GMRES AND WEIGHTED GMRES FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS

GUSTAV KRATZ

Bachelor's thesis
2016:K16

LUND UNIVERSITY

Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

Abstract

The Implementation and some mathematical properties of GMRES and weighted GMRES (WGMRES) are described. Numerical experiments originally performed by Essai (1998) are performed to compare the two methods. GMRES and WGMRES are used to solve a linear system arising from the Poisson equation and compared with respect to computational effort.

Acknowledgements

Thanks to my supervisor Philipp Birken. Solutions of large nonlinear and linear systems of equations are an inevitable part of his research area, making him an expert in the field. Without his guidance, careful reading and helpful suggestions this paper would not have been accomplished. Thanks to my examiner Carmen Arévalo for her helpfulness and valuable cooperation. Lastly I want to express my gratitude to all faculty members and teachers at the Centre for Mathematical Sciences at Lund University that made this thesis possible.

Contents

1 Introduction

GMRES and weighted GMRES (WGMRES) are Krylov subpsace methods suited for solving nonsymmetric linear systems

$$Ax = b.$$

It will be assumed throughout that $A \in \mathbb{R}^{n \times n}$ is invertible so that an exact solution $x = A^{-1}b$ exists for a given $b \in \mathbb{R}^n$. Linear systems arise frequently in scientific applications, not least in the numerical analysis of differential equations.

When solving a differential equation numerically, one is often encountered with the problem of solving a nonlinear system of equations. One way to approach such a problem is to solve a linear system iteratively until the error is within an acceptable error margin (e.g. the Newton Raphson method). The resulting linear systems often exhibit a sparse structure (most elements are zero). Given an efficient algorithm that exploits this structure when matrix-vector multiplications are carried out, GMRES is an efficient solver, being based mainly on matrix-vector multiplications.

A first step of treating a differential equation on a computer is to discretize the space where the equation is to be solved. For example, an interval $[0, T]$ can be discretized into n points $t_1 = 0, t_2, ..., t_n = T$, and the equation is then to be approximated at each of these points. Let $x = (x_1, x_2, ..., x_n)$ be an approximate solution to some differential equation . Let $E(x) = (e_1, e_2, ..., e_n)$ represent the difference between $x(t)$ and the exact solution on the middle of each interval. If the points t_k are equally spaced from one another, the l^2-norm of the error vector E is

$$l^2(E(x)) = \sqrt{\sum_{i=1}^{n} e_i^2} = \sqrt{E^T E},$$

However, some problems requires a non-equidistant discretization (e.g. *stiff problems*) and the l^2-norm becomes

$$l^2(E(x)) = \sqrt{\sum_{i=1}^{n} e_i^2 \Delta_{t_i}} = \sqrt{E^T D E},$$

where

$$d = (\Delta_{t_1}, \Delta_{t_2}, ..., \Delta_{t_n})$$

is the length of the intervals between each point and

$$D = \mathrm{diag}(d).$$

We are faced with a weighted scalar product

$$(u, v)_D = u^T D v = \sum_{i=1}^{n} d_i u_i v_i$$

unlike the Euclidean scalar product (where $D = I$). Measuring the error in the Euclidean scalar product would put equally much weight in all the components of E even though they do not account for an equally large share of the interval. For our purposes, of course, $d_i > 0 \ \forall i \in \{1, ..., n\}$ so $(u, v)_D$ indeed defines a scalar product.
We denote the D-norm by

$$||x||_D = \sqrt{(x, x)_D}.$$

This project is based on the paper "Weighted FOM and GMRES for solving nonsymmetric linear systems" by Azzedine Essai [1]. The goal is to describe the mathematics behind GMRES and WGMRES and their implementations. In section 2 we describe the Arnoldi iteration and GMRES and in section 3 we describe the weighted analogues. Links between WGMRES and GMRES and some convergence properties will be established in section 3 and 4 and in section 5 we present some numerical experiments aiming to compare the speed of convergence of the two methods. Conclusions are found in section 6.

2 Arnoldi Iteration and GMRES

The Cayley-Hamilton theorem states that an $n \times n$ matrix A satisfies its own characteristic equation:

$$\chi_A(A) = A^n + c_{n-1}A^{n-1} + ... + c_o I = 0.$$

Assuming additionaly that A is invertible, we see that the exact solution $A^{-1}b$ can be written as a linear combination of the form:

$$-\tfrac{1}{c_0}(A^{n-1}b + c_{n-1}A^{n-2}b + ...c_1 b) = A^{-1}b.$$

This shows that the exact solution $A^{-1}b$ can be expressed entirely in powers of A times b. The idea of GMRES is to repeatedly approximate the solution of a linear system $Ax = b$ by a vector in the sequence of *Krylov subspaces*

$$\mathcal{K}_m(A, b) =< b, Ab, ..., A^{m-1}b >$$

for $m = 1, 2...$ until the approximate solution is sufficiently close to the real solution. Alteratively, let $r_0 = b - Ax_0$, for some arbitrary vector x_0. Then $A^{-1}r_0 = A^{-1}b - Ix_0$ so that $A^{-1}b = x_0 + A^{-1}r_0$, hence looking for $x \in \mathcal{K}_m(A, b)$ is equivalent to look for $x \in x_0 + \mathcal{K}_m(A, r_0)$.
At step m, we solve a least squares problem to find the vector

$x_m \in x_0 + \mathcal{K}_m(A, r_0)$ that minimizes $||r_m||_2 = ||b - Ax_m||_2$. Solving this least squares problem directly in each iteration would be a numerically unstable procedure [2]. It is therefore essential to create a new set of vectors that span the same space but have better numerical properties. This is what the Arnoldi iteration takes care of. It creates an orthonormal basis $\{v_1, ..., v_m\}$ for the Krylov space and a *partial Hessenberg reduction* $AV_m = V_{m+1}H_{m+1}$. Just as when computing a QR factorization, Gram-Schmidt orthogonalization has the advantage that it can be stopped part-way, contrary to e.g. Householder reflection where the full QR-decomposition is computed. The situation is analogous when computing a Hessenberg reduction $A = VHV^T$. Exactly as the Gram-Schmidt algorithm, Arnoldi iteration can stopped part-way to give a partial Hessenberg reduction $AV_m = V_{m+1}H_{m+1}$, where H_{m+1} is the $(m+1) \times m$ upper left section of H and V_m is the first m columns of the matrix V:

$$\begin{bmatrix} \quad A \quad \end{bmatrix} \begin{bmatrix} v_1 | v_2 | ... | v_m \end{bmatrix} = \begin{bmatrix} v_1 | v_2 | ... | v_{m+1} \end{bmatrix} \begin{bmatrix} h_{11} & ... & & h_{1m} \\ h_{21} & h_{22} & & \vdots \\ & & \ddots & \\ & & h_{m,m-1} & h_{mm} \\ & & & h_{m+1,m} \end{bmatrix}$$

The mth column of this equation can be written

$$Av_m = h_{1,m}v_1 + ... + h_{m,m}v_m + h_{m+1,m}v_{m+1}.$$

This shows that the vector v_{m+1} satisfies a rucurrence relation involving the previously produced vectors. Arnoldi iteration implements this idea to create an orthonormal basis by a Gram-Schmidt procedure, such that the relation $AV_m = V_{m+1}H_{m+1}$ holds [2]. We will denote by H_m the matrix H_{m+1} with its last row removed. The starting vector v_1 in the Arnoldi iteration is arbitrary, but we will choose it to be $r_0 = b - Ax_0$, where x_0 is an initial guess of the solution of a linear system. $(.,.)_2$ denotes the euclidean scalar product.

Algorithm 1: Arnoldi Iteration

```
$v_1 = r_0/||r_0||_2$
for j=1:m
    $w = Av_j$
    for i=1:j
    $h_{i,j} = (w, v_i)_2$
    $w = w - h_{i,j}v_i$
end
$h_{j+1,j} = ||w||$, if $h_{j+1,j} = 0$, stop
$v_{j+1} = w/h_{j+1,j}$,
end
```

Now that we have an orthonormal basis $\{v_1, ..., v_m\}$ of $\mathcal{K}_m(A, r_0)$ and let V_m be the orthogonal matrix with $v_1, ..., v_m$ as its columns, we can derive the minimization problem that defines GMRES. The mth iterate x_m can be written as $x_0 + V_m y$ for some $y \in \mathbb{R}^m$ so for $x \in x_0 + \mathcal{K}_m(A, r_0)$, we have

$$||b - Ax||_2 = ||b - A(x_0 + V_m y)||_2 = ||b - Ax_0 - AV_m y||_2 = ||r_0 - AV_m y||_2,$$

thus

$$\min_{x \in x_0 + \mathcal{K}_m(A,R_0)} ||b - Ax||_2 = \min_{y \in \mathbb{R}^m} ||r_0 - AV_m y||_2 \tag{1}$$

Furthermore, for the vectors $v_1, ..., v_m$ and the matrix H_{m+1} produced by Algorithm 1, the relation

$$AV_m = V_{m+1}H_{m+1} \tag{2}$$

holds. Using relation (2), we get

$$||r_0 - AV_m y||_2 = ||r_0 - V_{m+1}H_{m+1}y||_2.$$

Since multiplying the vectors by V_{m+1}^T does not change the norm we get

$$||V_{m+1}^T(r_0 - V_{m+1}H_{m+1}y)||_2 = ||V_{m+1}^T r_0 - H_{m+1}y||_2.$$

The product $V_{m+1}^T r_0$ in this equation becomes $v_1^T r_0 = \frac{||r_0||_2^2}{||r_0||_2} = ||r_0||_2$ and $v_k^T r_0 = ||r_0|| v_k^T v_1 = 0$ for all $k > 1$.
Thus our final minimization problem becomes

$$\text{Find } y \in \mathbb{R}^m \text{ s.t. } ||H_{m+1}y - \beta e_1|| \text{ is minimal} \tag{3}$$

where $\beta = ||r_0||_2$.

The GMRES algorithm is presented in Algorithm 2. This algorithm is inefficient for two reasons, and it is not the way GMRES actually should be implemented.
1. We do not have to compute the solution explicitly in each step to check the convergence criteria.
2. The QR decomposition can be done by a single Givens rotation since the

matrix H_m is Hessenberg.
However, for pedagogical reasons, we present this rather theoretical version of GMRES and give the more efficient version of WGMRES in the next section instead.

Algorithm 2: GMRES

```
- Choose an initial guess $x_0$, a tolerance $\epsilon$ and compute the corresponding residual $r_0 = b - Ax_0$.
- Compute $\beta = ||r_0||_2$, and set $v_1 = r_0/\beta$
- For k=1...
    Construct the D-orthonormal basis $V_k$ by Algorithm 1 with starting vector $v_1$. Solve the least squares problem $y_k = \text{argmin}_{y \in \mathbb{R}^k} ||\beta e_1 - H_{k+1} y||_2$ by QR factorization. Set $x_k = x_0 + V_k y_k$ and $r_k = b - Ax_k$.
- Repeat until $||b - Ax_k|| < \epsilon$.
```

3 Weighted Arnoldi Iteration and WGMRES

In the previous section we created orthonormal bases with respect to the Euclidean norm (2-norm).
The weighted Arnoldi process is the same as the Arnoldi process, the only difference being the scalar product.

Algorithm 3: Weighted Arnoldi Iteration

```
$\widetilde{v_1} = v/||v||_D$
    for j=1:k
    $w = A\widetilde{v_j}$
    for i=1:j
$\widetilde{h}_{i,j} = (w, \widetilde{v_i})_D$
    $w = w - \widetilde{h}_{i,j}\widetilde{v_i}$
    end
$\widetilde{h}_{j+1,j} = ||w||_D$, if $\widetilde{h}_{j+1,j} = 0$ stop
$\widetilde{v}_{j+1} = w/\widetilde{h}_{j+1,j}$
end
```

The basis $V_m = (v_1, ..., v_m)$ constructed by the Arnoldi iteration is orthonormal, thus

$$V_m^T V_m = I_m.$$

Multiplying (2) from the left by V_m^T gives

$$V_m^T A V_m = V_m^T V_{m+1} H_{m+1}.$$

Looking at the right hand side, $V_m^T V_{m+1}$ becomes the $m \times m$ identity plus an additional column of zeros on the right. This column of zeros, when multiplied by H_{m+1}, deletes the last row of H_{m+1}, which is exactly H_m, hence

$$H_m = V_m^T A V_m$$

Similar relations hold for the D-orthonormal basis created in the weighted Arnoldi iteration (everything related to the weighted Arnoldi algorithm will be denoted with a tilde on top):

$$\widetilde{V}_m^T D \widetilde{V}_m = I_m, \tag{4}$$

$$\widetilde{H}_m = \widetilde{V}_m^T D A \widetilde{V}_m, \tag{5}$$

$$A\widetilde{V}_m = \widetilde{V}_{m+1} \widetilde{H}_{m+1}, \tag{6}$$

where also here $\widetilde{H}_m$ is $\widetilde{H}_{m+1}$ with it last row removed.

We now prove some relations connecting the matrices generated by Algorithms 1 and 2.

Proposition 1 Assuming that algorithms 1 and 2 do not break down before the mth step, there exists an upper triangular matrix U_m such that

$$\widetilde{V}_m = V_m U_m, \tag{7}$$

$$U_m = V_m^T \widetilde{V}_m, \tag{8}$$

$$U_m^{-1} = \widetilde{V}_m^T D V_m, \tag{9}$$

and $\widetilde{H}_{m+1}$ can be expressed in terms of H_{m+1} as

$$\widetilde{H}_{m+1} = U_{m+1}^{-1} H_{m+1} U_m \tag{10}$$

Proof

V_m and $\widetilde{V}_m$ are bases for the same space and hence U_m is a change of basis matrix, which always exist.

Multiplying (7) from the left by V_m^T gives (8).

Multiplying (7) from the left by $\widetilde{V}_m^T D$ gives $\widetilde{V}_m^T D \widetilde{V}_m = I_m = \widetilde{V}_m^T D V_m U_m$ since $\widetilde{V}_m$ is D-orthogonal. Hence $U_m^{-1} = \widetilde{V}_m^T D V_m$.

Using (6) and changing $\widetilde{V}_m$ on the left to $V_m U_m$ according to (7), we obtain

$$AV_m U_m = \widetilde{V}_{m+1} \widetilde{H}_{m+1}.$$

Also by (7) we have that $\widetilde{V}_{m+1} = V_{m+1} U_{m+1}$, hence

$$AV_m U_m = V_{m+1} U_{m+1} \widetilde{H}_{m+1}$$

Now changing AV_m to $V_{m+1} H_{m+1}$ according to (2) we obtain

$$V_{m+1} H_{m+1} U_m = V_{m+1} U_{m+1} \widetilde{H}_{m+1}.$$

Multiplying from the left by V_{m+1}^T and then by U_{m+1}^{-1} gives (10).

Proposition 2. Under the same assumptions as in Proposition 1, $\widetilde{H}_m$ can be expressed in terms of H_m by the relation

$$\widetilde{H}_m = U_m^{-1} H_m U_m + h_{m+1,m} u_{m,m} g_{m+1} e_m^T, \tag{11}$$

where $g_{m+1} \in \mathbb{R}^m$ is obtained from column $m+1$ of the matrix U_{m+1}^{-1} by deleting its last component.

Proof

Let $g_{i,j}$ ($1 \leq i, j \leq m$) be the entries of the matrix U_m^{-1}. We can write the matrix U_{m+1}^{-1} as

$$U_{m+1}^{-1} = \begin{bmatrix} & U_m^{-1} & & g_{m+1} \\ 0 & \dots & 0 & g_{m+1,m+1} \end{bmatrix} = \begin{bmatrix} & & \widehat{U}_{m+1}^{-1} & \\ 0 & \dots & 0 & g_{m+1,m+1} \end{bmatrix},$$

so that $\widehat{U}_{m+1}^{-1}$ is the matrix U_{m+1}^{-1} with its last row removed. Using (10), but omitting the last row of the matrix, we get

$$\widetilde{H}_m = \widehat{U}_{m+1}^{-1} H_{m+1} U_m$$

from which we can see the result by looking at the above equation in matrix form:

$$\widetilde{H}_m = \begin{bmatrix} U_m^{-1} & g_{m+1} \end{bmatrix} \begin{bmatrix} H_m \\ h_{m+1,m} e_m^T \end{bmatrix} U_m = U_m^{-1} H_m U_m + h_{m+1,m} g_{m+1} e_m^T U_m.$$

Since U_m is upper triangular, $e_m^T U_m = u_{m,m} e_m^T$ and we have the desired result.

Now that we have the weighted Arnoldi algorithm in place and some relations between it and the Arnoldi algorithm we derive the correct minimization problem and show how WGMRES is implemented.
The mth iterate x_m can be written as

$$x_m = x_0 + \widetilde{V}_m y,$$

for some $y \in \mathbb{R}^m$. Hence, the mth residual can be written

$$r_m = b - A(x_0 + \widetilde{V}_m y_m) = r_0 - A\widetilde{V}_m y_m = \widetilde{V}_{m+1}(\widetilde{\beta} e_1 - \widetilde{H}_{m+1} y_m),$$

where $\widetilde{\beta} = ||r_0||_D$ and the last equality follows from (6). This gives

$$||r_m||_D^2 = ||\widetilde{V}_{m+1}(\widetilde{\beta} e_1 - \widetilde{H}_{m+1} y_m)||_D^2 = ||\widetilde{\beta} e_1 - \widetilde{H}_{m+1} y_m||_2^2,$$

since the matrix $\widetilde{V}_{m+1}$ is D-orthonormal. Thus the minimization problem in WGMRES is

$$\text{Find } y \in \mathbb{R}^m \text{ s.t. } ||\widetilde{\beta} e_1 - \widetilde{H}_{m+1} y||_2 \text{ is minimal.} \tag{12}$$

We now present the weighted GMRES algorithm. At step m, this algorithm requires storing the matrix $\widetilde{V}_m$. If m grows large, this storage might be a constraint. A remedy to such a problem is to restart the process after some m iterations, where we simply let the approximate solution after m steps be our new initial guess, i.e. set $x_m = x_0$ and restart the process. Following the idea in [1], we will choose the elements of the weight matrix D to be $d_i = \frac{\sqrt{n}}{||r_0||_2}|r_0(i)|$. The idea of this is to speed up the convergence of the solution by favoring those elements in the residual at each step that are large. Furthermore, this weight matrix is updated whenever the algorithm is restarted to speed up the convergence even more. Restarted WGMRES is denoted WGMRES(m). Before presenting the algorithm in the way it is implemented, we describe how the convergence criteria can be checked without computing the solution in each step, as promised when Algorithm 2 was presented.

Looking at the minimization problem $||\widetilde{\beta}e_1 - \widetilde{H}_{m+1}y||_2$ and given a full QR-decomposition of $\widetilde{H}_{m+1} = \bar{Q}\bar{R}$, where

$$\bar{R} = \begin{bmatrix} & R & \\ 0 & ... & 0 \end{bmatrix}$$

we get

$$||\bar{Q}\widetilde{\beta}e_1 - \bar{R}y||_2 = ||\bar{g}_m - \bar{R}y||_2.$$

Since we know that

$$g_m = Ry$$

has a unique solution, i.e $g_m - Ry = 0$ we must have that

$$||\bar{g}_m - \bar{R}y||_2 = || \begin{bmatrix} 0 \\ \gamma_{m+1} \end{bmatrix} ||_2 = |\gamma_{m+1}|$$

Therefore the solution $x_m = x_0 + \widetilde{V}_m y$ is computed only if γ_{m+1}, the last component of $\bar{Q}\beta e_1$, is less than the given tolerance in absolute terms.
The QR-decomposition is done by a single Givens rotation (step 4 in Algorithm 4).

Algorithm 4 WGMRES(m)

```
1. Choose $m$, an initial guess $x_0$, a tolerance $\epsilon$ and compute the corresponding residual $r_0 = b - Ax_0$.
2. Compute $\widetilde{\beta} = ||r_0||_D$, $\widetilde{v}_1 = r_0/\widetilde{\beta}$ and choose weight vector $d$ ($D = \text{diag}(d)$)
3. For k=1...
    Construct the D-orthonormal basis $\widetilde{V}_k$ by Algorithm 3 with starting vector $\widetilde{v}_1$.
4. For i=1:k-1
    $\begin{bmatrix} h_{i,k} \\ h_{i+1,k} \end{bmatrix} = \begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix} \begin{bmatrix} h_{i,k} \\ h_{i+1,k} \end{bmatrix}$
$b = \sqrt{h_{k,k}^2 + h_{k+1,k}^2},\ s_k = h_{k+1,k}/b$
$c_k = h_{k,k}/\beta,\ h_{k,k} = b$
$\gamma_{k+1} = -s_k\gamma_k,\ \gamma_k = c_k\gamma_k$
    if $|\gamma_{k+1}| \geq \epsilon, v_{k+1} = w_k/h_{k+1,k}$ else
    For i=k,...,1
```

```
        $a_i = \frac{1}{h_{i,i}}(\gamma_i - \sum_{j=i+1}^{k} h_{i,j}a_j)$
    $x = x_0 + \sum_{i=1}^{k} a_i v_i$
    END
5. If k=m: form the solution and set $x_0 = x_m,\ r_0 = r_m$ and restart (go to step 2).
```

4 Convergence properties and links between GMRES and WGMRES

First off, we can conclude that the weighted Arnoldi algorithm requires more operations than the Arnoldi algorithm, coming from the use of a non-standard scalar product.

Computational cost

Let N_{nz} denote the number of nonzero elements of A. At each step, one matrix-vector product is carried out in both Arnoldi iteration and weighted Arnoldi iteration, requiring $\approx 2mN_{nz}$ operations after m steps. Secondly, the Euclidean inner product costs $\approx 2n$ operations for Arnoldi and $\approx 3n$ operations for the D-inner product in the weighted Arnoldi. The inner loop requires $\approx 2jn$ operations for Arnoldi and $\approx 3jn$ operations for weighted Arnoldi. Forming the vector v_{j+1} costs $2jn$ operations for both algorithms. In total, we have $2mN_{nz} + 2m^2n$ for Arnoldi and $2mN_{nz} + (5/2)m^2n$ for weighted Arnoldi [1]. WGMRES can only be faster if less iterations are needed.

Convergence properties

If $x \in x_0 + \mathcal{K}_m$, it can be written as

$$x = x_0 + \sum_{j=0}^{m-1} c_j A^j r_0,$$

so the residual can be written

$$b - Ax = b - Ax_0 - \sum_{j=0}^{m-1} c_j A^{j+1} r_0 = r_0 - \sum_{j=1}^{m} c_{j-1} A^j r_0.$$

The minimization problem can therefore be formulated as a polynomial approximation problem where we seek the polynomial $p_m \in \Pi_m$ s.t.

$$||p_m(A) r_0||_2$$

is minimized, where $\Pi_m = \{\text{polynomials } p \text{ of degree} \leq m \text{ with } p(0) = 1\}$. This results in the following theorem.

Theorem 1. Let A be nonsingular and x_m be the mth iterate. Then for all $\widehat{p} \in \Pi_m$,

$$||r_m||_2 = \min_{p \in \Pi_m} ||p(A) r_0||_2 \leq ||\widehat{p}(A) r_0||_2. \tag{13}$$

Note that since $||\widehat{p}(A) r_0||_2 \leq ||\widehat{p}(A)||_2 ||r_0||_2$ we also have that

$$\frac{||r_m||_2}{||r_0||_2} \leq ||\widehat{p}(A)||_2. \tag{14}$$

Assume that A is diagonalizable

$$A = V \Lambda V^{-1}$$

where V is the nonsingular matrix consisting of the eigenvectors of A and Λ is a diagonal matrix with the eigenvalues on the diagonal. We can then create the estimate

$$||\widehat{p}(A)||_2 \leq ||V||||\widehat{p}(\Lambda)||||V^{-1}|| \leq \kappa(V)\max_{z\in\sigma(A)}|\widehat{p}(z)|,$$

where $\sigma(A)$ is the sprectrum of A and $\kappa(V)$ is the condition number of V [2]. Using this and (14) we get the following result.

Theorem 2. Let $A = V\Lambda V^{-1}$ be nonsingular and diagonalizable. Then for all $\widehat{p} \in \Pi_m$ the mth iterate satisfies

$$\frac{||r_m||_2}{||r_0||_2} \leq \kappa(V)\max_{z\in\sigma(A)}|\widehat{p}(z)|. \tag{15}$$

The method that finds a polynomial whose size is small on the spectrum of A might exhibit faster convergence.

Links between GMRES and WGMRES

Let $x_m^W = x_0 + \widetilde{V}_m y_m^W$ denote the approximate solution generated by WGMRES at step m. By (7), $\widetilde{V}_m = V_m U_m$. Let $\widehat{y}_m^W = U_m y_m^W$. We can then write

$$x_m^W - x_0 = \widetilde{V}_m y_m^W = V_m \widehat{y}^W.$$

y_m^W is the solution to the minimization problem (12), i.e.

$$y_m^W = \text{argmin}_{y\in\mathbb{R}^m}||\widetilde{\beta}e_1 - \widetilde{H}_{m+1}y||_2.$$

Since $y_m^W = U_m^{-1}\widehat{y}_m^W$, we see that

$$\widehat{y_m}^W = \text{argmin}_{\widehat{y}\in\mathbb{R}^m}||\widetilde{\beta}e_1 - \widetilde{H}_{m+1}U_m^{-1}\widehat{y}||_2.$$

Using (10) of Proposition 1, we obtain ($\beta = ||r_0||_2$)

$$\widehat{y}_m^W = \text{argmin}_{\widehat{y}\in\mathbb{R}^m}||U_{m+1}^{-1}(\beta e_1 - H_{m+1}\widehat{y})||_2.$$

Now notice that $\widehat{y}_m^W$ is the solution to the same minimization problem as for GMRES but with the norm induced by $U_{m+1}^{-T}U_{m+1}^{-1}$ instead of I_{m+1}, the euclidean one. Therefore we have

$$\widehat{y}_m^W = \text{argmin}_{\widehat{y}\in\mathbb{R}^m}||\beta e_1 - H_{m+1}\widehat{y}||_{U_{m+1}^{-T}U_{m+1}^{-1}} \tag{16}$$

Remark The matrix $U_{m+1}^{-T}U_{m+1}^{-1}$ is symmetric since $(U_{m+1}^{-T}U_{m+1}^{-1})^T = (U_{m+1}^{-1})^T(U_{m+1}^{-T})^T = U_{m+1}^{-T}U_{m+1}^{-1}$.

(16) allows us to relate the residuals created in each step of GMRES and WGMRES [3].

Theorem 3 The mth residuals r_m and $\widetilde{r}_m$ created from GMRES and WGMRES respectively, satisfy

$$\sqrt{\lambda_{min}(U_{m+1}^{-T}U_{m+1}^{-1})}||r_m||_2 \leq ||\widetilde{r}_m||_D \leq \sqrt{\lambda_{max}(U_{m+1}^{-T}U_{m+1}^{-1})}||r_m||_2, \tag{17}$$

where λ_{min} and λ_{max} are the smallest and largest eigenvalues of the matrix in question, respectively.

Proof

From (16), we have that $||\widetilde{r}_m||_D = \min_{\widehat{y}\in\mathbb{R}^m}||\beta e_1 - H_{m+1}\widehat{y}||_{U_{m+1}^{-T}U_{m+1}^{-1}}$.

We also know that $U_{m+1}^{-T}U_{m+1}^{-1}$ is symmetric, and hence diagonalizable. Let $Q\Lambda Q^T$ be a diagonalization of $U_{m+1}^{-T}U_{m+1}^{-1}$, where Λ is a diagonal matrix with the egeinvalues $\{\lambda_i\}_{i=1}^{m+1}$ of $U_{m+1}^{-T}U_{m+1}^{-1}$ on its diagonal. Let

$$z_m = \text{argmin}_{z\in\mathbb{R}^m}||\beta e_1 - H_{m+1}z||_2.$$

Then,

$$||\widetilde{r}_m||_D^2 \le ||\beta e_1 - H_{m+1} z_m||^2_{U_{m+1}^{-T} U_{m+1}^{-1}} =$$
$$(\beta e_1 - H_{m+1} z_m)^T U_{m+1}^{-T} U_{m+1}^{-1} (\beta e_1 - H_{m+1} z_m) =$$
$$(\beta e_1 - H_{m+1} z_m)^T Q \Lambda Q^T (\beta e_1 - H_{m+1} z_m).$$

For notational simplicity, let $w = (\beta e_1 - H_{m+1} z_m)$. The last expression becomes

$$w^T Q \Lambda Q^T w = (Q^T w)^T \Lambda (Q^T w) = \sum_{i=1}^{m+1} \lambda_i (q_i^T w)^2$$
$$\le \lambda_{\max}(U_{m+1}^{-T} U_{m+1}^{-1}) \sum_{i=1}^{m+1} \lambda_i (q_i^T w)^2.$$

The summand is the squared norm of the vector $w = (\beta e_1 - H_{m+1} z_m)$. The norm of this vector is $||r_m||_2$ since z_m minimizes $||\beta e_1 - H_{m+1} z||_2$. By taking square roots, it follows that

$$||\widetilde{r}_m||_D \le \sqrt{\lambda_{\max}(U_{m+1}^{-T} U_{m+1}^{-1})}\, ||r_m||_2.$$

We use a similar procedure for the other inequality. Now let

$$y_m = \text{argmin}_{y \in \mathbb{R}^m} ||\beta e_1 - H_{m+1} y||_{U_{m+1}^{-T} U_{m+1}^{-1}}$$

Then

$$||\widetilde{r}_m||_D^2 = ||\beta e_1 - H_{m+1} y_m||^2_{U_{m+1}^{-T} U_{m+1}^{-1}} =$$
$$(\beta e_1 - H_{m+1} y_m)^T U_{m+1}^{-T} U_{m+1}^{-1} (\beta e_1 - H_{m+1} y_m).$$

Again using the diagonalization of $U_{m+1}^{-T} U_{m+1}^{-1}$, this expression becomes

$$\sum_{i=1}^{m+1} \lambda_i (q_i^T (\beta e_1 - H_{m+1} y_m))^2$$

We can estimate this by factorizing out $\lambda_{\min}(U_{m+1}^{-T}U_{m+1}^{-1})$ out of the expression. What remains is then the squared norm of the vector $(\beta e_1 - H_{m+1}y_m)$. We know that z_m minimizes this quantity, hence replacing y_m by z_m can only make this expression smaller. Still, $||\beta e_1 - H_{m+1}z_m||_2 = ||r_m||_2$. We get that

$$\sum_{i=1}^{m+1} \lambda_i (q_i^T(\beta e_1 - H_{m+1}y_m))^2 \geq \lambda_{\min}(U_{m+1}^{-T}U_{m+1}^{-1})||r_m||_2^2$$

and the proof is complete.

Corollary The mth residuals r_m and $\widetilde{r}_m$ created from GMRES and WGMRES respectively, satisfy

$$\sqrt{\lambda_{min}(D)}||r_m||_2 \leq ||\widetilde{r}_m||_D \leq \sqrt{\lambda_{max}(D)}||r_m||_2. \tag{18}$$

Proof

The result follows from the previous theorem and the calculation

$$U_{m+1}^{-T}U_{m+1}^{-1} \overset{(9)}{=} (\widetilde{V}_{m+1}^T D V_{m+1})^T U_{m+1}^{-1} = V_{m+1}^T D^T \widetilde{V}_{m+1} U_{m+1}^{-1} \overset{(7)}{=}$$
$$V_{m+1}^T D V_{m+1} U_{m+1} U_{m+1}^{-1} = V_{m+1}^T D V_{m+1},$$

which shows that $U_{m+1}^{-T}U_{m+1}^{-1}$ and D are similar and hence have the same singular values (and also the same eigenvalues since $U_{m+1}^{-T}U_{m+1}^{-1}$ is symmetric).

5 Numerical experiments

In this section we present some numerical experiments. Examples 1-5 are the experiments originally presented in Essai [1]. The WGMRES algorithm presented in Essai updates the weight matrix D at each restart. As our motivation of using WGMRES is different, we will also perform some experiments without re choosing the weight matrix and even investigate what happens if WGMRES and GMRES without restart is applied to the problems. As will be seen, the speedup is lost when the two methods is run without restart for these examples. The matrices tested in examples 1-5 are from the Matrix Market Web server [4] and the right hand side b is a random vector with entries uniformly distributed in [0,1]. The initial guess is $x_0 = (0, ..., 0)$. The iteration is stopped after a predefined maximum number of iterations or when

$$||r||_2/||b||_2 < \epsilon,$$

where ϵ depends on the problem. As the right-hand side b in the experiments is a random vector, the exact results from [1] are never recreated. In example 2 we investigate how the convergence is affected if we do not re choose the matrix D at each restart. For all other examples we also run the algorithms without restart. The code was written in MATLAB as described by Algorithm 4. All tables show the average over ten trials if not otherwise stated, while the figures provide an example of one of those trials. In Example 6 we solve the linear system arising from the Poisson equation with five different weight matrices.

Example 1. The matrix add20 is a 2395x2395 matrix with 17319 nonzero entries. For this matrix we compare GMRES(10) and WGMRES(10) with $\epsilon = 10^{-12}$. The result is presented in Figure 1 and Table 1.

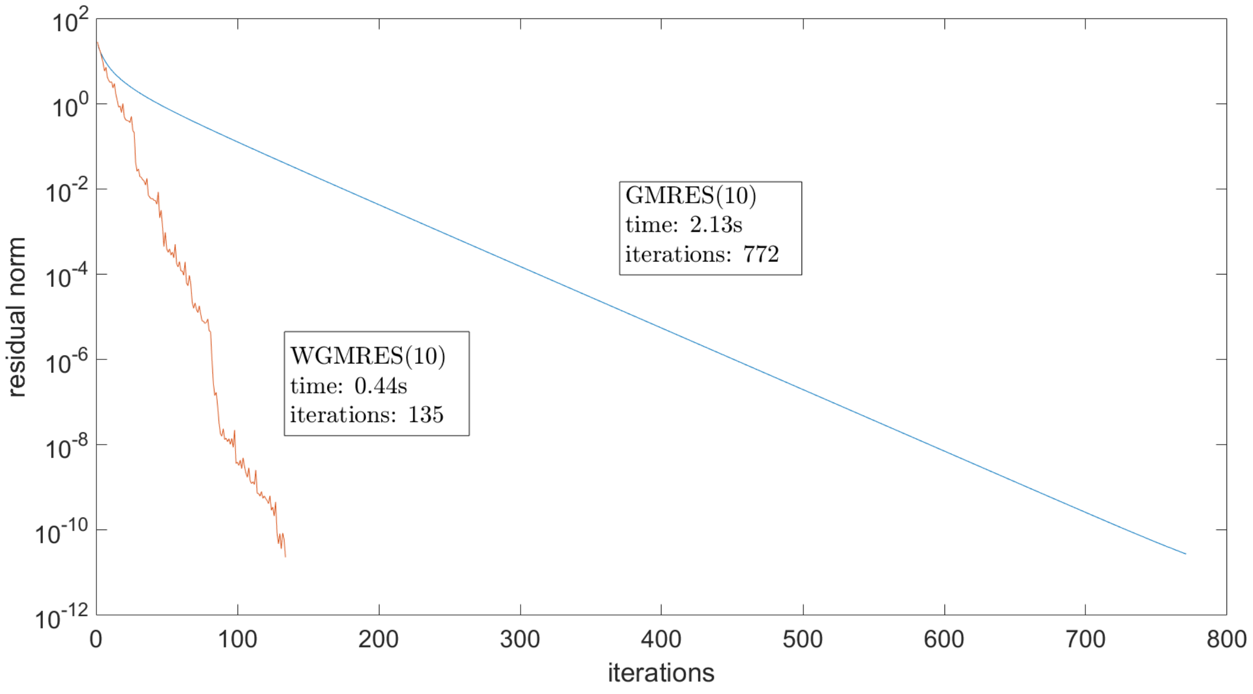

Figure 1: GMRES(10) vs WGMRES(10) on the matrix add20.

Table 1: add20

Method	GMRES(10)	WGMRES(10)	GMRES	WGMRES
Iterations	773	112	309	308
Time	2.61s	0.49s	0.64s	0.81s

Example 2. orsirr_1 is a 1030x1030 matrix with 6858 nonzero entries. We let $\epsilon = 10^{-11}$ and compare GMRES(m) and WGMRES(m) for eight different values of m, ranging from 10 to 80. We also include a third method, WGMRES(m) but without re choosing the weight matrix D after each cycle. This is to demonstrate that the speedup caused by WGMRES(m) compared to GMRES(m) is lost if the weight matrix is not re chosen at each restart for this example. The results are shown in Table 2. The optimal choice of m with respect to time agrees with Essai's findings. * indicates that the method failed to converge within 2000 iterations for at least one of the ten trials.

Table 2: orrsirr_1

	GMRES		WGMRES		WGMRES (constant D)	
m	iterations	time	iterations	time	iterations	time
10	*	*	*	*	*	*
20	806	3.33	208	**1.04**	1000	6.06
30	246	1.97	139	1.38	230	2.72
40	127	1.63	70	1.27	111	2.15
50	94	1.68	53	1.19	107	2.92
60	64	**1.506**	40	1.32	52	2.09
70	48	1.58	31	1.41	37	**1.80**
80	37	1.510	26	1.41	35	2.25

Example 3. fs_541_2 is a 541x541 matrix with 4285 nonzero entries. Both methods performed poorly on this matrix, failing to converge for $m = 40$. Essai found that the convergence curve oscillates for GMRES(40) but converges in 138 iterations for WGMRES(40), even though it also oscillates until iteration 107. This matrix was tested six times for both GMRES(m) and WGMRES(m) for $m = 40, 60, 80, 100, 120$. Neither GMRES(m) or WGMRES(m) converged for $m < 100$. ϵ was set to 10^{-10} and

maximum number of iterations to 1000. For $m = 100$ the result varied as much as from 11 iterations to not converging at all. The results over six runs for GMRES(m) are shown in table 3 and for WGMRES(m) in table 4. k_l indicates the number of iterations in trial l and t_l time elapsed in trial l. It failed to converge for both GMRES and WGMRES without restart with tolerance $\epsilon = 10^{-10}$. The results with $\epsilon = 10^{-9}$ are shown in figure 2.

Table 3: GMRES(m) on fs_541_2

m	k_1	t_1	k_2	t_2	k_3	t_3	k_4	t_4	k_5	t_5	k_6	t_6
100	23	1.45	29	1.79	*	*	*	*	*	*	11	0.69
120	9	0.75	11	0.60	11	0.87	10	0.74	10	0.66	11	0.80

Table 4: WGMRES(m) on fs_541_2

m	k_1	t_1	k_2	t_2	k_3	t_3	k_4	t_4	k_5	t_5	k_6	t_6
100	81	6.92	97	7.89	413	29.76	169	12.37	*	*	11	1.09
120	8	0.93	9	0.91	9	1.02	9	0.83	11	1.02	9	0.77

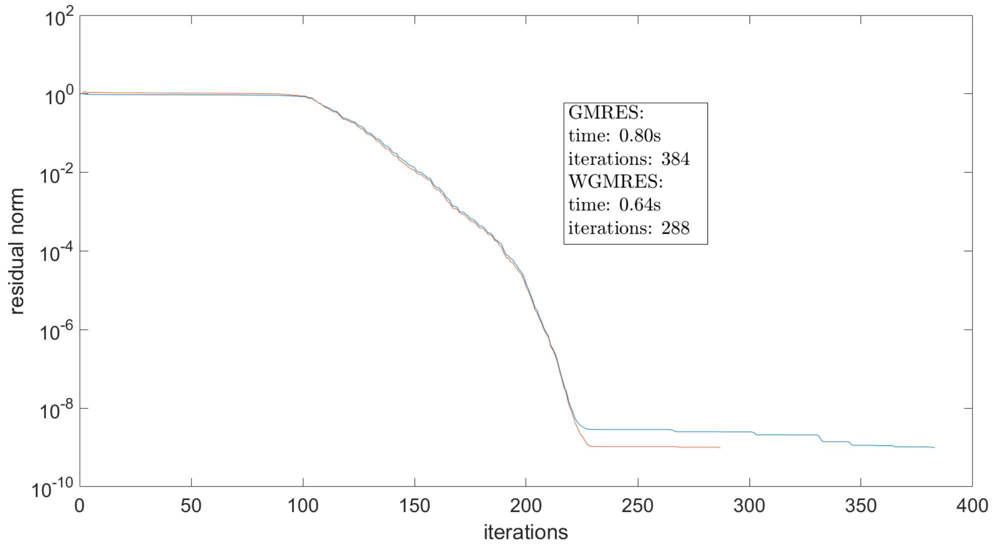

Figure 2: GMRES vs WGMRES without restart on the matrix fs_541_2 with tolerance $\epsilon = 10^{-9}$.

Example 4. bfw782a is a 782x782 matrix with 7514 nonzero entries. The result with m=20 and $\epsilon = 10^{-12}$ is presented in Figure 3 and table 5

Table 5: bfw782a

Method	GMRES(20)	WGMRES(20)	GMRES	WGMRES
Iterations	418	159	615	607
Time	1.47s	0.92s	5.00s	5.11s

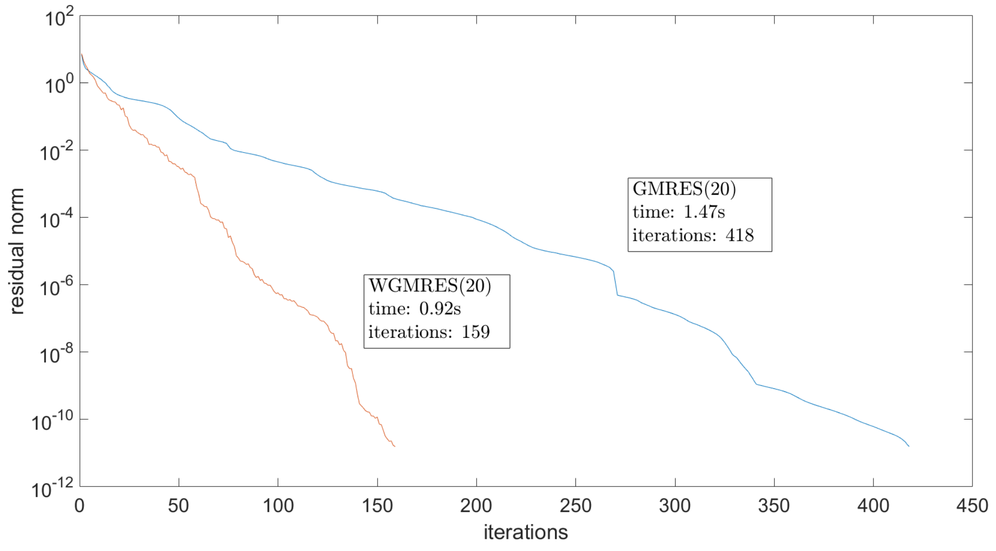

Figure 3: GMRES(20) vs WGMRES(20) on the matrix bfw782a.

Example 5. memplus is a 17758x17758 matrix with 126150 nonzero entries. The result for m=30 and $\epsilon = 10^{-12}$ are shown in figure 4 and table 6.

Table 6: memplus

Method	GMRES(30)	WGMRES(30)	GMRES	WGMRES
Iterations	434	126	1048	1046
Time	7.49s	19.97s	190.48s	245.79s

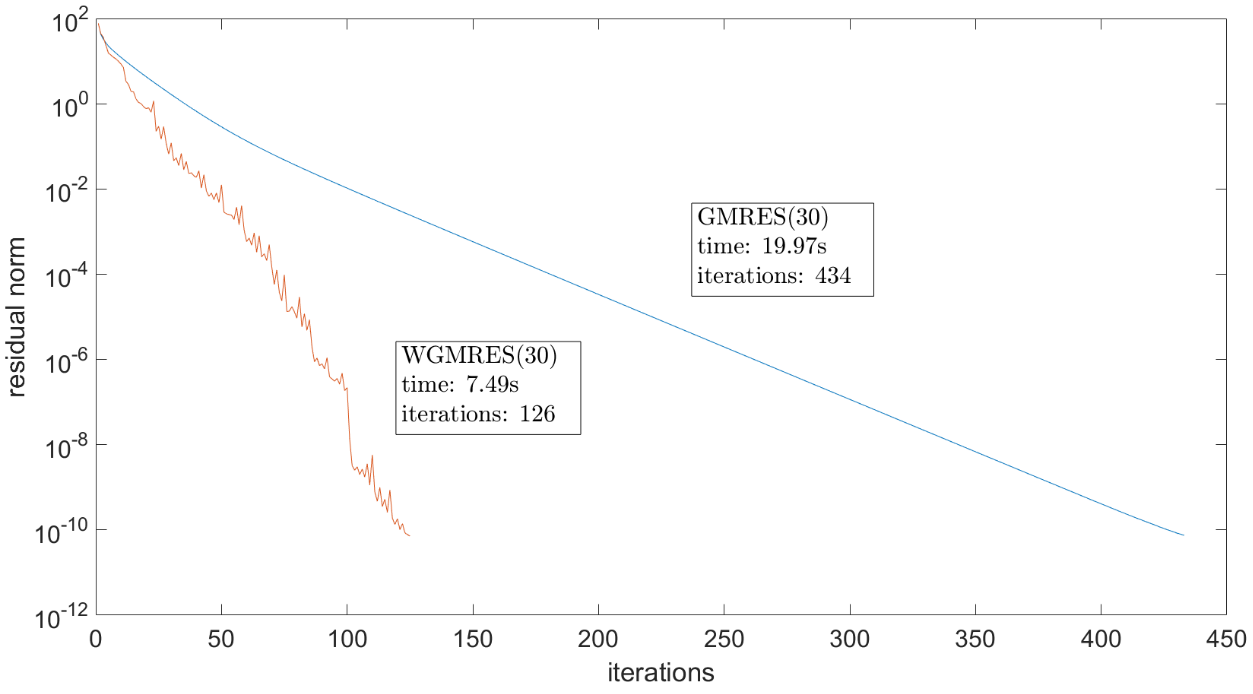

Figure 4: GMRES(30) vs WGMRES(30) on the matrix memplus.

The Poisson equation

Example 6. In examples 1-5 the weight matrix D was chosen in a way to speed up convergence. As mentioned in the introduction, we can imagine situations where the weight matrix is given from the problem, rather than chosen in order to speed up convergence. This example aims to investigate what happens when WGMRES is run with a weight matrix D that is somewhat arbitrarily chosen. Consider the linear system that arising from the

Poisson equation on an interval [a,b] with Dirichlet boundary conditions:

$$\begin{cases} -u'' = f \\ u(a) = \alpha,\ u(b) = \beta \end{cases}$$

Applying a finite difference method

$$-\tfrac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} = f(x_j),\ j = 2, 3, ..., N-1$$
$$u_0 = \alpha,\ u_{N+1} = \beta$$

we get the system

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} \Delta x^2 f(x_1) + \alpha \\ \Delta x^2 f(x_2) \\ \vdots \\ \Delta x^2 f(x_N) + \beta \end{bmatrix}$$

Now let

$$d = (\Delta x_1, \Delta x_1, ..., m, \Delta x_2, ..., \Delta x_2),$$

representing a non-equidistant discretization, where $m = \frac{\Delta x_1 + \Delta x_2}{2}$. The linear system arizing from this discretization is then

$$\begin{bmatrix} \frac{2}{\Delta x_1^2} & \frac{-1}{\Delta x_1^2} & & & & & & \\ \frac{-1}{\Delta x_1^2} & \frac{2}{\Delta x_1^2} & \frac{-1}{\Delta x_1^2} & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \frac{-1}{m^2} & \frac{2}{m^2} & \frac{-1}{m^2} & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & & \frac{-1}{\Delta x_2^2} & \frac{2}{\Delta x_2^2} & \frac{-1}{\Delta x_2^2} \\ & & & & & & \frac{-1}{\Delta x_2^2} & \frac{2}{\Delta x_2^2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix} = \begin{bmatrix} f(x_1) + \frac{\alpha}{\Delta x_1^2} \\ f(x_2) \\ \vdots \\ f(x_N) + \frac{\beta}{\Delta x_2^2} \end{bmatrix}$$

Let [a,b]=[0,1], $\alpha = \beta = 0$ and N=500. We solve the first system for two different right-hand sides, $f(x) = \sin(\pi x)$ and $f(x) = x(e^{1-x} - 1)$. Five tests are run on this problem for each right-hand side f.

We use $D = I$ which would correspond to an equidistant discretization of the interval. We then use the same weight matrix as in examples 1-5, namely $D_{r_0} = \frac{\sqrt{n}}{||r_0||_2} \text{diag}(|r_0|)$. Lastly we construct three weight matrices, representing non-equidistant discretizations of the problem, as

$D = \frac{\sqrt{n}}{||r_0||_2} \text{diag}(\Delta x_1, ..., \Delta x_1, m, \Delta x_2, ..., \Delta x_2)$, for three different values of Δx_1 and Δx_2. These three weight matrices will be denoted $D_{\min=h}$ where h represent the smallest entry on the diagonal.

The results are found in Table 7 and 8. The tolerance was set to $\epsilon = 10^{-10}$.

Table 7: The Poisson equation with $f(x) = \sin(\pi x)$

	$D = I$	D_{r_0}	$D_{\min=0.8}$	$D_{\min=0.5}$	$D_{\min=0.33}$
Iterations	252	252	252	252	252
Time	0.42s	0.42s	0.44s	0.46s	0.47s

Table 8: The Poisson equation with $f(x) = x(e^{1-x} - 1)$

	$D = I$	D_{r_0}	$D_{min=0.8}$	$D_{min=0.5}$	$D_{min=0.33}$
Iterations	500	500	500	500	500
Time	1.46s	1.56s	1.69s	1.66s	1.56s

6 Conclusions

WGMRES(m) performs better than GMRES(m) on all matrices tested in examples 1-5, verifying the results in Essai [1], although both GMRES(m) and WGMRES(m) performed poorly on the matrix fs_541_2 for $m < 100$. It was also shown that for the matrix orsirr_1, the speedup was lost when the weight matrix D was not re chosen at each restart. For the other examples, the two methods needed almost exactly as many iterations when run without restart. WGMRES needed more time, being more computationally costly.
In example 6, the difference between the five different choices of the weight matrix was negligible.

References

[1] Azeddine Essai. Weighted fom and gmres for solving nonsymmetric linear systems. *Numerical Algorithms*, 18(3-4):277–292, 1998.

[2] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[3] Jennifer Pestana and Andrew J Wathen. On choice of preconditioner for minimum residual methods for nonsymmetric matrices. 2010.

[4] Matrix Market Web Server. *http://math.nist.gov/MatrixMarket/*, accessed April 1, 2016.

Bachelor's Theses in Mathematical Sciences 2016:K16
ISSN 1654-6229

LUNFNA-4012-2016

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/