DQGMRES: a Direct Quasi – Minimal Residual Algorithm Based on Incomplete Orthogonalization *†

Yousef Saad and Kesheng Wu
University of Minnesota, Computer Science Department

May 1, 1995

Abstract

We describe a Krylov subspace technique, based on incomplete orthogonalization of the Krylov vectors, which can be considered as a truncated version of GMRES. Unlike GMRES(m), the restarted version of GMRES, the new method does not require restarting. Our numerical experiments show that DQGMRES method often performs better than GMRES(m). In addition, the algorithm is flexible to variable preconditioning, i.e., it can accommodate variations in the preconditioner at every step. In particular, this feature allows us to use any iterative solver as a right-preconditioner for DQGMRES. This inner-outer iterative combination often results in a robust approach for indefinite linear problems.

*This work was supported in part by DARPA under grant number NIST 60NANB2D1272 and in part by NSF under grant number NSF/CCR-9214116.

†This is a revision of a previous technical report of UMSI-93/131, Minnesota Supercomputing Institute, University of Minnesota, 1993

1 Introduction

There has been a flurry of activity in the general area of iterative methods for solving large sparse linear systems of equations in recent years, spurred in part by the increased demand for efficient solvers for three-dimensional problems. Among the methods of choice are the Krylov subspace techniques which find approximate solutions to a linear system $Ax = b$, that are of the form $x_m = x_0 + q_{m-1}(A)r_0$, where x_0 is an initial guess, $r_0 = b - Ax_0$, and q_{m-1} is a polynomial of degree $\leq m-1$. The GMRES algorithm [8], is a method which minimizes the residual norm over such approximations and is, at least in its standard non-restarted form, optimal in some sense. There are also a number of Krylov subspace methods that do not obey any optimality property but which perform quite well in practice. These are methods such as the bi-conjugate gradient method and techniques derived from it such as BiCGSTAB and TFQMR. In this paper we will present an algorithm in this category which combines quasi-minimization concepts and incomplete orthogonalization. To be specific, the Arnoldi vectors are only orthogonalized against a small number of previous vectors, a technique which we refer to as incomplete orthogonalization. In addition, the algorithm attempts to extract an approximate smallest-residual norm solution, via a 'quasi-minimization' process which ignores the non-orthogonality of the basis of the Krylov subspace resulting from the incomplete orthogonalization.

Since the algorithm to be presented is closely related to some of techniques already described in the literature, we will start by recalling in Section 2 some of the main ideas in this context. Sections 3 and 4 describe the idea of incomplete orthogonalization and present the DQGMRES algorithm. Section 5 describes a few properties of DQGMRES. Section 6 reports on some numerical experiments and Section 7 is a tentative conclusion.

2 Methods based on full Arnoldi orthogonalization

Given an initial guess x_0 to the linear system

$$Ax = b, \tag{1}$$

a general *projection method* seeks an approximate solution x_m from an affine subspace $x_0 + K_m$ of dimension m by imposing the Petrov-Galerkin condition

$$b - Ax_m \perp L_m, \tag{2}$$

where L_m is another subspace of dimension m. A Krylov subspace method is a method for which the subspace K_m is the Krylov subspace

$$K_m(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\}, \tag{3}$$

in which $r_0 = b - Ax_0$.

Arnoldi's algorithm introduced in 1951 builds an orthogonal basis of the Krylov subspace K_m. The following is a version of the algorithm based on the modified Gram-Schmidt process.

ALGORITHM **1 Arnoldi**

1. **Start.** *Choose a vector v_1 of norm 1.*

2. **Iterate.**

3. *For $j = 1, 2, \ldots, m$ do:*

   - *Compute $w_j := Av_j$*
   - *For $i = 1, \ldots, j$ do* $\left\{ \begin{array}{l} h_{ij} = (w_j, v_i) \\ w_j := w_j - h_{ij} v_i \ , \end{array} \right.$
   - *Compute $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ stop.*
   - *Compute $v_{j+1} = w_j / h_{j+1,j}$.*

We denote by V_m the $n \times m$ matrix with column vectors $v_1, \ldots, v_m$, by $\bar{H}_m$ the $(m+1) \times m$ Hessenberg matrix whose nonzero entries are defined by the algorithm. and by $\bar{H}_m$ the matrix obtained from by deleting its last row is denoted by H_m. Then the following relations hold:

$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^H && (4) \\ &= V_{m+1} \bar{H}_m, && (5) \\ V_m^H A V_m &= H_m \ . && (6) \end{aligned}$$

The algorithm breaks down in case the norm of w_j vanishes at a certain step j. In fact, it can be shown that Arnoldi's algorithm breaks down at step j (i.e., $w_j = 0$ in Algorithm 1) if and only if the minimal polynomial of v_1 is of degree j. In this situation, the subspace K_j is invariant.

Using an orthogonal projection method onto the Krylov subspace consists of imposing the condition that the residual vector be orthogonal to the subspace K_m. The Full Orthogonalization Method (FOM) [12] uses the Arnoldi basis to implement this condition. Given an initial guess x_0 to the linear system (1) we consider the choice

$$v_1 = r_0 / \beta, \quad \beta \equiv \|r_0\|_2 \tag{7}$$

in the Arnoldi procedure. Then from (6) we have $V_m^T A V_m = H_m$, and by (7) we have,

$$V_m^T r_0 = V_m^T (\beta v_1) = \beta e_1.$$

As a result, writing the sought approximate solution as $x = x_0 + V_m y$, the orthogonality condition

$$b - A[x_0 + V_m y] \perp \text{span}\{V_m\}$$

immediately yields

$$V_m^T [r_0 - AV_m y] = 0.$$

Therefore, the approximate solution using the above m-dimensional subspaces is given by

$$\begin{aligned} x_m &= x_0 + V_m y_m, & (8)\\ y_m &= H_m^{-1}(\beta e_1). & (9)\end{aligned}$$

The Generalized Minimum Residual Method (GMRES) [8] is a projection method which minimizes the residual norm over all vectors in the affine subspace $x_0 + K_m$. The implementation of an algorithm based on this approach exploits again the Arnoldi basis. Writing again any vector x in $x_0 + K_m$ in the form $x = x_0 + V_m y$, where y is an m-vector we now define,

$$J(y) = \|b - Ax\|_2 = \|b - A\,(x_0 + V_m y)\,\|_2\ . \qquad (10)$$

Exploiting the relation (5), and the fact that V_{m+1} is orthonormal it is easily shown that

$$J(y) = \|\beta e_1 - \bar{H}_m y\|_2\ .$$

As a result of the above formula, the GMRES approximation from the m-th Krylov subspace can be obtained as

$$\begin{aligned} x_m &= x_0 + V_m y_m \quad \text{where} & (11)\\ y_m &= \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2\ . & (12)\end{aligned}$$

The minimizer y_m is inexpensive to compute since it requires the solution of an $(m+1) \times m$ least squares problem and m is typically small.

In order to solve the least squares problem $\min \|\beta e_1 - \bar{H}_m y\|$, it is natural to transform the Hessenberg matrix into upper triangular form by using plane rotations. We define the rotation matrices

$$\Omega_i = \begin{array}{cc} \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} & \begin{array}{l} \\ \\ \\ \leftarrow \text{row } i \\ \leftarrow \text{row } i+1 \\ \\ \\ \\ \end{array} \end{array}, \qquad (13)$$

with $c_i^2 + s_i^2 = 1$. If we are dealing with the GMRES approximation from K_m, then these matrices are of size $(m+1) \times (m+1)$.

We can multiply the Hessenberg matrix $\bar{H}_m$ and the corresponding right-hand-side $\bar{g}_0 \equiv \beta e_1$ by a sequence of such matrices from the left, at each time choosing the coefficient s_i, c_i so as to eliminate $h_{i+1,i}$. For example, when $m = 5$, after applying the 5 such rotations, we transform the problem into,

$$\bar{H}_5^{(5)} = \begin{pmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & 0 \end{pmatrix}, \quad \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \cdot \\ \cdot \\ \gamma_6 \end{pmatrix}. \tag{14}$$

The scalars c_i and s_i of the ith rotation Ω_i are defined by

$$s_i = \frac{h_{i+1,i}}{\sqrt{(h_{ii}^{(i-1)})^2 + (h_{i+1,i})^2}}, \quad c_i = \frac{h_{ii}^{(i-1)}}{\sqrt{(h_{ii}^{(i-1)})^2 + (h_{i+1,i})^2}}. \tag{15}$$

We now define Q_m to be the product of the matrices Ω_i,

$$Q_m = \Omega_m \Omega_{m-1} \dots \Omega_1, \tag{16}$$

and

$$\bar{R}_m = \bar{H}_m^{(m)} = Q_m \bar{H}_m \tag{17}$$

$$\bar{g}_m = Q_m(\beta e_1) = (\gamma_1, \dots, \gamma_{m+1})^T . \tag{18}$$

Then, Q_m is unitary and as a result we have

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

The solution to the above least squares problem is obtained by simply solving the triangular system resulting from ignoring the last row of the matrix $\bar{R}_m$ and right-hand-side $\bar{g}_m$ in (14). In addition, it is clear that for the solution y_* the 'residual' norm $\|\beta e_1 - \bar{H}_m y_*\|$ is nothing but the last element of the right-hand-side, i.e., the term γ_6 in the above illustration.

Consider the residual vector associated with a generic vector in $x_0 + K_m$ of the form $x = x_0 + V_m y$. We have

$$\begin{aligned} b - Ax &\equiv b - A(x_0 + V_m y) \\ &= \beta v_1 - V_{m+1} \bar{H}_m y \\ &= V_{m+1} Q_m^T \, Q_m \, (\beta e_1 - \bar{H}_m y) \\ &= V_{m+1} Q_m^T \, (\bar{g}_m - \bar{R}_m y) \ . \end{aligned}$$

Thus, because of the fact that the last row of $\bar{R}_m$ is zero, the 2-norm of $\bar{g}_m - \bar{R}_m y$ is minimized when y annihilates all components of the right hand side $\bar{g}_m$ except the last one, which is equal to γ_{m+1}. Therefore,

$$b - Ax_m = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1}) \tag{19}$$

and, as a consequence,

$$\|b - Ax_m\|_2 = |\gamma_{m+1}| \ . \tag{20}$$

In practice, this QR procedure is implemented in a progressive manner, i.e., at each step of the GMRES algorithm the QR factorization is performed on the new column of $\bar{H}_m$. This allows us in particular to have the residual norm at every step, with virtually no additional arithmetic operations. The idea is simply to save the previous rotations, then apply them on each newly computed column of $\bar{H}_m$. Once this is done we can then determine the last rotation needed to eliminate $h_{m+1,m}$. For details see [8].

3 DQGMRES: a Truncated version of GMRES

In GMRES and FOM the dimension m of the Krylov subspace increases by one at each step and this makes the procedure impractical for large m. There are two standard remedies for this. The first is to restart the algorithm. The dimension m is fixed and the algorithms is restarted as many times as necessary for convergence, with an initial vector defined as equal to the latest approximation obtained from the previous outer iteration.

A popular alternative is to resort to a truncation of the vector recurrences. In this context we would like to truncate the Arnoldi recurrence to obtain a procedure which is described next.

3.1 Incomplete Arnoldi Orthogonalization

In the incomplete Arnoldi procedure, we select an integer k and perform the following 'incomplete' Gram-Schmidt orthogonalization.

ALGORITHM **2 Incomplete Arnoldi Procedure:**

1. For $j = 1, 2, .., m$ do:

   (a) Compute $w := Av_j$;

   (b) For $i = \max\{1, j-k+1\}, \ldots, j$ do $\left\{ \begin{array}{l} h_{i,j} = (w, v_i), \\ w := w - h_{ij} v_i \end{array} \right.$

   (c) Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w / h_{j+1,j}$.

The only difference with the full Arnoldi orthogonalization is that at each step the current vector is orthogonalized only against the k previous ones instead of all of them. The vectors generated by the above algorithm are known to be 'locally' orthogonal to each other, in that

$$(v_i, v_j) = \delta_{ij} \quad \text{for} \quad |i-j|<k.$$

In addition, the relations (4) – (5) are still valid, but the matrix H_m now has a particular structure, namely, it is banded Hessenberg with total bandwidth of $k+1$, since $h_{i,j} = 0$ for $i \leq j-k$.

The IOM algorithm [12, 11] is defined similarly to the FOM algorithm except that the Arnoldi vectors obtained are not orthogonal but locally orthogonal. Because of the band structure of H_m, it is possible to derive a CG-like scheme called Direct IOM (DIOM) whereby the approximate solution x_m is updated from x_{m-1} by $x_m = x_{m-1} + \zeta_m p_m$ and p_m is obtained from a short term recurrence. For detail see [11].

3.2 DQGMRES

Similarly to IOM we can define an Incomplete GMRES algorithm which we call Quasi GMRES (QGMRES) for the sake of notational uniformity with other existing algorithms developed in the literature. In simple terms, QGMRES implements the GMRES procedure in which the Arnoldi Algorithm is replaced by the Incomplete Orthogonalization Algorithm 2. This technique was first described by Brown and Hindmarsh [2] who reported some numerical tests with it in the context of systems of ODE's.

ALGORITHM **3 Quasi-GMRES Algorithm**
Run a modification of GMRES algorithm in which the Arnoldi process is replaced by the Incomplete Orthogonalization process and every other computation remains unchanged.

Note that we are implicitly ignoring the fact that vectors obtained from the incomplete Orthogonalization procedure are not fully orthonormal. Thus, the approximate solution does not obey any obvious optimality property.

In the Incomplete Arnoldi process we only need to keep the k previous v_i's. However, although this may potentially save us some computations, it does not save any storage, since when we compute the solution by the formula (11) we must again access all the vectors v_i. Fortunately, it is also possible to implement a 'Direct' version called DQGMRES the goal of which is to obtain the approximation progressively, i.e., in a CG-like scheme whereby the approximate solution x_m is updated from x_{m-1} using auxiliary vectors which obey a short-term recurrence. The derivation of DQGMRES is very similar to that of DIOM.

We first note that if $\bar{H}_m$ is banded as for example when $m = 5, k = 2$,

$$\bar{H}_5 = \begin{pmatrix} h_{11} & h_{12} & & & \\ h_{21} & h_{22} & h_{23} & & \\ & h_{32} & h_{33} & h_{34} & \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \; g = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{21}$$

then the premultiplications by the rotations matrices Ω_i as described in the previous section, will only introduce an additional diagonal, in this case we will get,

$$\bar{H}_5^{(5)} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & & \\ & r_{22} & r_{23} & r_{24} & \\ & & r_{33} & r_{34} & r_{35} \\ & & & r_{44} & r_{45} \\ & & & & r_{55} \\ & & & & 0 \end{pmatrix}, \quad \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ . \\ . \\ \gamma_6 \end{pmatrix}. \tag{22}$$

The DQGMRES solution is given by,

$$x_m = x_0 + V_m R_m^{-1} g_m,$$

where R_m and g_m are obtained by removing the last row of $\bar{R}_m$ and $\bar{g}_m$ respectively. Similarly to DIOM [12] we define

$$P_m \equiv V_m R_m^{-1}.$$

Then,

$$x_m = x_0 + P_m g_m \ .$$

We also note that similarly with DIOM, we have

$$g_m = \begin{bmatrix} g_{m-1} \\ \gamma_{m+1} \end{bmatrix}$$

in which

$$\gamma_m = c_m \gamma_m^{(m-1)}, \qquad \gamma_{m+1} = -s_m \gamma_m^{(m-1)}$$

where $\gamma_m^{(m-1)}$ is the last component of the vector $\bar{g}_{m-1}$, i.e., the right-hand-side before the m-th rotation is applied. As a result, x_m can be updated at each step, via the relation,

$$x_m = x_{m-1} + \gamma_m p_m \ .$$

ALGORITHM 4 *DQGMRES*

1. **Start:** *Choose an initial guess* x_0 *then compute* $r_0 = b - Ax_0$ *and* $\gamma_0 := \|r_0\|_2$, $v_1 := r_0/\gamma_1$.

2. **Loop:** *For* $m = 1, 2, \ldots,$ *until convergence do,*

   1. *Compute* h_{im}, $i = \max\{1, m-k+1\}, \ldots, m-1$, *and* v_{m+1} *as in Steps 1-a, 1-b, 1-c of Algorithm 2.*
   2. *Update the QR factorization of* $\bar{H}_m$, *i.e.,*
      - *Apply the rotations* Ω_i, $i = m-k, \ldots, m-1$ *to the* m*-th column of* $\bar{H}_m$ *just computed;*
      - *Compute the rotation coefficients* c_m, s_m *by (15);*
   3. *Apply rotation* Ω_m, *to* $\bar{H}_m$ *and* $\bar{g}_m$, *i.e., compute*
      - $\gamma_{m+1} := -s_m \gamma_m$,
      - $\gamma_m := c_m \gamma_m$, *and,*
      - $h_{mm} := c_m h_{mm} + s_m h_{m+1,m}, \quad (= \sqrt{h_{m+1,m}^2 + h_{mm}^2})$
   4. $p_m = \left(v_m - \sum_{i=m-k}^{m-1} h_{im} p_i \right) / h_{mm}$
   5. $x_m = x_{m-1} + \gamma_m p_m$
   6. *If* $|\gamma_{m+1}|$ *is small enough then stop.*

4 Properties of DQGMRES

The DQGMRES algorithm does not minimize the norm of the residual vector over $x_0 + K_m$ but attempts to perform an approximate minimization. Indeed, formula (19) is still valid since orthogonality is not used to derive it. If the v_i's were orthogonal to each other then we are back in the situation of GMRES and the residual norm is minimized over all vectors of the form $x_0 + V_m y$. Since the v_i's are only locally orthogonal we will only realize an approximate minimization. In addition, we no longer have the relation (20) which provides the residual norm from a known quantity. Indeed, this relation had been derived by exploiting the orthogonality of the v_i's. It turns out that in practice $|\gamma_{m+1}|$ remains a reasonably good estimate of the actual residual norm in general because of the fact that the v_i's are nearly orthogonal. The following inequality which is easy to show

$$\|b - Ax_m\| \leq \sqrt{m+1} |\gamma_{m+1}| \tag{23}$$

provides an actual upper bound of the residual norm in terms of computable quantities. The proof of this inequality is an immediate consequence of (19).

If we call $q = (\eta_i)_{i=1,\ldots,m+1}$ the unit vector $q = Q_m^T e_{m+1}$ then

$$\begin{aligned} \|b - Ax_m\|_2 &= |\gamma_{m+1}| \, \|V_{m+1} q\|_2 = |\gamma_{m+1}| \left\| \sum_{i=1}^{m+1} v_i \eta_i \right\|_2 \\ &\leq |\gamma_{m+1}| \sum_{i=1}^{m+1} \|v_i\|_2 |\eta_i| \\ &\leq |\gamma_{m+1}| \sqrt{m+1}. \end{aligned}$$

The last relation is due to the Cauchy-Schwartz inequality. As a result, using $|\gamma_{m+1}|$ as a residual estimate, we will be making an error by a factor of $\sqrt{m+1}$ at most. In general, this is an overestimate and $|\gamma_{m+1}|$ tends to give enough accuracy as an estimate for the residual norm.

It is also interesting to observe that if we are willing to sacrifice a little bit of arithmetic, we can actually compute the exact residual vector and norm. This is based on the observation that, according to (19), the residual vector is γ_{m+1} times the vector z_{m+1} which is the last column of the matrix

$$Z_{m+1} \equiv V_{m+1} Q_m^T$$

It is easy to verify that this last column can be updated from v_{m+1} and z_m. Indeed,

$$\begin{aligned} Z_{m+1} &= [V_m, v_{m+1}] Q_{m-1}^T \Omega_m^T \\ &= [V_m Q_{m-1}^T, v_{m+1}] \Omega_m^T \\ &= [Z_m, v_{m+1}] \Omega_m^T, \end{aligned}$$

and by equating the last columns of both sides, we get,

$$z_{m+1} = -s_m z_m + c_m v_{m+1}. \qquad (24)$$

The z_i's can be updated at the cost of one extra vector in memory and $3n$ operations at each iteration. The norm of z_{m+1} can be computed at the cost of $2n$ operations and the exact residual norm for the current approximate solution can then be obtained by multiplying this norm by $|\gamma_{m+1}|$,

$$\|r_m\| = |\gamma_{m+1}| \, \|z_{m+1}\| \, . \qquad (25)$$

This is a little expensive so we may elect to just 'correct' the estimate provided by γ_{m+1} by exploiting the above recurrence relation and writing,

$$\|z_{m+1}\|_2 \leq |s_m| \, \|z_m\|_2 + |c_m|.$$

If we set $\zeta_m \equiv \|z_m\|_2$, then we have the recurrence relation,

$$\zeta_{m+1} \leq |s_m| \zeta_m + |c_m|. \qquad (26)$$

The above relation which costs virtually nothing to update provides an upper bound that is sharper than (23).

An important characteristic of DQGMRES is that it is *flexible*, i.e., it allows variations in the preconditioner. Specifically, when right preconditioning is used, the preconditioner M is allowed to vary at each step. The idea is similar to that of FGMRES [14]. In both cases we must compute the vectors $M_j^{-1}v_j$'s and in the case of FGMRES, we need to save these vectors which requires extra storage [14]. In the case of DQGMRES, this is no longer required since the preconditioned vectors only affect the update of the vector p_j in the formula,

$$p_j = \frac{1}{h_{jj}} \left(M_j^{-1} v_j - \sum_{i=j-k+1}^{j-1} h_{ij} p_i \right).$$

Thus, $M_j^{-1}v_j$ can be discarded immediately after it is used in the above formula. In fact, we can simply overwrite it onto the space used for p_j and modify step 4 of Algorithm 4 accordingly. Because of this flexible preconditioning feature, we can use DQGMRES in a nested fashion, for example, GMRES or DQGMRES can be used as preconditioners to DQGMRES. We note that similar features are also presented in the GMRESR family of algorithms introduced by Van der Vorst and Vuik [16] and in the algorithms presented by Axelsson and Vassilevski [1].

We next examine the relation between DQGMRES and the full GMRES algorithm. In what follows we denote by r_m^Q and r_m^G the residuals of the m-th iterate obtained by DQGMRES and GMRES respectively. With this we can state the following result.

Theorem 1 *If both DQGMRES and the full GMRES do not break-down, then,*

$$\|r_m^Q\|_2 \leq \kappa_2(V_{m+1}) \|r_m^G\|_2, \tag{27}$$

where $\kappa_2(V_{m+1})$ is the condition number of V_{m+1} with respect to the 2-norm.

This result was first given in an early draft of this paper [9]. It is proven by Jia [6]. The proof uses the pseudo-inverse V_{m+1}^{+} to express the DQGMRES residual. A similar result was proved for the QMR algorithm [5, 7].

As is known, the GMRES algorithm breaks down at a given step if and only if the exact solution is found at this step. Next we study the break-down behavior of DQGMRES.

Theorem 2 *Assuming A is non-singular, if DQGMRES breaks down at step m, then x_m is exact.*

Proof. There are two divisions in Algorithm 4, one in the Incomplete Arnoldi Process, and the other in step 2.4. If the incomplete Arnoldi process breaks down, $h_{m+1,m}$ is zero. If step 2.4 breaks down, $h_{mm}^{(m)}$ is zero which implies that $\bar{H}_m$ is rank deficient. Since A is not singular, and V_m is a basis for $K_m(A, r_0)$, by equation (5) $\bar{H}_m$ must have full column rank. Thus if DQGMRES breaks down, $h_{m+1,m}$ must be zero.

If $h_{m+1,m}$ is zero, then s_m is zero (equation (15)), γ_{m+1} in Algorithm 4 step 2.3 is zero, and r_m is zero according to equation (25). □

We should remark that, according to the proof, the only way in which DQGMRES can break down for a non-singular matrix, is when the Incomplete Arnoldi Process breaks down, i.e., when $h_{m+1,m} = 0$. For a Krylov subspace method, $h_{m+1,m} = 0$ implies that it is possible to compute the exact solution. An important point here is that the exact solution can be computed.

Theorem 3 *Let A be a non-singular matrix. If DQGMRES finds the exact solution at step m, and not in a previous step, then the minimal polynomial for r_0 is of degree m.*

Proof. According to equation (25), if DQGMRES finds the exact solution, we have either $\gamma_{m+1} = 0$ or $z_{m+1} = 0$. Since DQGMRES did not find the exact solution in previous steps, γ_i and z_i must be nonzero for for $i = 1, 2, \ldots, m$.

If $\gamma_{m+1} = 0$, then $s_m = 0$. According to equation (15), $h_{m+1,m}$ must be zero in this case. Therefore the grade of r_0, i.e., the degree of the minimal polynomial for r_0 is m.

Since z_{m+1} is a linear combination of $r_0, Ar_0, \ldots, A^m r_0$, if z_{m+1} is zero, the grade of r_0 must be $\leq m$. That the grade is exactly m follows from the relation (24). First, we observe that we cannot have $c_m = 0$. Otherwise (24) would give us $z_{m+1} = z_m$ and this would lead to the contradiction that $z_m = 0$. Therefore, by (24) z_{m+1} has a nonzero component in v_{m+1} which can easily be proven to have a nonzero component in $A^m v_1$. Hence, the grade of r_0 is m. □

Similarly to GMRES, if DQGMRES breaks down, we have a *lucky* breakdown. If GMRES finds the exact solution, it will break down, but DQGMRES may not necessarily break down if it finds the exact solution. In the full GMRES, the grade of r_0 is equal to $m - 1$ if and only if $h_{m+1,m} = 0$. In other words GMRES breaks down iff the grade of r_0 is equal to $m - 1$ iff x_m is exact. However, this is not true for DQGMRES, and the results must be separated as was done above.

In practice, the lucky breakdown scenario is very unlikely to occur except in special cases. Theorem 2 and 3 seem to hint that DQGMRES is less likely to break down than the full GMRES essentially because DQGMRES cannot reach the exact answer as fast as the GMRES.

5 Numerical Experiments

We now illustrate the behavior of DQGMRES with a few numerical tests a collection of problems. The methods we will refer to in this section are the following.

1. **BiCGSTAB** Bi-Conjugate Gradient method stabilized, a method due to van der Vorst [18] and based on the bi-CG algorithm.

2. **TFQMR** Transpose-Free Quasi-Minimum Residual method due to Freund [4].

3. **GMRES** Generalized Minimum Residual method of Saad and Schultz [8]. Specifically, we use a restarted version, i.e., GMRES(m).

4. **DIOM** The Direct implementation of Incomplete Orthogonalization Method [12].

5. **ORTHMIN** The full version of ORTHOMIN [17, 3] is mathematically equivalent to GMRES. The truncated version ORTHMIN(k) is similar in spirit to DQGMRES(k). For details see, e.g., [3].

6. **FGMRES** The Flexible-GMRES [14] allows variable right-preconditioning.

7. **DQGMRES** The new method described in section 4.

Both TFQMR and DQGMRES compute the exact residual norm since we wish to use the residual norm in the stopping test for all iterative solvers. Thus TFQMR and DQGMRES are more expensive than their cheapest forms respectively.

Table 1 shows the required workspace size and number of floating-point operations averaged over the number of matrix-vector multiplications used by each iterative solver. Most of the iterative solvers are from the iterative solution module of SPARSKIT and PSPARSLIB [13, 10]. Both complexity and number of FLOP shown here are as implemented in SPARSKIT. In particular they do not account for the small order costs associated with the solutions of the small projected matrix problems. The FLOP reported are the FLOP per matrix-vector multiplication excluding the operations used by the matrix-vector multiplications. We implemented the Arnoldi and incomplete Arnoldi processes with re-orthogonalization. This further modification of the Gram-Schmidt process performs a reorthogonalization only when it suspects a significant loss of accuracy due to cancellation.

	space	FLOP
BiCGSTAB	8n	11n
TFQMR	12n	19n
GMRES(m)	(m+2)n	(2m+5)n
FGMRES(m)	2(m+1)n	(2m+5)n
ORTHMIN(k)	2(k+1)n	(6k+8)n
DIOM(k)	(2k+1)n	(6k+4)n
DQGMRES(k)	2(k+1)n	(6k+12)n

Table 1: Complexity of the iterative solvers used.

This makes GMRES and its variants more expensive than what is shown in Table 1. However, the parameters are chosen so that re-orthogonalization is performed very infrequently.

The linear systems tested are constructed from the matrices to have random solutions. The iterative solvers are given a different set of random vectors as initial guesses. The test matrices are part of a collection extracted from the solution of the fully coupled Navier-Stokes Equations in the test examples provided in FIDAP – a fluid dynamics package. These examples consist of a wide variety of problems from Couette flow to turbulent flow, some of which include heat transfer and chemical reactions. The elements used are of type Q2 for the velocity and P1 for the pressure for 2D problems. For three-dimensional problems, the package uses the so-called Brick elements (8 nodes). The grids used in the problems are regularly structured but not uniform. There are 35 matrices in the set, out of which 13 are symmetric, and one is very small in size. We excluded these 14 small or symmetric matrices from the tests. We show in Table 2 the names of the matrices used along with their sizes and number of nonzero elements.

We report the results of applying 14 different solvers on the 21 linear systems constructed. In our tests, a solution x_j is considered to have converged if the residual satisfies,

$$\|r_j\| \le 10^{-6}\|r_0\| + 10^{-12}.$$

The abbreviations used in the table headings are explained in Table 3. The iterative solvers are allowed to use 1000 matrix-vector multiplications in the first part of this experiment. In the inner-outer schemes, we allow a total of up to 10,000 matrix-vector multiplications, altogether.

Table 4 compares the number of successful convergence cases for different values of the maximum number of matrix-vector multiplications allowed. This test is done without using any preconditioning. We observe that

Name	N	NNZ	Name	N	NNZ
EX6	1651	49533	EX25	848	24612
EX7	1633	54543	EX26	2163	74464
EX8	3096	90841	EX27	974	40782
EX11	16614	109648	EX28	2603	77781
EX18	5773	71805	EX29	2870	23754
EX19	12005	259577	EX31	3909	91223
EX20	2203	69981	EX35	19716	227872
EX21	656	19144	EX36	3079	53843
EX22	839	22715	EX37	3565	67591
EX23	1409	43703	EX40	7740	458012
EX24	2283	48737			

Table 2: The 21 test matrices from FIDAP.

$\mathcal{B}$	BiCGSTAB
$\mathcal{T}$	TFQMR
$\mathcal{G}$	GMRES
$\mathcal{D}$	DIOM
$\mathcal{O}$	ORTHMIN
$\mathcal{Q}$	DQGMRES

Table 3: Abbreviations used in table headings.

DQGMRES(k) always solves more problems than DIOM(k). DQGMRES(k) is also better than GMRES($2k$). ORTHMIN(k) and DQGMRES(k) perform similarly in this case. The number of linear systems solved by BiCGSTAB and TFQMR is less than that of DIOM(k), ORTHMIN(k) and DQGMRES(k), but is more than that of GMRES(m). The performances of DIOM(k), ORTHMIN(k) and DQGMRES(k) do not seem to depend significantly on the value of k.

Tables 5–6 show similar results when preconditioning is used. Three preconditioners are employed.

- Scaling. The matrix is scaled both from the left and right. The ith row is scaled by the square root of the row norm, and the jth column is scaled by the square root of the column norm, $a_{ij} = a_{ij}/\sqrt{\|a_{i\cdot}\|\|a_{\cdot j}\|}$. The objective is to reduce the difference between the ith row norm and the ith column norm.

	$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(10)$	$\mathcal{G}(20)$	$\mathcal{G}(40)$	$\mathcal{D}(5)$	$\mathcal{D}(10)$
100	1	2	1	2	2	2	2
500	3	3	3	3	3	3	3
1000	4	4	3	3	3	5	5
	$\mathcal{D}(20)$	$\mathcal{O}(5)$	$\mathcal{O}(10)$	$\mathcal{O}(20)$	$\mathcal{Q}(5)$	$\mathcal{Q}(10)$	$\mathcal{Q}(20)$
100	2	3	3	3	3	3	3
500	3	4	4	4	4	4	4
1000	5	7	7	7	7	7	7

Table 4: Number of successful convergence cases versus number of matrix-vector multiplications allowed for unpreconditioned linear systems.

	$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(10)$	$\mathcal{G}(20)$	$\mathcal{G}(40)$	$\mathcal{D}(5)$	$\mathcal{D}(10)$
50	4	4	4	4	5	3	4
100	5	5	4	6	7	4	5
500	9	10	7	8	9	8	7
1000	10	10	7	8	11	8	8
	$\mathcal{D}(20)$	$\mathcal{O}(5)$	$\mathcal{O}(10)$	$\mathcal{O}(20)$	$\mathcal{Q}(5)$	$\mathcal{Q}(10)$	$\mathcal{Q}(20)$
50	4	3	3	4	3	4	4
100	5	4	4	6	4	5	5
500	8	5	6	7	8	7	8
1000	8	5	6	7	8	8	8

Table 5: Number of successful convergence cases versus number of matrix-vector multiplications allowed for preconditioned linear systems.

- The ILUT(k, ϵ) preconditioner [15] is an incomplete LU factorization with a dual dropping strategy. The number of fill-in elements in both L and U factors are limited by k, and those elements with magnitude less than ϵ of the row norm are dropped. The ILUT preconditioned used here is ILUT(10, 10^{-8}).

- The ILUTP(k, ϵ, τ) preconditioner. This is an extension of ILUT which performs column pivoting. The pivoting threshold is τ. We used ILUTP(10, 10^{-8}, 0.02).

The preconditioners are used in the order shown here. First, the scaling is used on all 21 linear systems, 5 are solved. We then apply ILUT(10, 10^{-8}) to the 16 linear systems left. Four additional systems are solved this way. The ILUTP(10, 10^{-8}, 0.02) is used on the unconverged 12, of which two more

	$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(10)$	$\mathcal{G}(20)$	$\mathcal{G}(40)$	$\mathcal{D}(5)$	$\mathcal{D}(10)$
scale	5	5	3	3	5	4	4
ILUT	3	3	3	3	4	3	3
ILUTP	2	2	1	2	2	1	1

	$\mathcal{D}(20)$	$\mathcal{O}(5)$	$\mathcal{O}(10)$	$\mathcal{O}(20)$	$\mathcal{Q}(5)$	$\mathcal{Q}(10)$	$\mathcal{Q}(20)$
scale	4	4	4	4	4	4	4
ILUT	3	1	2	3	3	3	3
ILUTP	1	0	0	0	1	1	1

Table 6: Number of successful convergence cases for each preconditioner.

converged. Table 5 shows results from all the preconditioners altogether. Table 6 shows the results of each of the three preconditioners.

Table 5 shows that if less than 100 matrix-vector multiplications are allowed, GMRES(40) solves more linear systems than all others. This is somewhat expected. TFQMR takes the lead if 500 matrix-vector multiplications are allowed, followed closely by BiCGSTAB and GMRES(40). The number of linear systems solved by ORTHMIN(k) is less than DQGMRES(k) and DIOM(k). Similarly to the unpreconditioned case, the performances of DIOM(k) and DQGMRES(k) do not depend significantly on k, whereas ORTHMIN(k) does show some dependence on k, and GMRES(m) shows the strongest dependence on its parameter m. From Table 6 we can see that GMRES(40) works well with ILUT and ILUTP preconditioner, while ORTHMIN does not do as well, especially for small values of the parameter k.

	unpreconditioned				preconditioned			
	$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(40)$	$\mathcal{Q}(20)$	$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(40)$	$\mathcal{Q}(20)$
$\mathcal{B}$	0	1	1	0	0	6	2	4
$\mathcal{T}$	2	0	1	0	4	0	2	4
$\mathcal{G}(40)$	3	3	0	0	9	9	0	5
$\mathcal{Q}(20)$	7	7	6	0	6	6	2	0

Table 7: Number of cases where the methods heading the rows are better than the methods heading the columns.

The next table, Table 7, reflects the efficiency of the solvers. It reports the number of cases where the method heading a row is considered *better* than the method heading a column. When solving one linear system, method

A is considered to be *better* than method B if (1) method A converges but not method B; or, (2) method A uses at least 2 matrix-vector multiplications[1] less than method B. A method can win over another one only if it has converged on a problem.

Using this comparison criterion, we can see that DQGMRES(20) compares very favorably with BiCGSTAB, TFQMR, and GMRES(40). For example, there are 7 linear systems where DQGMRES(20) works more efficiently than BiCGSTAB without preconditioning, while there is no linear system where BiCGSTAB works faster than DQGMRES(20). Using roughly the same amount of workspace, DQGMRES(20) solves 6 linear systems faster than GMRES(40) without preconditioning. However when preconditioned, GMRES(40) is more efficient than DQGMRES(20). There are 14 methods shown in the Tables 4–5. We can build a table similar to Table 7 with all 14 methods, the following table shows the row sum of this 14 $\times$ 14 table for the unpreconditioned case,

$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(10)$	$\mathcal{G}(20)$	$\mathcal{G}(40)$	$\mathcal{D}(5)$	$\mathcal{D}(10)$
10	10	1	5	18	23	22
$\mathcal{D}(20)$	$\mathcal{O}(5)$	$\mathcal{O}(10)$	$\mathcal{O}(20)$	$\mathcal{Q}(5)$	$\mathcal{Q}(10)$	$\mathcal{Q}(20)$
29	53	58	69	57	64	71

The number shown for each column represents the total number of instances where the method indicated is found better than another method. For example there were a total of 18 instances where GMRES(40) was better than another method. Similarly, we find the following counts for the preconditioned case,

$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(10)$	$\mathcal{G}(20)$	$\mathcal{G}(40)$	$\mathcal{D}(5)$	$\mathcal{D}(10)$
58	55	28	54	86	27	28
$\mathcal{D}(20)$	$\mathcal{O}(5)$	$\mathcal{O}(10)$	$\mathcal{O}(20)$	$\mathcal{Q}(5)$	$\mathcal{Q}(10)$	$\mathcal{Q}(20)$
40	25	34	46	34	40	51

According to the criterion used, we observe that in the unpreconditioned case DQGMRES(20) is the most effective method. In the preconditioned case GMRES(40) leads others with a wide margin, while DQGMRES(20) is now behind BiCGSTAB, TFQMR, and GMRES(20).

DQGMRES(5) solved 7 linear systems without preconditioning which is 3 more than most of other methods (see Table 4). These three linear systems are constructed from matrices EX8, EX18 and EX35. A surprising fact is that when these matrices are scaled, DQGMRES can not solve any one of them, nor can any of the other iterative solvers used, even with

[1]A gap of 2 matrix-vector multiplication is chosen here because BiCGSTAB and TFQMR uses 2 matrix-vector multiplication in one iteration.

ILUT(10,10^{-8}) and ILUTP(10,10^{-8},0.02). If these three linear systems are so hard to solve, why is DQGMRES(k) capable of solving them *without preconditioning*? After investigating this, we concluded that the reason may be the variation in the degree of normality of the matrices involved. We define

$$\mathcal{N} = \frac{\|AA^H - A^H A\|_F}{\|AA^H\|_F}$$

to be our indicator of deviation from normality for A, where $\|\cdot\|_F$ denotes the Frobenius norm. The following table shows the $\mathcal{N}$ values before and after the scaling.

	Before Scaling		After Scaling	
	$\mathcal{N}$	MATVEC	$\mathcal{N}$	MATVEC
EX7	5×10^{-11}	100	3×10^{-4}	68
EX11	1×10^{-16}	919	2×10^{-16}	156
EX29	2×10^{-5}	61	8×10^{-4}	13
EX37	1×10^{-7}	61	1×10^{-8}	20
EX8	6×10^{-9}	499	4×10^{-3}	> 1000
EX18	2×10^{-10}	778	0.44	> 1000
EX35	3×10^{-10}	845	0.41	> 1000

The columns headed with MATVEC are the number of matrix-vector multiplications used to solve this linear system by DQGMRES(5). We observe that DQGMRES(5) did not solve any problem where $\mathcal{N} > 10^{-3}$. This observation suggests that DQGMRES is sensitive to normality, in that it does tend to perform poorly for matrices that are far from normal.

outer	FGMRES(20)				DQGMRES(20)			
inner	$\mathcal{B}$	$\mathcal{G}(20)$	$\mathcal{O}(10)$	$\mathcal{Q}(10)$	$\mathcal{B}$	$\mathcal{G}(20)$	$\mathcal{O}(10)$	$\mathcal{Q}(10)$
100	3	3	3	3	3	3	3	3
1000	6	6	6	7	6	6	6	7
5000	12	19	18	20	11	18	18	20
10000	19	20	19	21	18	20	18	21

Table 8: Number of linear systems solved using the inner-outer iteration schemes.

Table 8 shows some tests of the flexible preconditioning feature of the new method. To test this feature, we constructed an inner-outer iteration scheme where the outer iterations are FGMRES(20) or DQGMRES(20), the right-preconditioners for the outer iterative solvers are one of BiCGSTAB, GMRES(20), ORTHMIN(10) or DQGMRES(10). The inner iterative solvers

	$\mathcal{B}$	$\mathcal{G}(20)$	$\mathcal{G}(40)$	$\mathcal{O}(10)$	$\mathcal{Q}(10)$	$\mathcal{Q}(20)$
100	2	3	3	3	3	3
1000	5	3	5	4	4	4
5000	16	5	7	8	8	8
10000	19	6	11	9	10	10

Table 9: Number of linear systems solved by the iterative solvers used in the inner-outer iteration scheme.

will terminate if $\|r_j\| \leq 0.1 \times \|r_0\| + 10^{-12}$ or more than 100 matrix-vector multiplications have been consumed. In this part of the experiment we scale the matrices as described earlier prior to calling the iterative solvers. The data in Table 8 show that DQGMRES(20) could work just as well as FGMRES(20) using the same amount work space and the same number of inner iterations. DQGMRES(10) seems to be the best inner iterative solver, both FGMRES(20) and DQGMRES(20) were able to solve all linear systems using DQGMRES(10) as preconditioner. Table 9 shows the results of allowing the iterative solvers involved in the inner-outer iteration scheme to use up to 10,000 matrix-vector multiplications. From comparing Tables 8 and 9 it is clear that the inner-outer iteration scheme is more effective than allowing more iterations in any one particular scheme used on its own.

6 Conclusion

The DQGMRES algorithm compares well with the best Krylov subspace techniques available for solving linear systems. Similarly to GMRES, it can only encounter *lucky* break-downs. Our experience indicates that without preconditioning, DQGMRES(k) usually performs as well as GMRES($2k$). When preconditioned is used our experience does not show a clear advantage of DQGMRES over GMRES or vice-versa.

Without preconditioning DQGMRES(k) behaves very similarly to ORTHMIN(k). With preconditioning, DQGMRES(k) does generally better than ORTHMIN(k). Both DQGMRES(k) and ORTHMIN(k) save the k most recent Krylov subspace basis vectors. The algorithms differ in the inner product used for enforcing the orthogonality of the basis vectors. DQGMRES(k) uses 2-norm, while ORTHMIN(k) uses A^TA-norm. When the condition number of A is large, A^TA may be numerically rank-deficient which could cause ORTHMIN(k) to produce linearly dependent basis vectors. Therefore, the orthogonality of the basis vectors can be more easily maintained in DQGMRES(k), in general.

The performance of DQGMRES(k) does not depend as significantly on the size of k as GMRES(m) does on m. This is an advantage of DQGMRES(k) since it it can perform quite well with a rather small size of workspace used. On the negative side, it is a disadvantage when compared with GMRES(m) which typically benefits much more from increased workspace size when facing a hard problem.

DQGMRES(k) works well when the matrix is normal or nearly normal. In the case when the matrix is symmetric, DQGMRES(1) can be as effective as GMRES without restart. ¿From a number experiments which we did not report in this paper, we also found that DQGMRES(k), for k>1 appears to perform considerably better than the Conjugate Gradient method.

DQGMRES is *flexible*, it allows the preconditioner to vary at each step. While FGMRES(m) uses about twice the workspace as GMRES(m) in order to achieve this feature, DQGMRES requires no modification, and therefore no additional workspace. With this feature we can use arbitrary iterative linear system solvers as preconditioners. The resulting inner-outer iteration scheme is often a very robust procedure. In this case, DQGMRES(k) can be used as a good outer loop solver as well as a very effective inner loop preconditioner.

Acknowledgment.
The authors would like to acknowledge the support of the Minnesota Supercomputer Institute which provided the computer facilities and an excellent research environment to conduct this research.

References

[1] O. Axelsson and P.S. Vassilevski. A block generalized conjugate gradient solver with inner iterations and variable step preconditioning. *SIAM J. Mat. Anal.*, 12, 1991.

[2] P.N. Brown and A.C. Hindmarsh. Matrix-free methods for stiff systems of ODEs. *SIAM J. Numer. Anal.*, 23:610–638, 1986.

[3] S. C. Eisenstat, H. C. Elman, and M. H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20:345–357, 1983.

[4] Roland W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14(2):470–482, March 1993.

[5] R.W. Freund and N.M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[6] Zhongxiao Jia. On IGMRES(q): incomplete generalized minimal residual methods for large unsymmetric linear systems. Technical Report 94-047, Department of Mathematics, University of Bielefeld, 1994. Last revision March, 1995.

[7] Noel M. Nachtigal. *A look-ahead variant of the Lanczos Algorithm and its application to the Quasi-Minimal Residual method for non-Hermitian linear systems.* PhD thesis, Applied Mathematics, MIT, Cambridge, Massachusetts, 1991.

[8] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[9] Y. Saad and K. Wu. DQGMRES: a quasi-minimal residual algorithm based on incomplete orthogonalization. Technical Report UMSI-93/131, Minnesota Supercomputing Institute, Minneapolis, 1993.

[10] Y. Saad and K. Wu. Design of an iterative solution module for a parallel matrix library (p_sparslib). In W. Schonauer, editor, *Proceedings of IMACS conference, Georgia, 1994*, 1995. TR 94-59, Department of Computer Science, University of Minnesota.

[11] Youcef Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comput.*, 37:105–126, 1981.

[12] Youcef Saad. practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 5:203–228, 1984.

[13] Youcef Saad. SPARSKIT: A basic toolkit for sparse matrix computations. Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990.

[14] Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, March 1993.

[15] Youcef Saad. ILUT: a dual threshold incomplete ilu factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994. Technical Report 92-38, Minnesota Supercomputer Institute, University of Minnesota, 1992.

[16] H.A. van der Vorst and C. Vuik. GMRESR: a family of nest GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386, 1994.

[17] P. K. W. Vinsome. Orthomin, an iterative method for solving sparse sets of simultaneous linear equations. In *Proceedings of the Fourth Symposium on Reservoir Simulation*, pages 149–159. Society of Petroleum Engineers of AIME, 1976.

[18] H.A. Van Der Vorst. BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13:631–644, 1992.