D^{(k)}} \max_{\lambda \in \Lambda(A)} |\tilde{p}^{(k)}(\lambda)|,$$

so that the convergence of (restarted) GMRES in the 2-norm can be understood in terms of the uniform convergence of residual polynomials on the discrete set of eigenvalues $\Lambda(A)$.

In this section we compare the harmonic Ritz values of GMRES(m) with the weighted harmonic Ritz values of WGMRES(m). As well as Essai's weight matrix (3) we consider an alternative, proposed by Najafi and Zareamoghaddam [19], who were concerned that as the entries of the residual became smaller (3) would be difficult to compute with; D_{rand} is a diagonal matrix with random uniformly distributed entries in $(0.5, 1.5)$. Although the following three examples have little practical relevance, we believe that they serve the purpose of giving insight into how weighting can possibly improve the convergence of restarted GMRES (Examples 1 and 3), or how weighting can have absolutely no effect on the convergence (Example 2).

Example 1 (interval) We examine the harmonic Ritz values of a diagonal matrix with diagonal entries (and, hence, eigenvalues) $1, 2, \dots, 100$. The right-hand side \mathbf{b} is a vector of all ones, scaled to unit length. As one can see in Figure 1 (a), it appears that for unweighted GMRES(m) with $m = 5$ the harmonic Ritz values of every second cycle have m accumulation points, giving asymptotically $2m$ accumulation points $\theta_1^*, \dots, \theta_{2m}^*$ in total. In this example these accumulation points are approximately

$$\begin{aligned} &3.348, 22.208, 51.510, 79.318, 96.908, \\ &3.453, 20.616, 49.477, 79.784, 98.155. \end{aligned}$$

It should be noted that a similar 2-cyclic behaviour has been observed and analyzed for the so-called optimum gradient method in the 1950's [11, 2] (this method can be interpreted as restarted FOM), and in the context of matrix function approximations in [1]. A detailed investigation of this behaviour for restarted GMRES is beyond the scope of this paper, but we expect that similar tools to those used in the mentioned papers can be applied. In Figure 1 (b) we show the level lines of the modulus of the nodal polynomial $q_{2m}(z) = \prod_{j=1}^{2m} (z - \theta_j^*)$ (these level lines are also known as lemniscates, see also the discussion in [1]). One can read off from this plot that the level line $10^{14.535}$ is the smallest one containing $\Lambda(A)$ in its interior, and the level line $10^{14.909}$ passes through the origin. By the normalization condition of residual polynomials, the modulus of $q_{2m}(\lambda)/q_{2m}(0)$ is at most $10^{-0.3740} \approx 0.4227$ for all $\lambda \in \Lambda(A)$. This residual polynomial is the result of two restart cycles, hence the expected convergence rate of restarted unweighted GMRES(m) in this example is approximately $\sqrt{0.4227} \approx 0.6502$. This rate is shown in Figure 1 (c) as the black dashed line, and it coincides well with the observed linear convergence of unweighted GMRES(m) (black curve with + markers).

The convergence of the weighted GMRES(m) variants under consideration appears much less regular. The harmonic Ritz values associated with Essai's weighting appear to cover the spectral interval of A more evenly, and this is also indicated by the histogram in Figure 1 (d), which shows the frequency of harmonic Ritz values appearing on the spectral interval of A . This “randomization” of interpolation nodes causes the method to converge faster than linearly. A similar effect is achieved by random weighting with D_{rand} .

Example 2 (circle) Our second example is a diagonal matrix with $N = 100$ diagonal elements (eigenvalues) $\beta \cdot e^{2i\pi j/N} + 1$ on a circle of radius $\beta = 0.9$ centered at $z = 1$, $j = 1, \dots, N$. The right-hand side \mathbf{b} is a vector of all ones, scaled to unit length, and the restart length is $m = 5$. As can be seen from Figure 2 (a), the harmonic Ritz values “spiral” towards the point $z = 1$, being almost evenly spaced on concentric circles. This effect appears for all types of weighting under investigation, and the convergence shown in Figure 2 (b) seems to be unaffected by whatever weighting method we use. To explain this observation, assume that at some cycle the harmonic Ritz values are unit roots of order m shifted and scaled to a circle of radius $\alpha < \beta$ centered at $z = 1$. The corresponding nodal polynomial is $q_m(z) = (z - 1)^m - \alpha^m$. As can be

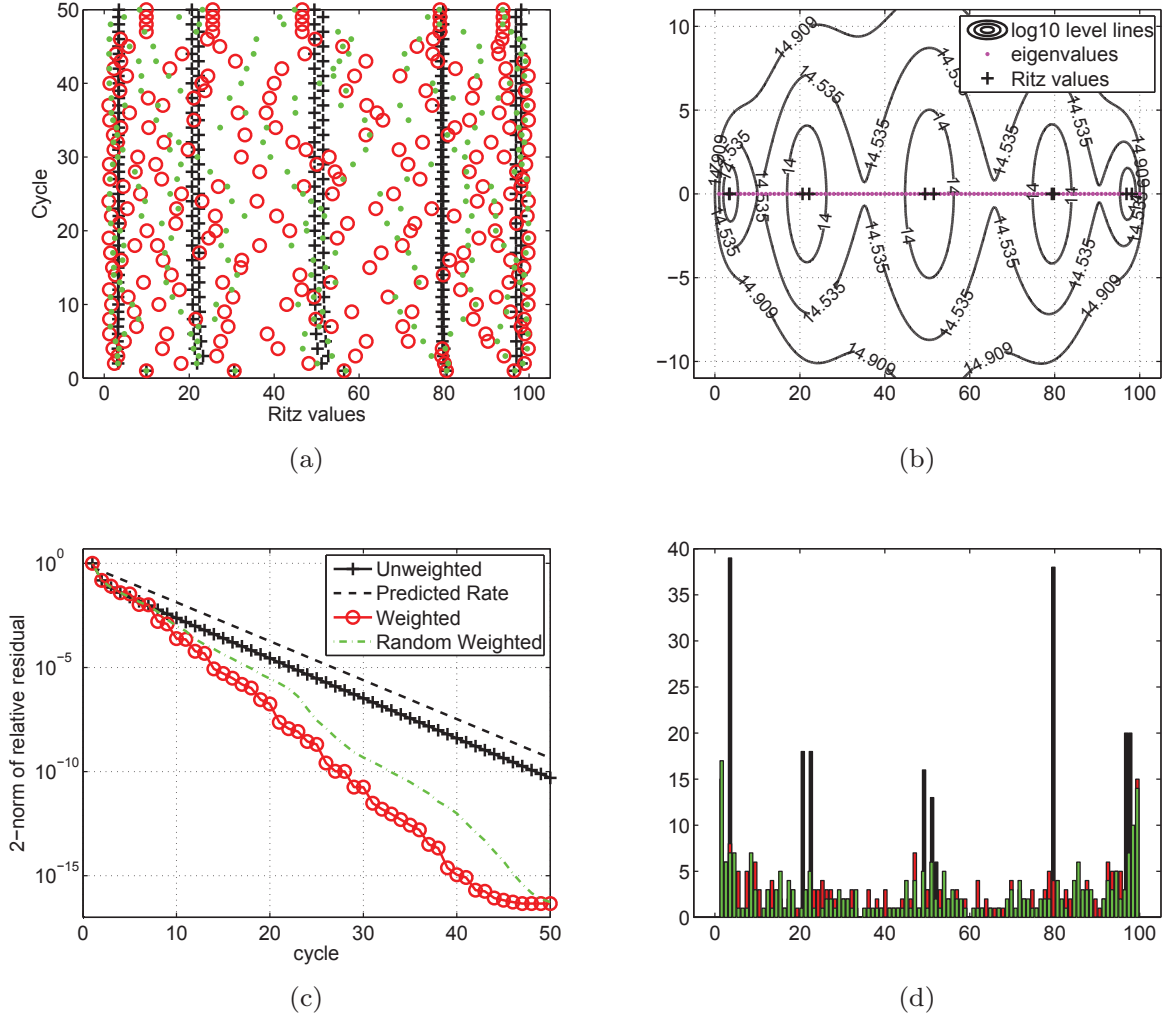


Fig. 1: (a) Harmonic Ritz values for GMRES(5) (black +), WGMRES(5) with (3) (red o), and WGMRES(5) with D_{rand} (green dots) for a diagonal matrix with equispaced eigenvalues on $[1, 100]$. The harmonic Ritz values are shown at the end of each of 50 cycles. (b) Lemniscates associated with unweighted GMRES(5). (c) Relative 2-norm residuals for the various GMRES(5) variants. (d) Histogram indicating the distribution of harmonic Ritz values.

verified easily, the maximal modulus of q_m on the circle of radius β centered at $z = 1$ is attained at points “in the middle” of two neighboring eigenvalues, for example, at the point $z^* = 1 + \beta e^{\pi i/N}$. Hence, the modulus of the residual polynomial is bounded by

$$\left| \frac{q_m(z^*)}{q_m(0)} \right| = \left| \frac{q_m(1 + \beta e^{\pi i/N})}{q_m(0)} \right| = \left| \frac{\beta^m e^{\pi i m/N} - \alpha^m}{(-1)^m - \alpha^m} \right| \approx \beta^m$$

for sufficiently small α . This explains why we see convergence with rate β in Figure 2 (b), indicated by the black dashed line, and weighting has essentially no effect on the convergence here.

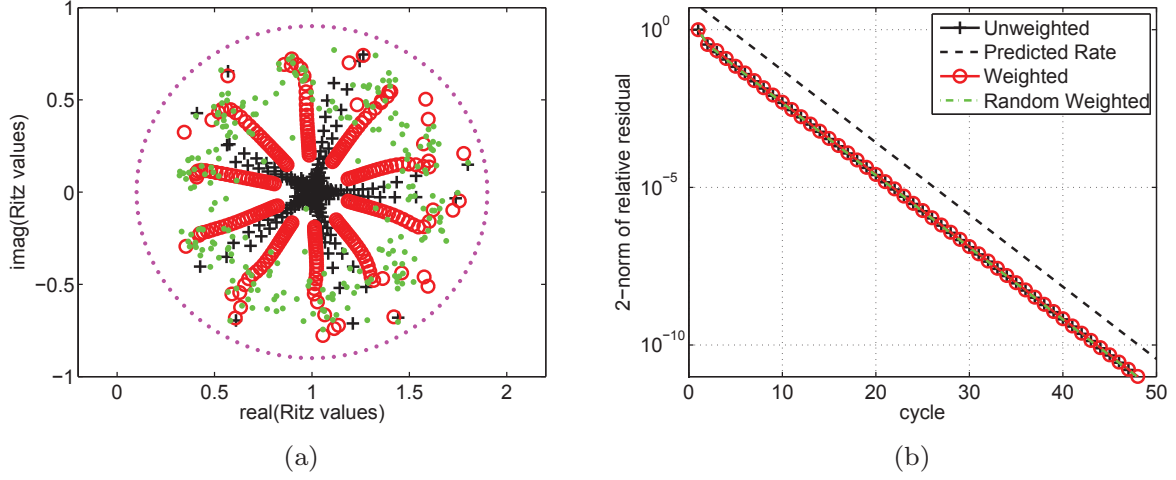


Fig. 2: (a) Harmonic Ritz values for GMRES(5) (black +), WGMRES(5) with (3) (red o), and WGMRES(5) with D_{rand} (green dots) for a 100×100 matrix with eigenvalues distributed on the shifted unit circle. The harmonic Ritz values are shown at the end of each of 50 cycles. (b) Relative 2-norm residuals for the various GMRES(5) variants.

Example 3 (Jordan block) Our next example is a Jordan block of size $N = 100$ with eigenvalue 1. The right-hand side \mathbf{b} is a vector of all ones, scaled to unit length. As one can see in Figure 3 (a), the unweighted GMRES(5) method will stagnate except in the first cycle, where a little progress is made. The corresponding harmonic Ritz values reappear at 10 points on a circle of radius one around the eigenvalue 1, with the real point $\theta = 2$ being counted twice due to symmetry, see Figure 3 (b). The harmonic Ritz values associated with the weighted GMRES(5) method move towards the eigenvalue λ with each cycle. After 23 cycles, weighted GMRES(5) has found the exact solution of $A\mathbf{x} = \mathbf{b}$. This example demonstrates that looking at the harmonic Ritz values alone is not enough for explaining convergence when A is a (highly) nonnormal matrix.

To still give some insight into the different behaviour of the unweighted and weighted GMRES variants, we visualize in Figure 3 (c) and (d) the entries of the residual vector after each cycle. The special structure of the Jordan matrix results in large residual entries being shifted up the vector with each cycle. With unweighted GMRES(5) the “support” of nonzero entries forms a band that gets wider with each cycle, eventually polluting all entries of the residual vector and causing the method to stagnate. GMRES(5) with Essai’s weighting (3), on the other hand, “cleans up” the entries of the residual vector which were large in the previous cycle because more weight is placed at those entries. Eventually, this WGMRES(5) variant finds the exact solution in the 24-th cycle. The random weighting matrix D_{rand} leads to stagnation just as unweighted GMRES(5).

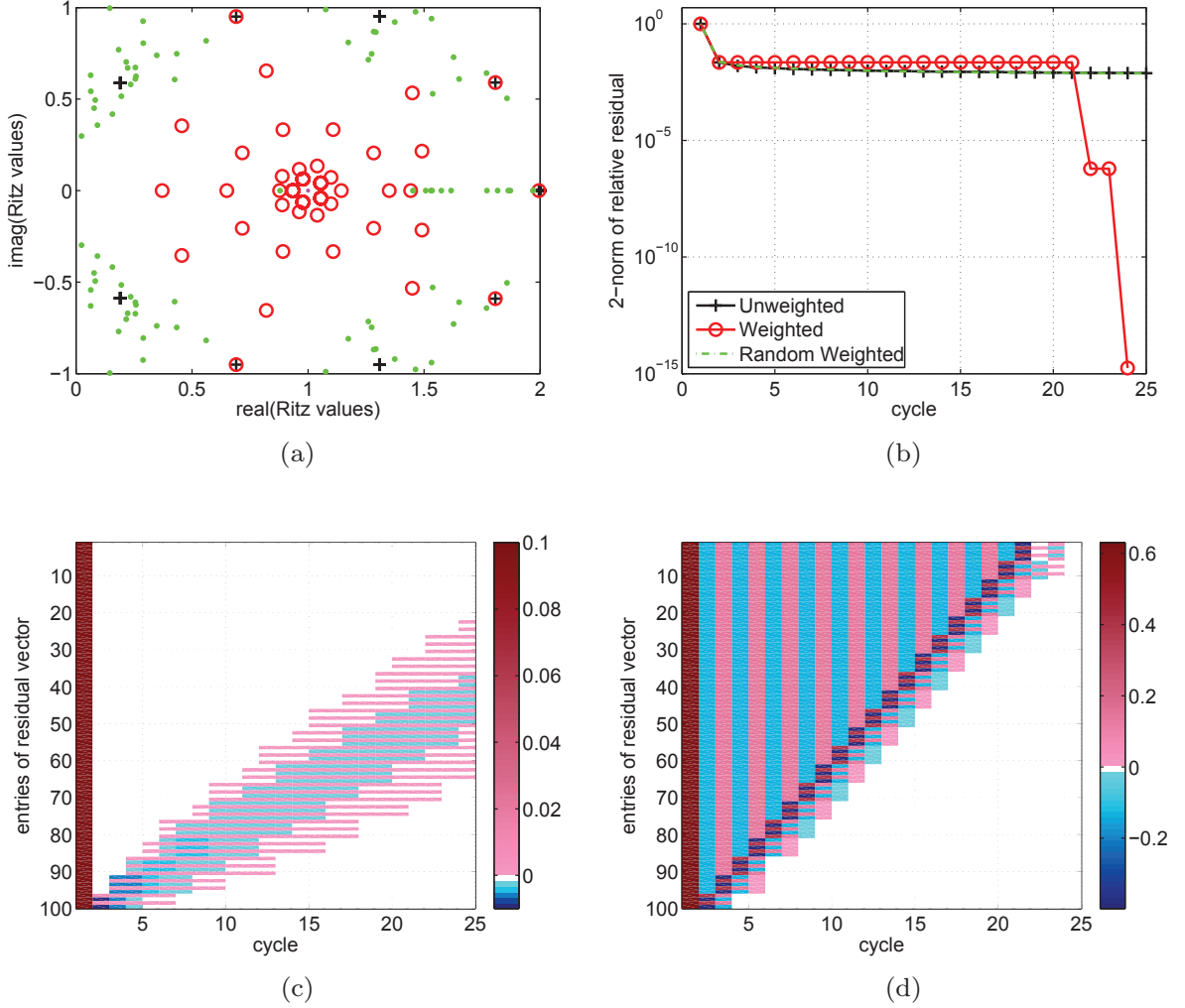


Fig. 3: (a) Harmonic Ritz values for GMRES(5) (black +), WGMRES(5) with (3) (red o), and WGMRES(5) with D_{rand} (green dots) for a Jordan block with eigenvalue 1. The harmonic Ritz values are shown at the end of each of the 25 cycles. The eigenvalue at 1 is plotted as an orange dot. (b) Relative 2-norm residuals for the various GMRES(5) variants. (c) Entries of the residual vectors after each cycle of unweighted GMRES(5). (d) Entries of the residual vectors after each cycle of GMRES(5) with Essai's weighting.

We end this section by remarking on the observations of Cao and Yu [4], who found little benefit in using WGMRES(m) on ILU-preconditioned problems unless the restart length was short. In particular, they found that although the number of cycles of WGMRES(m) was slightly lower than that required by GMRES(m) the CPU time of WGMRES(m) was greater. It is known that the application of ILU-preconditioners often clusters eigenvalues. In such cases weighting—the effect of which is typically to shift the harmonic Ritz values from those obtained by GMRES(m)—appears to offer little benefit. We have found that, for a number of linear systems right-preconditioned

by ILU(0), WGMRES(m) requires a number of cycles that is similar to, or slightly greater than, that required by GMRES(m) (see Example 9). However, WGMRES(m) may be of some benefit when the restart length is short or when the spectrum of the preconditioned matrix is not nicely clustered.

3 The weighted Arnoldi algorithm

In this section we discuss different variants of the weighted Arnoldi algorithm for constructing the D -orthonormal basis required by WGMRES.

3.1 Variants of the algorithm

The most straightforward implementation of the weighted Arnoldi algorithm replaces Euclidean inner products in a standard Arnoldi algorithm with D -inner products (see, e.g., Essai [10], Sarkis and Szyld [25]). The j -th iteration of such a weighted Arnoldi algorithm with modified Gram–Schmidt orthogonalization (MGS) requires the computation of j D -inner products (see Algorithm 1). If D is a diagonal matrix such as (3), the inner products in Algorithm 1 can be efficiently implemented as $(\mathbf{v} \circ \mathbf{d})^H \mathbf{u}$ or $\mathbf{v}^H (\mathbf{d} \circ \mathbf{u})$, where \circ represents the Hadamard product and \mathbf{d} the vector of diagonal elements of D . Algorithm 2 shows that when using the classical Gram–Schmidt orthogonalization (CGS) instead, the nonstandard inner-products can be replaced by Euclidean inner products at the expense of two matrix-vector products with D ; these can be computed as Hadamard products of the form $\mathbf{d} \circ \mathbf{w}$ when D is diagonal. In addition, the CGS method is more easily parallelizable and may be faster in practice, as it can make use of higher level BLAS routines. Even so, each step of weighted Arnoldi with Algorithms 1 or 2 is more expensive than a step of the Arnoldi algorithm in the Euclidean inner product because of the D -inner products and D -norms. The number of multiplications by D is fixed for the CGS version but increases as k increases in the MGS algorithm. Note that Algorithms 1 and 2 can be used in combination with a preconditioner in a straightforward manner.

As an alternative to computing D -inner products at each step of the weighted Arnoldi algorithm, we can perform the Arnoldi algorithm in the Euclidean inner product on a row- and column-scaled matrix and a scaled starting vector. The resulting algorithm with MGS orthogonalization is given in Algorithm 3; Algorithm 4 is the CGS version. Again, the CGS version is more amenable to parallelization and can utilize higher level BLAS, although the number of (Euclidean) inner products is identical to that required by the MGS version. We note that orthogonalizing a transformed matrix with respect to the Euclidean inner product, to obtain a basis that is orthogonal with respect to a nonstandard inner product, was considered in the context of rounding error analysis in [22]. Additionally, Heyouni and Essai [14] considered the use of matrix square roots for enforcing D -orthogonality when solving systems with

Variants of the weighted Arnoldi algorithm:

Inputs: Matrix $A \in \mathbb{C}^{N \times N}$, diagonal positive definite weight matrix $D \in \mathbb{C}^{N \times N}$, vector $\mathbf{r} \in \mathbb{C}^N$, number of Arnoldi iterations m

Outputs: D -orthonormal Arnoldi vectors $\{\mathbf{v}_1 \dots, \mathbf{v}_m\}$ of $\mathcal{K}_m(A, \mathbf{r})$ and upper Hessenberg matrix $\underline{H}_m \in \mathbb{C}^{(m+1) \times m}$

Algorithm 1: Explicit D inner products and MGS orthogonalization

```

 $\mathbf{v}_1 = \mathbf{r}$ 
for  $k = 1, 2, \dots, m$  do
   $\mathbf{w} = A\mathbf{v}_k$ 

  for  $j = 1, \dots, k$  do
     $h_{jk} = \mathbf{v}_j^* D \mathbf{w}$ 
     $\mathbf{w} = \mathbf{w} - h_{jk} \mathbf{v}_j$ 
  end
   $h_{k+1,k} = \|\mathbf{w}\|_D$ 
  if  $h_{k+1,k} = 0$  then
    Stop
  end
   $\mathbf{v}_{k+1} = \mathbf{w} / h_{k+1,k}$ 
end

```

Algorithm 2: Explicit D inner products and CGS orthogonalization

```

 $\mathbf{v}_1 = \mathbf{r}$ 
for  $k = 1, 2, \dots, m$  do
   $\mathbf{w} = A\mathbf{v}_k$ 
   $\mathbf{y} = D\mathbf{w}$ 
  for  $j = 1, \dots, k$  do
     $h_{jk} = \mathbf{v}_j^* \mathbf{y}$ 
     $\mathbf{w} = \mathbf{w} - h_{jk} \mathbf{v}_j$ 
  end
   $h_{k+1,k} = \|\mathbf{w}\|_D$ 
  if  $h_{k+1,k} = 0$  then
    Stop
  end
   $\mathbf{v}_{k+1} = \mathbf{w} / h_{k+1,k}$ 
end

```

Algorithm 3: Implicit D inner products and MGS orthogonalization

```

 $\tilde{A} = D^{\frac{1}{2}} A D^{-\frac{1}{2}}$ 
 $\tilde{\mathbf{v}}_1 = D^{\frac{1}{2}} \mathbf{r}$ 
for  $k = 1, 2, \dots, m$  do
   $\mathbf{w} = \tilde{A} \tilde{\mathbf{v}}_k$ 

  for  $j = 1, \dots, k$  do
     $h_{jk} = \tilde{\mathbf{v}}_j^* \mathbf{w}$ 
     $\mathbf{w} = \mathbf{w} - h_{jk} \tilde{\mathbf{v}}_j$ 
  end
   $h_{k+1,k} = \|\mathbf{w}\|_2$ 
  if  $h_{k+1,k} = 0$  then
    Stop
  end
   $\tilde{\mathbf{v}}_{k+1} = \mathbf{w} / h_{k+1,k}$ 
end

```

$[\mathbf{v}_1, \dots, \mathbf{v}_m] = D^{-\frac{1}{2}} [\tilde{\mathbf{v}}_1, \dots, \tilde{\mathbf{v}}_m]$

Algorithm 4: Implicit D inner products and CGS orthogonalization

```

 $\tilde{A} = D^{\frac{1}{2}} A D^{-\frac{1}{2}}$ 
 $\tilde{\mathbf{v}}_1 = D^{\frac{1}{2}} \mathbf{r}$ 
for  $k = 1, 2, \dots, m$  do
   $\mathbf{w} = \tilde{A} \tilde{\mathbf{v}}_k$ 
   $\mathbf{y} = \mathbf{w}$ 
  for  $j = 1, \dots, k$  do
     $h_{jk} = \tilde{\mathbf{v}}_j^* \mathbf{y}$ 
     $\mathbf{w} = \mathbf{w} - h_{jk} \tilde{\mathbf{v}}_j$ 
  end
   $h_{k+1,k} = \|\mathbf{w}\|_2$ 
  if  $h_{k+1,k} = 0$  then
    Stop
  end
   $\tilde{\mathbf{v}}_{k+1} = \mathbf{w} / h_{k+1,k}$ 
end

```

$[\mathbf{v}_1, \dots, \mathbf{v}_m] = D^{-\frac{1}{2}} [\tilde{\mathbf{v}}_1, \dots, \tilde{\mathbf{v}}_m]$

multiple right-hand sides, although they still computed D -inner products at each iteration of their weighted Arnoldi algorithm. It does not seem feasible to use Algorithms 3 and 4 in combination with a preconditioner.

It is straightforward to verify that, in exact arithmetic, Algorithms 1 and 2 are equivalent to Algorithms 3 and 4. Applying the Arnoldi algorithm in the Euclidean inner product with the transformed matrix $\tilde{A} = D^{\frac{1}{2}}AD^{-\frac{1}{2}}$ and starting vector $\tilde{\mathbf{r}} = D^{\frac{1}{2}}\mathbf{r}$ gives matrices \tilde{V}_m and \tilde{H}_m such that

$$\tilde{A}\tilde{V}_m = \tilde{V}_{m+1}\tilde{H}_m, \quad (5)$$

where \tilde{H}_m is an upper Hessenberg matrix, $\tilde{V}_m^H\tilde{V}_m = I_m$, and the columns of \tilde{V}_m form a basis of $\mathcal{K}_m(\tilde{A}, \tilde{\mathbf{r}})$. Premultiplying both sides of (5) by $D^{-\frac{1}{2}}$ gives the Arnoldi decomposition

$$AV_m = V_{m+1}H_m,$$

where $V_m = D^{-\frac{1}{2}}\tilde{V}_m$ and $H_m = \tilde{H}_m$. The columns of $V_m = D^{-\frac{1}{2}}\tilde{V}_m$ are D -orthonormal since $V_m^H DV_m = \tilde{V}_m^H \tilde{V}_m = I_m$. Moreover, since the columns of \tilde{V}_m span $\mathcal{K}_m(\tilde{A}, \tilde{\mathbf{r}}) \equiv \mathcal{K}_m(D^{\frac{1}{2}}AD^{-\frac{1}{2}}, D^{\frac{1}{2}}\mathbf{r})$, the columns of $V_m = D^{-\frac{1}{2}}\tilde{V}_m$ span $\mathcal{K}_m(A, \mathbf{r})$. We therefore conclude that Algorithms 1 and 2 are mathematically equivalent to Algorithms 3 and 4.

In the remainder of this manuscript, we use Algorithms 1–4 to refer to both the different weighted Arnoldi variants and the corresponding WGMRES(m) methods. The meaning will be clear from the context.

3.2 Operation counts

In all four variants of the Arnoldi algorithm discussed here, the number of matrix-vector products is m , and their computation requires $2m \times \text{Nnz}$ arithmetic operations, where Nnz is the number of nonzero elements in the matrix A or \tilde{A} , respectively. Furthermore, the successive orthogonalization of m Krylov basis vectors requires $m(m+1)/2$ inner products and vector updates of the form $\mathbf{w} = \mathbf{w} - \mathbf{v}_j h_{j,k}$. One such vector update requires $2N$ arithmetic operations. Computing a single inner product requires $2N$ or $3N$ arithmetic operations in the unweighted or weighted case, respectively. The scaling of the matrix to form \tilde{A} and vector \mathbf{r} to form $\tilde{\mathbf{r}}$ in Algorithms 3 and 4 requires $2N + 2 \times \text{Nnz}$ operations, counting the computation of a square root as a single arithmetic operation. Algorithms 3 and 4 also require Nm multiplications to scale the basis vectors at the end of the weighted Arnoldi algorithm.

In Section 4 we will compare different variants of WGMRES(m) with GMRES-DR(m, ℓ) [18] on various numerical examples, but it is easy to discuss the computational costs theoretically as well. The GMRES-DR(m, ℓ) method augments the Krylov basis computed by GMRES(m) with the Schur vectors corresponding to the ℓ smallest harmonic Ritz values. These Schur vectors, and the initial residual of the new cycle k , form the columns of a matrix

$P_{\ell+1} \in \mathbb{C}^{(m+1) \times (\ell+1)}$, from which the first $\ell + 1$ basis vectors of the new cycle are obtained via

$$V_{\ell+1}^{(k)} = V_{m+1}^{(k-1)} P_{\ell+1}. \quad (6)$$

The remaining columns of $V_{m+\ell}^{(k)}$ are then computed by the Arnoldi algorithm. The dense matrix-matrix multiplication (6) requires $2N(m+1)(\ell+1)$ arithmetic operations and so dominates the cost of augmenting the Krylov basis. Depending on the restart length m and the number of Schur vectors ℓ , the cost of this product can make GMRES-DR(m, ℓ) more expensive than WGMRES(m).

We summarize the operation counts of Algorithms 1–4 in the following table.

Algorithm 1	$2m \times \text{Nnz} + \frac{5}{2}Nm^2$
Algorithm 2	$2m \times \text{Nnz} + 2Nm^2 + 2Nm$
Algorithm 3	$2m \times \text{Nnz} + 2Nm^2 + 2N + 2 \times \text{Nnz} + Nm$
Algorithm 4	$2m \times \text{Nnz} + 2Nm^2 + 2N + 2 \times \text{Nnz} + Nm$
GMRES-DR	$2m \times \text{Nnz} + 2Nm^2 + 2Nm\ell$

Although the performance of each algorithm is machine dependent, Algorithms 3 and 4, for which the cost of the weighted inner product is independent of the restart length, can become more efficient than Algorithms 1 and 2 in exact arithmetic when the restart length increases and the number of nonzeros Nnz is sufficiently small. However, the numerical stability of these four variants can be quite different, as discussed in the next subsection.

3.3 Numerical stability

It is well known that classical Gram–Schmidt orthogonalization in the Euclidean inner product is less stable than modified Gram–Schmidt orthogonalization (see, for example, [21]). Our experience is that D -orthogonalization, when D is given by (3), also becomes less stable by CGS orthogonalization than by MGS orthogonalization, i.e., Algorithms 2 and 4 are less stable than Algorithms 1 and 3. These observations are supported by bounds on

$$\|I_m - V_m^H D V_m\|_2. \quad (7)$$

obtained by Rozložník *et al.* [22]. The MGS bound depends on $\kappa = \kappa_2(D^{\frac{1}{2}}K)$, where

$$K = [\mathbf{r}, A\mathbf{r}, \dots, A^{(m-1)}\mathbf{r}],$$

while the CGS bound depends on κ^2 and, consequently is much worse than the MGS bound when κ is large.

In WGMRES(m) this loss of orthogonality principally affects the small least squares problem that must be solved. By the minimization property of WGMRES(m), at the k -th cycle the residual vector satisfies

$$\begin{aligned}\|\mathbf{r}^{(k)}\|_D &= \min_{\mathbf{y} \in \mathbb{C}^m} \|V_{m+1}(\underline{H}_m \mathbf{y} - \|\mathbf{r}^{(k-1)}\|_D \mathbf{e}_1)\|_D \\ &= \min_{\mathbf{y} \in \mathbb{C}^m} \|\underline{H}_m \mathbf{y} - \|\mathbf{r}^{(k-1)}\|_D \mathbf{e}_1\|_{V_m^H D V_m},\end{aligned}$$

where, although all matrices change with each cycle, we drop the superscript k for clarity. Provided the columns of V_m are D -orthogonal,

$$\mathbf{r}^{(k)} = \arg \min_{\mathbf{y} \in \mathbb{C}^m} \|\underline{H}_m \mathbf{y} - \|\mathbf{r}^{(k-1)}\|_D \mathbf{e}_1\|_2$$

and it is this small least squares problem that is solved in WGMRES(m) to update the iterate and residual vector. However, if $V_m^H D V_m \neq I_m$, then

$$\hat{\mathbf{r}}^{(k)} = \arg \min_{\mathbf{y} \in \mathbb{C}^m} \|\underline{H}_m \mathbf{y} - \|\mathbf{r}^{(k-1)}\|_D \mathbf{e}_1\|_2$$

is only an approximation of \mathbf{r}_m . However, for any vector $\mathbf{r} \in \mathbb{C}^m$,

$$\sigma_{\min}(D^{\frac{1}{2}} V_m) \|\mathbf{r}\|_2 \leq \|\mathbf{r}\|_{V_m^H D V_m} \leq \sigma_{\max}(D^{\frac{1}{2}} V_m) \|\mathbf{r}\|_2, \quad (8)$$

where $\sigma_{\min}(D^{\frac{1}{2}} V_m)$ and $\sigma_{\max}(D^{\frac{1}{2}} V_m)$ are the smallest and largest singular values of $D^{\frac{1}{2}} V_m$. This shows that $\hat{\mathbf{r}}^{(k)}$ and $\mathbf{r}^{(k)}$ can only differ significantly when the singular values of $D^{\frac{1}{2}} V_m$ start to differ from 1.

A detailed analysis of the effect of loss of orthogonality on WGMRES(m) convergence appears complicated. Of course, convergence may be unaffected, similarly to GMRES in the Euclidean inner product (see, for example, Simoncini and Szyld [26], Greenbaum et al [13] and Drkošová *et al.* [7] for examples and analysis of the convergence of GMRES in finite precision). However, there are certainly cases for which convergence is affected by the orthogonalization procedure. Moreover, this effect is influenced by the variant of the weighted Arnoldi algorithm that is used, and becomes more pronounced with large restart lengths. To see this, we apply WGMRES(70) to the matrix `fs_541_2` of size $N = 541$, which is available from the University of Florida Sparse Matrix Collection [6]. We choose \mathbf{b} such that the solution is $\mathbf{x} = [1, 1, \dots, 1]^T / \sqrt{N}$, and use the initial guess $\mathbf{x}^{(0)} = \mathbf{0}$. Figure 4 shows the relative residuals $\|\mathbf{r}^{(k)}\|_2 / \|\mathbf{b}\|_2$ at the end of each cycle, and the loss of D -orthogonality of V_m in (2) at the end of each weighted Arnoldi run, measured by (7). We note that the difference between $\sigma_{\min}(D^{\frac{1}{2}} V_m)$ and 1, and between $\sigma_{\max}(D^{\frac{1}{2}} V_m)$ and 1, is approximately equal to (7) for this problem.

We first apply WGMRES(m) with Algorithm 1 (which uses MGS orthogonalization), and Algorithm 2 (which uses CGS orthogonalization). It is clear from Figure 4 (a) that the CGS basis vectors are further from D -orthogonality than those computed by the MGS weighted Arnoldi algorithm. Moreover, when the relative residuals for the CGS implementation reach the level of

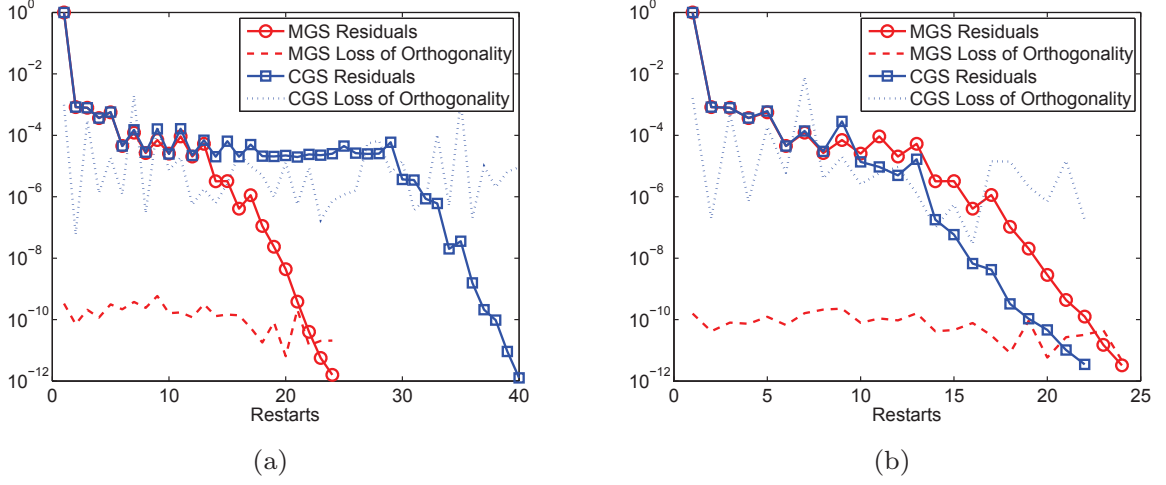


Fig. 4: Relative residuals at the end of each cycle of WGMRES(70) using (a) Algorithms 1 and 2 and (b) Algorithms 3 and 4 with MGS (solid red with \circ) and CGS (solid blue with \square) D -orthogonalization. Also plotted is $\|I_m - V_m^H D V_m\|_2$ at the end of each weighted Arnoldi cycle with MGS (dashed red) and CGS (dotted blue) orthogonalization.

$\|I_m - V_m^H D V_m\|_2$, there is a noticeable delay in convergence. A partial explanation for this delay is given by (8); when the difference between $\sigma_{\max}(D^{\frac{1}{2}} V_m)$ or $\sigma_{\min}(D^{\frac{1}{2}} V_m)$ and 1 is of the same order as the norm of the relative residual, we might expect $\mathbf{r}^{(k)}$ and $\hat{\mathbf{r}}^{(k)}$ to differ.

When WGMRES(m) with Algorithm 3 (MGS) and Algorithm 4 (CGS) is used, we also find that the loss of orthogonality is more pronounced for the CGS basis vectors (see Figure 4 (b)). However, now convergence is slightly faster with CGS orthogonalization. The convergence delay for the MGS variant of weighted Arnoldi is certainly less pronounced than in Figure 4 (a), yet it emphasizes that numerical stability depends on the combination of the orthogonalization procedure and the way in which D -inner products are computed.

In the following section we compare the numerical stability of MGS and CGS orthogonalization for a number of examples. It is found that neither is definitively more stable than the other, and a more detailed analysis of their finite precision behaviour would be required to assess the merits of each. Further complicating matters, the relationship between the choice of the weighted Arnoldi variant and the orthogonalization procedure (MGS or CGS) appears to influence stability. However, from numerical experimentation (see Section 4), we find the modified Gram–Schmidt procedure to be more stable in general, particularly when comparing Algorithms 1 and 2.

m	GMRES-DR($m, 5$)	GMRES-DR($m, 10$)	Alg 1	Alg 2	Alg 3	Alg 4
40	20(23)	18(20)	21(35)	21(35)	21(35)	21(35)
50	14(44)	14(5)	15(32)	15(32)	15(32)	15(32)
60	11(49)	11(26)	12(43)	12(43)	12(43)	12(43)
70	10(36)	10(15)	11(12)	11(12)	11(12)	11(12)
80	9(12)	8(69)	10(5)	10(3)	10(5)	10(5)

Table 1: Number of cycles, with the number of iterations in the last cycle given in parentheses, for `add20`.

4 Numerical experiments

In this section we compare different variants of WGMRES(m) with each other and with more established Krylov subspace methods, such as GMRES with deflated restarting (GMRES-DR) and BICGSTAB. The first five examples are those used by Essai [10] and are available from the University of Florida Sparse Matrix Collection [6]. For these, we compare the four variants of WGMRES(m) in Algorithms 1–4 with each other and with GMRES-DR(m, ℓ). The last example compares two WGMRES(m) variants, unweighted GMRES(m), GMRES-DR(m, ℓ) and BICGSTAB over a large set of right-preconditioned test problems that is described below.

In all examples we take $\mathbf{x}^{(0)} = \mathbf{0}$ as the initial guess. A method is stopped when the relative residual $\|\mathbf{r}_j^{(k)}\|_2 / \|\mathbf{b}\|_2$ has decreased to below 10^{-10} . If this tolerance is not reached after 100 cycles we terminate the method, which we denote by ‘—’. Iteration counts are given in the form $it_{\text{out}}(it_{\text{in}})$, where it_{out} is the number of cycles and it_{in} is the number of steps in the last cycle. In our notation GMRES-DR(m, ℓ) augments a Krylov subspace of dimension m with ℓ approximate Schur vectors. This makes sure that all GMRES variants require exactly m matrix-vector products with A in a cycle.

Example 4 (add20) *Our first test problem is `add20`, a matrix from a circuit simulation problem of dimension 2395 with 13151 nonzeros. The right-hand side \mathbf{b} is a random vector. Table 1 shows the number of cycles required by GMRES-DR(m, ℓ) with $\ell = 5, 10$ as well as WGMRES(m) with Algorithms 1–4. We first observe that GMRES-DR requires slightly fewer cycles than WGMRES for all choices of m ($m = 40, 50, \dots, 80$). Comparing the WGMRES variants, we find that for this example all algorithms require the same number of cycles, except for $m = 80$ for which Algorithm 2 converges slightly faster. Overall, however, it appears that for this example all methods behave stably.*

Example 5 (bfwa782) *The matrix `bfwa782` comes from an electromagnetics problem and is of dimension 782 with 7514 nonzeros. Again \mathbf{b} is a random vector. The number of cycles for the different methods are given in Table 2. We see that the difference between GMRES-DR and WGMRES is more pronounced, with the former requiring significantly fewer cycles for convergence. When $m = 40$, Algorithm 1 (with MGS orthogonalization) requires four fewer cycles than the other WGMRES variants. As the restart length increases, how-*

m	GMRES-DR($m, 5$)	GMRES-DR($m, 10$)	Alg 1	Alg 2	Alg 3	Alg 4
40	17(13)	9(2)	65(34)	69(20)	69(24)	69(24)
50	11(34)	7(23)	34(26)	34(26)	34(27)	34(27)
60	8(33)	6(18)	28(58)	28(58)	28(58)	28(58)
70	7(42)	5(36)	22(5)	22(5)	22(5)	22(5)
80	6(46)	4(70)	14(56)	14(56)	14(56)	14(56)

Table 2: Number of cycles, with the number of iterations in the last cycle given in parentheses, for **bfwa782**.

m	GMRES-DR($m, 5$)	GMRES-DR($m, 10$)	Alg 1	Alg 2	Alg 3	Alg 4
40	35(27)	29(21)	—	—	—	—
50	22(18)	20(49)	23(47)	23(33)	23(42)	23(42)
60	18(49)	16(53)	25(60)	24(24)	25(60)	25(60)
70	13(48)	12(62)	21(65)	37(60)	22(9)	22(9)
80	10(78)	9(76)	—	—	—	—

Table 3: Number of cycles, with the number of iterations in the last cycle given in parentheses, for **fs_541_2**.

ever, the different WGMRES implementations behave similarly and require identical numbers of cycles to converge.

Example 6 (fs_541_2) Our next example **fs_541_2** has dimension 541 and 4282 nonzeros, with **b** a random vector. Table 3 again shows that GMRES-DR converges faster than WGMRES, with the latter failing to converge after 100 cycles when $m = 40, 80$. For restart lengths $m = 50, 60$, Algorithm 2 (with CGS orthogonalization) requires fewest matrix-vector products among the WGMRES variants. However, when $m = 70$ convergence for the same algorithm is significantly delayed (see also Figure 4). Although CGS orthogonalization can give slightly faster convergence, it does not appear to be as stable as MGS orthogonalization.

Example 7 (memplus) The matrix **memplus** is another circuit simulation problem, having dimension 17758 and 99147 nonzeros. The vector **b** is chosen at random. We see from Table 4 that for restart lengths smaller than $m = 80$, WGMRES(m) converges in fewer cycles than GMRES-DR($m, 5$). GMRES-DR($m, 10$) outperforms WGMRES, except when Algorithms 3 and 4 are applied with $m = 50$.

Considering now the different WGMRES implementations, we see that there is some variation between them. For this problem WGMRES with Algorithms 3 and 4 require fewer cycles than WGMRES with Algorithms 1 or 2 when $m = 50, 60, 80$ and more when $m = 40, 70$.

Example 8 (orsirr_1) A matrix from oil reservoir simulation, **orsirr_1** has dimension 1030 and 6858 nonzeros, and **b** is a random vector. This is the one example for which WGMRES outperforms GMRES-DR($m, 10$) (see Table 5). For smaller restart lengths, WGMRES(m) also converges faster than GMRES-DR($m, 5$).

m	GMRES-DR($m, 5$)	GMRES-DR($m, 10$)	Alg 1	Alg 2	Alg 3	Alg 4
40	60(40)	48(32)	50(24)	52(5)	54(29)	54(29)
50	42(27)	33(41)	35(33)	36(44)	33(2)	33(2)
60	33(9)	25(54)	28(31)	28(22)	27(60)	27(60)
70	25(51)	21(50)	26(26)	24(34)	26(37)	26(37)
80	20(51)	18(2)	20(49)	21(66)	20(9)	20(9)

Table 4: Number of cycles, with the number of iterations in the last cycle given in parentheses, for `memplus`.

m	GMRES-DR($m, 5$)	GMRES-DR($m, 10$)	Alg 1	Alg 2	Alg 3	Alg 4
40	70(9)	—	61(35)	61(21)	58(2)	69(37)
50	55(13)	—	46(46)	45(32)	48(11)	48(41)
60	34(55)	—	35(41)	34(42)	34(32)	35(30)
70	25(12)	—	28(39)	27(54)	28(18)	27(41)
80	—	—	22(32)	24(31)	23(8)	24(13)

Table 5: Number of cycles, with the number of iterations in the last cycle given in parentheses, for `orsirr_1`.

Different implementations of WGMRES again require different numbers of cycles for convergence, particularly for smaller restart lengths. The best performing WGMRES(m) variant changes with the restart length, so that overall none outperforms the others. We note in particular that when $m = 40$ Algorithm 3 converges fastest while Algorithm 4, which uses CGS orthogonalization, requires 11 more cycles, again highlighting the impact of the orthogonalization method on performance.

Example 9 (performance profile) *In order to get a general idea of whether WGMRES is a practical method when preconditioners are used, we compare GMRES, WGMRES with modified and classical Gram–Schmidt orthogonalization (see Algorithms 1 and 2), as well as GMRES-DR and BICGSTAB on a large number of problems from the University of Florida Sparse Matrix Collection. In contrast to Cao and Yu [4] we use right-preconditioning, which minimizes the norm of the residual vector, rather than left-preconditioning, which minimizes the norm of the preconditioned residual. The reason is that typically the residual vector is of greater interest than the preconditioned residual.*

Our method of comparison is as follows. We first retrieve all nonsymmetric matrices A of sizes between 10^4 and 10^6 having no more than 15 nonzero elements per row on average. There are 220 such matrices. We then apply sparse reverse Cuthill–McKee reordering as implemented in Matlab’s `symrcm`. Next we scale the columns of A to have unit Euclidean norm, followed by a scaling of the rows of A to unit norm. Our aim is to compute an ILU preconditioner with thresholding and pivoting via Matlab’s `ilu`. For stability reasons we compute an ILU factorization of $A + \sigma I$, where $\sigma = 10^{-12}$ if all diagonal elements of A are zero, or $\sigma = 10^{-12} \max\{|a_{ii}|\}$ if some but not all diagonal elements a_{ii} of A are zero, or $\sigma = 0$ otherwise. This procedure follows some recommendations given in [5]. We successively use a drop tolerance of $10^{-3}, 10^{-4}, \dots, 10^{-8}$,

and stop when the U factor of the factorization has a condition number below 10^{15} , so that it can be assumed numerically nonsingular. If this condition is not satisfiable with a drop tolerance of 10^{-8} , then the matrix A is skipped. The right-hand side vector \mathbf{b} either comes with the matrix from the collection, or is generated randomly.

We now run the Krylov methods GMRES(10), the two WGMRES(10) variants, GMRES-DR(10,5), BICGSTAB on the test matrices. A method is marked as failed if more than 50 restart cycles or 250 BICGSTAB iterations are required to obtain a relative residual norm of 10^{-8} . Note that in our notation GMRES-DR(10,5) requires 10 MVP per cycle, exactly like GMRES(10) and WGMRES(10), and BICGSTAB requires 2 MVP per iteration, so that the maximal number of MVP is 500 for all methods. If all methods fail on a matrix, then this matrix is excluded from the test. Since the collection also contains many singular matrices, related to eigenvalue or least-squares problems, only 109 out of the 220 retrieved matrices are finally included in our test.

The performance profile in Figure 5 allows us to compare the number of MVP needed for each method to converge across all test problems. More specifically, if for each linear system the performance ratio measures the number of MVP for the k -th method to converge to the number of MVP for the best performing method to converge, then the function $f_k(\alpha)$ measures the fraction of problems in the test set for which the performance ratio of method k is less than or equal to α . Thus, $\alpha = 1$ shows the fraction of problems for which the k -th method requires the fewest MVP of all methods. Also, $\lim_{\alpha \rightarrow \infty} f_k(\alpha)$ indicates the number of failures.

It is somewhat disappointing that WGMRES(10) does not generally outperform GMRES(10). However, we note that the CGS version does not perform considerably worse than the MGS version of WGMRES and, considering the potential computational savings outlined in Section 3, it seems that one should generally favour the CGS version when using WGMRES. We note that GMRES(10) and both variants of WGMRES(10) fail on at least 16 matrices from our test set; this failure rate is considerably higher than for GMRES-DR and BICGSTAB. Overall, GMRES-DR(10,5) requires fewest MVP in general, and thereby outperforms all other methods under consideration. It is, however, less robust than BICGSTAB, the latter of which fails for 3 matrices only but typically requires the most MVP. To summarize, we believe that WGMRES should not be used in combination with preconditioners, although we are aware that for some examples it may perform satisfactorily.

5 Conclusions

The weighted GMRES variant presented by Essai has recently gained interest for solving linear systems. This method is justified by a heuristic that emphasizes large residual components via a weighted inner product. With the help of simple model problems we have given insight into how weighting affects the distribution of harmonic Ritz values, or how it affects entries in the residual

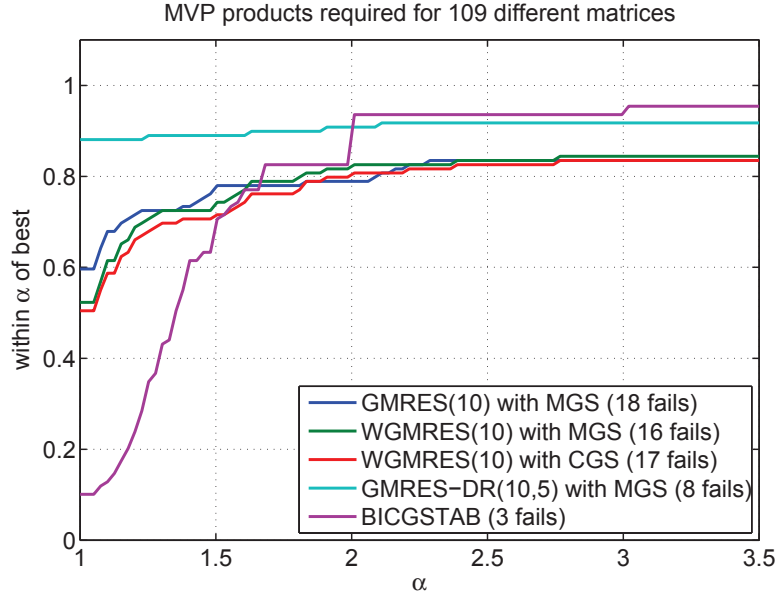


Fig. 5: Performance profile of matrix vector products required by various Krylov methods applied to 109 matrices from the University of Florida Sparse Matrix Collection with an ILU preconditioner.

vector after each cycle. For example, in one case where the harmonic Ritz values appeared in cyclic pairs on the spectral interval of a matrix, weighting had the effect of “randomizing” these Ritz values and thereby covering the spectral interval more evenly. This led to an improved convergence of WGMRES compared to the linear convergence observed for GMRES on the same example. We presented four different implementations of weighted GMRES and compared their numerical cost and stability properties. Our numerical results suggest that these variants generally converge similarly to Essai’s original implementation, but may require fewer arithmetic operations. When applied to some unpreconditioned problems, WGMRES(m) can outperform GMRES(m). A test of weighted GMRES applied to a large number of problems from the University of Florida Sparse Matrix Collection revealed, similarly to observations made in [4], that weighted GMRES is typically outperformed by GMRES if a preconditioner is used. In addition, we have compared these methods with other state-of-the-art Krylov methods like GMRES-DR with deflated restarting and BICGSTAB. GMRES-DR required fewest matrix-vector products, whereas BICGSTAB appeared to be the most robust method in our test, at the cost of requiring the most matrix-vector products. One advantage of WGMRES(m) over GMRES-DR(m, ℓ) is that there is no parameter ℓ to be chosen.

We find that, although weighted GMRES may outperform unweighted GMRES for some examples, in general this method is not competitive with other Krylov subspace methods like BICGSTAB or deflated GMRES, in particular when preconditioners are used. If, for whatever reason, weighted GMRES

needs to be used with a moderate restart length, then the CGS version (Algorithm 2) presented in Section 3 is recommended.

Acknowledgements We are grateful to Andy Wathen for inspiring this work and giving valuable comments.

References

1. Afanasjew, M., Eiermann, M., Ernst, O.G., Güttel, S.: A generalization of the steepest descent method for matrix functions. *Electron. Trans. Numer. Anal.* **28**, 206–222 (2008)
2. Akaike, H.: On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Ann. Inst. Statist. Math.* **11**, 1–16 (1959)
3. Baker, A., Jessup, L., Manteuffel, T.: A technique for accelerating the convergence of restarted GMRES. *SIAM J. Matrix Anal. Appl.* **26**, 962–984 (2005)
4. Cao, Z.H., Yu, X.Y.: A note on weighted FOM and GMRES for solving nonsymmetric linear systems. *Appl. Math. Comput.* **151**, 719–727 (2004)
5. Chow, E., Saad, Y.: Experimental study of ILU preconditioners for indefinite matrices. Tech. rep., Department of Computer Science and Minnesota Supercomputer Institute (1997)
6. Davis, T.A., Hu, Y.: The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.* **38**, 1:1–1:25 (2011)
7. Drkošová, J., Grenbaum, A., Rozložník, M., Strakoš, Z.: Numerical stability of GMRES. *BIT* **35**, 309–330 (1995)
8. Eiermann, M., Ernst, O.G., Schneider, O.: Analysis of acceleration strategies for restarted minimal residual methods. *J. Comput. Appl. Math.* **123**, 261–292 (2000)
9. Embree, M.: The tortoise and the hare restart GMRES. *SIAM Rev.* **45**, 259–266 (2003)
10. Essai, A.: Weighted FOM and GMRES for solving nonsymmetric linear systems. *Numer. Algorithms* **18**, 277–292 (1998)
11. Forsythe, G.E., Motzkin, T.S.: Asymptotic properties of the optimum gradient method (abstract). *Bull. Am. Math. Soc.* **57**, 183 (1951)
12. Goossens, S., Roose, D.: Ritz and harmonic Ritz values and the convergence of FOM and GMRES. In: *Proceedings of the Conference on Preconditioned Iterative Solution Methods for Large Scale Problems in Scientific Computations*
13. Greenbaum, A., Rozložník, M., Strakoš, Z.: Numerical behaviour of the modified Gram-Schmidt GMRES implementation. *BIT* **37**, 706–719 (1997)
14. Heyouni, M., Essai, A.: Matrix Krylov subspace methods for linear systems with multiple right-hand sides. *Numer. Algorithms* **40**, 137–156 (2005)
15. Jing, Y.F., Huang, T.Z.: Restarted weighted full orthogonalization method for shifted linear systems. *Comput. Math. Appl.* **57**, 1583–1591 (2009)
16. Meijerink, J.A., van der Vorst, H.A.: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M -matrix. *Math. Comput.* **31**, 148–162 (1977)
17. Morgan, R.B.: A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.* **16**, 1154–1171 (1995)
18. Morgan, R.B.: GMRES with deflated restarting. *SIAM J. Sci. Comput.* **24**, 20–37 (2002)
19. Najafi, H.S., Zareamoghaddam, H.: A new computational GMRES method. *Appl. Math. Comput.* **199**, 527–535 (2008)
20. Niu, Q., Lu, L., Zhou, J.: Accelerate weighted GMRES by augmenting error approximations. *Int. J. Comput. Math.* **87**, 2101–2112 (2010)
21. Rice, J.R.: Experiments on Gram-Schmidt Orthogonalization. *Math. Comput.* **20**, 325–328 (1966)
22. Rozložník, M., Tůma, M., Smoktunowicz, A., Kopal, J.: Numerical stability of orthogonalization methods with a non-standard inner product. Tech. rep., Institute of Computer Science, Academy of Sciences of the Czech Republic (2011)

23. Saad, Y.: Analysis of augmented Krylov subspace methods. *SIAM J. Matrix Anal. Appl.* **18**, 435–449 (1997)
24. Saad, Y., Schultz, M.H.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* **7**, 856–869 (1986)
25. Sarkis, M., Szyld, D.B.: Optimal left and right additive Schwarz preconditioning for minimal residual methods with Euclidean and energy norms. *Comput. Methods Appl. Mech. Eng.* **196**, 1612–1621 (2007)
26. Simoncini, V., Szyld, D.B.: The effect of non-optimal bases on the convergence of Krylov subspace methods. *Numer. Math.* **100**, 711–733 (2005)
27. Simoncini, V., Szyld, D.B.: On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods. *SIAM Rev.* **47**, 247–272 (2005)
28. Vecharynski, E., Langou, J.: The cycle-convergence of GMRES for normal matrices is sublinear. *SIAM J. Sci. Comput.* **32**, 186–196 (2010)
29. Vecharynski, E., Langou, J.: Any admissible cycle-convergence behavior is possible for restarted GMRES at its initial cycles. *Numer. Linear Algebra Appl.* **18**, 499–511 (2011)
30. Wang, Z., Qi, J., Liu, C., Li, Y.: Weighted harmonic Arnoldi method for large interior eigenproblems. *World Acad. Sci. Eng. Technol.* **80**, 1473–1476 (2011)
31. Zavorin, I., O’Leary, D.P., Elman, H.: Complete stagnation of GMRES. *Linear Algebra Appl.* **367**, 165–183 (2003)
32. Zhong, B., Morgan, R.B.: Complementary cycles of restarted GMRES. *Numer. Linear Algebra Appl.* **15**, 559–571 (2008)
33. Zhong, B.J.: A product hybrid GMRES algorithm for nonsymmetric linear systems. *J. Comput. Math.* **23**, 83–92 (2005)
34. Zhong, H.X., Wu, G.: Thick restarting the weighted harmonic Arnoldi algorithm for large interior eigenproblems. *Int. J. Comput. Math.* **88**, 994–1012 (2011)