Numerical Linear Algebra with Applications, Vol. 3(4), 329–343 (1996)

DQGMRES: a Direct Quasi-minimal Residual Algorithm Based on Incomplete Orthogonalization

Yousef Saad and Kesheng Wu

University of Minnesota, Department of Computer Science, 200 Union St. S.E., Minneapolis, MN 55455, USA

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. Like GMRES, it does not break down. Numerical experiments show that DQGMRES(k) often performs as well as the restarted GMRES using a subspace of dimension $m = 2k$. 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 the use of any iterative solver as a right-preconditioner for DQGMRES(k). This inner-outer iterative combination often results in a robust approach for solving indefinite non-Hermitian linear systems.

KEY WORDS iterative methods; GMRES; Krylov methods; incomplete orthogonalization; quasi-minimization

1. Introduction

In recent years, there has been a flurry of activity in the general area of iterative methods for solving large sparse linear systems of equations. This is spurred in part by the increased need to solve three-dimensional problems efficiently. 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 [12] is a method which minimizes the residual norm over such approximations and is, at least in its standard non-restarted form, optimal. There are also a number of Krylov subspace methods that do not obey similar global optimality properties but which perform quite well in practice. Among these are methods such as the bi-conjugate gradient method and techniques derived from it, such as BiCGSTAB[22] and TFQMR[9]. In this paper we will present an algorithm in this category which combines quasi-minimization concepts and incomplete orthogonalization [3,15,16,23]. To be specific, the Arnoldi vectors are only orthogonalized against a small

CCC 1070–5325/96/040329–15

Received 1 January 1995
Revised 7 January 1996

number of previous vectors, a technique which we refer to as incomplete orthogonalization [3]. In addition, the algorithm attempts to extract an approximate smallest-residual norm solution, via a *quasi-minimization* process. This quasi-optimal solution is obtained in a manner identical to GMRES, but the non-orthogonality of the basis of the Krylov subspace resulting from the incomplete orthogonalization [6,8] is ignored.

Since the algorithm to be presented is closely related to existing ones, we will start by defining the process and describing its main features in the next section. Section 3 describes a few properties of the algorithm. Section 4 reports the results of an extensive numerical experimentation. Finally, Section 5 contains a few concluding remarks.

2. DQGMRES: a truncated version of GMRES

GMRES [12] is a common iterative method used for solving non-Hermitian linear system of equations. The algorithm uses the Arnoldi process [1] to generate an orthonormal basis of the Krylov subspace and extracts an *optimal* solution of this subspace. In GMRES 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, with an initial vector defined as the latest approximation obtained from the previous outer iteration. This is denoted by GMRES(m). An alternative is to truncate the long-recurrence of the Arnoldi process, as is discussed next.

2.1. *Incomplete Arnoldi orthogonalization*

In the incomplete Arnoldi procedure, an integer k is selected and the following 'incomplete' Gram–Schmidt orthogonalization is performed.

Algorithm 2.1. Incomplete Arnoldi procedure: *Given* v_1, *for* $j = 1, 2, .., m$ *do:*

1. *Compute* $w := Av_j$;
2. *For* $i = \max\{1, j-k+1\}, \ldots, j$ *do* $\begin{cases} h_{i,j} = (w, v_i) \\ w := w - h_{ij} v_i \end{cases}$
3. *Compute* $h_{j+1,j} = \|w\|_2$ *and* $v_{j+1} = w/h_{j+1,j}$

While the full Arnoldi procedure orthogonalizes w against all previous v_i's, the above algorithm only orthogonalizes w against the k most recent v_i's. The result is that the vectors generated by the above algorithm are *locally* orthogonal to each other, i.e., they satisfy the relation:

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

Let $V = (v_1, \ldots, v_m)$, $\bar{H}_m = (h_{i,j})$, $i = 1, \ldots, m+1$, $j = 1, \ldots, m$. Then we have the following important relation:

$$AV_m = V_{m+1}\bar{H}_{m+1} \tag{2.1}$$

This is identical with a relation satisfied by the Arnoldi process, the only difference being that the matrix $\bar{H}_m$ generated by the new algorithm is banded Hessenberg and the v_i's are not orthonormal. The total bandwidth of $\bar{H}_m$ is $k+1$, since $h_{i,j} = 0$ for $i \leq j-k$.

The incomplete Arnoldi process can now be used to define a truncated GMRES variant. This technique was first described by Brown and Hindmarsh [3] who reported some numerical tests with it in the context of systems of ODEs.

2.2. *QGMRES and DQGMRES*

It is possible to simply replace the full Arnoldi process in GMRES with Algorithm 2.1, while the rest of the GMRES algorithm is carried out exactly as in GMRES. The new algorithm associated with this approximate solution will be referred to as quasi-GMRES (QGMRES)—(this is identical with the incomplete GMRES (IGMRES) in [3]). However, QGMRES requires saving all the v_i's as in GMRES, in order to compute the solution x_m at the end of the Arnoldi loop. Fortunately, it is possible to obtain the solution via a short-term recursion. This ingredient referred to as a *direct implementation,* is used in the DIOM algorithm [16] and later in the QMR algorithm [8]. We call Direct QGMRES (DQGMRES) the resulting algorithm.

Because of the band structure of $\bar{H}_m$, it is possible to obtain a progressive update scheme 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. We briefly outline the main points of the technique, and refer the reader to [15,16] for additional details.

Consider the least-squares problem to be solved at the end of the QGMRES process, involving the banded Hessenberg matrix $\bar{H}_m$, and the right-hand side $\bar{g}$, as for example, when $m = 5, k = 2$,

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

After premultiplication by a sequence of Givens rotations, Ω_i, $i = 1, \ldots, m$ to eliminate the subdiagonal, $h_{21}, \ldots, h_{65}$, the above least-squares problem becomes

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

In which the superscript $^{(5)}$ indicates the number of Givens rotations applied.

Defining $Q_m \equiv \Omega_m \ldots \Omega_1$, $\bar{R}_m \equiv Q_m \bar{H}_m$, the QGMRES solution is given by,

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

where R_m and g_m are obtained from $\bar{R}_m$ and $\bar{g}_m$ by removing their last rows respectively. Following the notation of DIOM [16] let

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

and observe that thanks to the particular structure of R_m, the last column p_m of P_m can be updated via,

$$p_m = \frac{1}{r_{mm}} \left(v_m - \sum_{i=m-k}^{m-1} r_{im} p_i \right) \tag{2.2}$$

The above expression for x_m becomes, $x_m = x_0 + P_m g_m$ and, similar to DIOM, g_m and g_{m-1} are simply related by

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

where γ_{m+1} is an updatable scalar. As a result, x_m can be updated at each step, via the relation,

$$x_m = x_{m-1} + \gamma_m p_m \tag{2.3}$$

Equations (2.3) and (2.2) define the DQGMRES recurrence. It remains to show how to update γ_m.

The scalars γ_{m+1}, satisfy the recurrence

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

where $\gamma_m^{(m-1)}$ is the last component of the vector $\bar{g}_{m-1}$, i.e., the right-hand-side before the mth rotation is applied, c_m and s_m are the cosine and sine of the last Given rotation angle, where

$$s_m = \frac{h_{m+1,m}}{\sqrt{|h_{mm}^{(m-1)}|^2 + |h_{m+1,m}|^2}}, \quad c_m = \frac{h_{mm}^{(m-1)}}{\sqrt{|h_{mm}^{(m-1)}|^2 + |h_{m+1,m}|^2}} \tag{2.4}$$

More details on the derivation can be found in [13]. The DQGMRES algorithm is as follows.

Algorithm 2.2. DQGMRES

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

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

1. *Compute* h_{im}, $i = \max\{1, m-k+1\}, \ldots, m-1$, *and* v_{m+1} *as in Steps 1–3 of Algorithm 2.1.*
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 equation (2.4);*
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.*

Table 1. Complexity of some iterative solvers

	space	flops
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)	$2(k+1)n$	$(6k+12)n$

The memory requirement of this algorithm is about $(2k+1)n$ where n is the size of the matrix. The average number of floating-point operations (flops) per DQGMRES(k) iteration is about $(6k+7)n$ plus the matrix–vector multiplication and preconditioning operation. If the exact residual norm is to be computed (see next section), then one additional vector of storage and $5n$ additional flops are required. In Table 1, the complexity of a real implementation is shown [14].

3. Properties of DQGMRES

In this section, we state a few properties of the DQGMRES algorithm and compare it with GMRES.

3.1. Residual vectors

If K_m represents the usual Krylov subspace $K_m(A, r_0)$, then DQGMRES does not minimize the norm of the residual vector over $x_0 + K_m$ but attempts to perform an approximate minimization. It is easy to show that the residual norm of $x_m = x_0 + V_m y_m$ obtained from DQGMRES is

$$b - Ax_m = V_{m+1}(\beta e_1 - \bar{H}_m y_m) \tag{3.1}$$

$$= V_{m+1} Q_m^{\mathrm{T}} (\gamma_{m+1} e_{m+1}) \tag{3.2}$$

The residual norm of GMRES satisfies the same formulas. Both GMRES and QGMRES (and therefore also DQGMRES) obtain y_m by minimizing $\|\beta e_1 - \bar{H}_m y\|_2$ over y. In the case of GMRES, V_{m+1} consists of a set of orthonormal columns and therefore, according to (3.1), the corresponding x_m minimizes the residual norm over $x_0 + K_m$. This not true of QGMRES. Since the v_i's are only locally orthogonal QGMRES can only achieve an approximate minimization.

In Step 2.6 of Algorithm 2.2, the size of $|\gamma_{m+1}|$ is used to monitor convergence. In GMRES this is a perfectly valid measure of the residual norm, thanks to (3.2) and the fact that V_{m+1} and Q_m are both unitary. For DQGMRES $|\gamma_{m+1}|$ remains a reasonably good estimate of the actual residual norm because of the fact that the v_i's are nearly orthonormal. The following inequality

$$\|b - Ax_m\| \leq \sqrt{m-k+1}\,|\gamma_{m+1}| \tag{3.3}$$

provides an upper bound of the residual norm in terms of computable quantities. This inequality is an immediate consequence of equation (3.2) and the fact that the vectors v_{m-k}, ..., v_{m+1} are orthonormal. In general, this expression gives an overestimate of the actual residual norm and $|\gamma_{m+1}|$ tends to give enough accuracy as an estimate for the residual norm.

If we are willing to sacrifice a little bit of arithmetic, we can actually compute the exact residual vector and its norm. This is based on the observation that, according to (3.2), the residual vector is equal to $\gamma_{m+1} z_{m+1}$, where z_{m+1} is the last column of the matrix

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

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

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

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

$$z_{m+1} = -s_m z_m + c_m v_{m+1} \tag{3.4}$$

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 $|\gamma_{m+1}|$, i.e.,

$$\|r_m\| = |\gamma_{m+1}| \, \|z_{m+1}\| \tag{3.5}$$

Spending $5n$ flops to compute the residual norm may be considered expensive in some cases. We may prefer to just *correct* the estimate provided by γ_{m+1} by exploiting the above recurrence relation,

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

If we set $\zeta_m \equiv \|z_m\|_2$, then

$$\zeta_{m+1} \leq |s_m| \, \zeta_m + |c_m|$$

The above relation which costs virtually nothing to update provides a dynamic upper bound that should be sharper than (3.3).

3.2. *Flexible preconditioning*

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 [18]. In both cases we must compute the vectors $M_j^{-1} v_j$. For FGMRES, we need to save these vectors which requires extra storage. 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 2.2 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 [21] and in the algorithms presented by Axelsson and Vassilevski [2]. This feature is particularly useful in situations where a good preconditioner is not readily available, or in domain decomposition-type techniques when the preconditioner is expressed in terms of subdomain solves. These solves can be performed by an iterative procedure instead of a direct solution technique as is often done.

3.3. Relations with full GMRES

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 mth iterate obtained by DQGMRES and full GMRES respectively. With this we can state the following result.

Proposition 3.1. *If both DQGMRES and the full GMRES do not break down, then,*

$$\|r_m^Q\|_2 \leq \kappa_2(V_{m+1})\|r_m^G\|_2 \tag{3.6}$$

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

This result is a simple adaptation of a similar result proved in [11]. See also [13] and [10]. The proof uses the pseudo-inverse V_{m+1}^+ to express the DQGMRES residual. A general theory of quasi-minimization can be found in [6,7] and particular results for the QMR and TFQMR algorithms can be found in [8,11].

As is known, GMRES encounters what is referred to as a *lucky break-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.

Proposition 3.2. *Assuming A is non-singular, if DQGMRES breaks down at step m, then x_m is exact.*

Proof There are two divisions in Algorithm 2.2, 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 $h_{m+1,m}$ is zero according to the expression of $h_{m,m}^{(m)}$ given in step 2.3 of the algorithm. Thus, if DQGMRES breaks down, $h_{m+1,m}$ must be zero.

If $h_{m+1,m}$ is zero, then s_m is zero, γ_{m+1} in Algorithm 2.2, Step 2.3 is zero, and r_m is zero according to equation (3.5). ∎

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 $h_{m+1,m} = 0$. This type of break down does not prevent the computation of x_m, which is exact in this case.

Proposition 3.3. *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 (3.5), 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 non-zero for for $i = 1, 2, \ldots, m$.

If $\gamma_{m+1} = 0$, then $s_m = 0$. According to equation (2.4), $h_{m+1,m}$ 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 equation (3.4). First, we observe that we cannot have $c_m = 0$. Otherwise (3.4) would give us $z_{m+1} = z_m$ and this would lead to the contradiction that $z_m = 0$. Therefore, by (3.4) z_{m+1} has a non-zero component in v_{m+1} which can easily be proven to have a non-zero 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* break-down. If the grade of r_0 is equal to $m - 1$, GMRES will find the exact solution at step m. However, this is not true for DQGMRES. Equation (3.6) does not imply $\|r_m^{\mathcal{Q}}\| \leq 0$, because V_{m+1} is rank deficient.

In practice, the lucky break-down scenario is very unlikely to occur except in special cases. Propositions 3.2 and 3.3 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.

There are a few trivial cases where DQGMRES(k) and the full GMRES behave in an identical manner, for example, when the grade of r_0 is less than k, or when the matrix is symmetric or per-symmetric. Generally speaking, DQGMRES(k) is equivalent to the full GMRES algorithm if and only if A is a matrix such that

$$A^H v \in K_k(A, v) \equiv \text{span}\{v, Av, \ldots, A^{k-1}v\}, \forall v$$

or the degree of its minimal polynomial is not higher than k. Following the notation of [5], let $\nu(A)$ be the smallest degree of polynomial q such that $A^H = q(A)$. Faber and Manteuffel [5] have shown that a matrix A satisfies $A^H v \in K_k(A, v)$ for every vector v if and only if A is normal and $\nu(A) < k$. The next proposition is a simple consequence of this result.

Proposition 3.4. *DQGMRES(k) is equivalent to the full GMRES algorithm if and only if the degree of minimal polynomial is not higher than k or A is normal and* $\nu(A) < k$.

4. Numerical experiments

We now illustrate the behavior of DQGMRES(k) with a few numerical tests on 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 [22] and based on the Bi-CG algorithm.
2. **TFQMR** Transpose-free quasi-minimum residual method due to Freund [9].
3. **GMRES** Generalized minimum residual method of Saad and Schultz [12]. Specifically, we use a restarted version, i.e., GMRES(m).
4. **DIOM** The direct implementation of incomplete orthogonalization method [16].
5. **ORTHMIN** The full version of ORTHOMIN, GCR, [4,20] is mathematically equivalent to GMRES. The truncated version ORTHMIN(k) is similar to DQGMRES(k) in that

Table 2. The 21 test matrices from FIDAP

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			

they are both truncated variants of families of Krylov subspace methods using long-recurrence. For details see, e.g., [4].

6. **FGMRES** The flexible-GMRES [18] allows variable right-preconditioning.
7. **DQGMRES** The new method described in Section 2.

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.

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 [14,17]. Both complexity and number of flops 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 flops reported are the flops per matrix–vector multiplication excluding the operations used by the matrix–vector multiplication and preconditioning. We implemented the Arnoldi and incomplete Arnoldi processes with reorthogonalization. The modified Gram–Schmidt process performs a reorthogonalization only when it suspects a significant loss of accuracy due to cancellation. This makes GMRES and its variants more expensive than what is shown in Table 1. However, the parameters are chosen so that reorthogonalization is performed very infrequently.

The linear systems tested are constructed to have pseudo-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 finite element based 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 finite elements used are of type $Q2$ for the velocity and $P1$ for the pressure for 2D problems. For 3D problems, the package uses the so-called Brick elements (eight 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 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 dimensions and number of non-zero elements.

Table 3. Abbreviations used in table headings

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

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

	$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(10)$	$\mathcal{G}(20)$	$\mathcal{G}(40)$	$\mathcal{D}(5)$	$\mathcal{D}(10)$
100	1	2	1	2	2	2	2
500	3	3	3	3	3	3	3
1 000	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
1 000	5	7	7	7	7	7	7

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,

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

The abbreviations used in the table headings are explained in Table 3. The iterative solvers are allowed to use 1 000 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 any preconditioning. Overall, we observe that DQGMRES(k) and ORTHMIN(k) are relatively more successful than other solvers.

Table 5 shows 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 goal is to reduce the difference between the ith row norm and the i-th column norm.
- The ILUT(k, ϵ) preconditioner [19] 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).

Table 5. Number of successful convergence cases versus number of matrix–vector multiplications allowed for preconditioned 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 6. Number of cases where the methods heading the rows are better than the methods heading the columns

unpreconditioned

	$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(40)$	$\mathcal{Q}(20)$	
$\mathcal{B}$	0	1	1	0	2
$\mathcal{T}$	2	0	1	0	3
$\mathcal{G}(40)$	3	3	0	0	6
$\mathcal{Q}(20)$	7	7	6	0	20
	12	11	8	0	

preconditioned

	$\mathcal{B}$	$\mathcal{T}$	$\mathcal{G}(40)$	$\mathcal{Q}(20)$	
$\mathcal{B}$	0	6	2	4	12
$\mathcal{T}$	4	0	2	4	10
$\mathcal{G}(40)$	9	9	0	5	23
$\mathcal{Q}(20)$	6	6	2	0	14
	19	21	6	13	

The preconditioners are used in the order shown here. First, the scaling is used on all 21 linear systems, five are solved. We then apply ILUT(10, 10^{-8}) to the 16 linear systems left. Four additional systems are solved with this preconditioner. The ILUTP(10, 10^{-8}, 0.02) is used on the remaining 12 linear systems, of which two more converged. Table 5 shows results from all the preconditioners altogether.

Table 5 shows that if no more than 100 matrix–vector multiplications are allowed, GMRES(40) succeeds in solving more linear systems than all other methods. TFQMR takes the lead if 500 matrix–vector multiplications are allowed, followed closely by BiCGSTAB and GMRES(40). Generally speaking, GMRES(40) is more successful with the preconditioners used in the tests.

The next table, Table 6, reflects the comparative 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 two

Table 7.

	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

matrix–vector multiplications[1] less than method B. A method can be *better* than another one in solving one linear system only if it has converged for the problem. The numbers at the end of each row in the table are the row sums. The numbers below the columns are the column sums. With the given comparison criterion, the larger the row sum, the more competitive the solver; the smaller the column sum, the better the solver. According to the criteria used, we observe that in the unpreconditioned case DQGMRES(20) is more effective than other methods. In the preconditioned case GMRES(40) is better.

DQGMRES(5) solved seven linear systems without preconditioning, which is three more than most other solvers (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(5) cannot solve any one of them, nor can any of the other iterative solvers tested, even with ILUT(10,10^{-8}) and ILUTP(10,10^{-8},0.02) as preconditioners. If these three linear systems are so *hard* to solve, why is DQGMRES(k) capable of solving them *without preconditioning*? We think the degree of non-normality of the matrices may be the reason. Define

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

to be the indicator of deviation from normality for A, where $\|\cdot\|_F$ denotes the Frobenius norm. Table 7 shows the $\mathcal{N}$ values before and after the scaling.

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 for which $\mathcal{N} > 10^{-3}$. Note that $\mathcal{N}$ is bounded from above by $\sqrt{2}$. However, there does not seem to be any obvious correlation between the number of steps used and $\mathcal{N}$ for those linear systems for which DQGMRES(5) converged. This observation suggests that DQGMRES is sensitive to normality, but normality may not be the determining factor for the convergence of DQGMRES(k).

The tests in Table 8 illustrate the flexible preconditioning feature of the new method. We constructed an inner-outer iteration scheme where the outer iteration is one of FGMRES(20) or DQGMRES(20) and the right-preconditioners for these outer iterative solvers are one of BiCGSTAB, GMRES(20), ORTHMIN(10) or DQGMRES(10). The inner iterative solvers terminate if $\|r_j\| \leq 0.1 \times \|r_0\| + 10^{-12}$, or more than 100 matrix–vector

[1] A gap of two matrix–vector multiplications is chosen here because BiCGSTAB and TFQMR uses two matrix–vector multiplications in one iteration.

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

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
1 000	6	6	6	7	6	6	6	7
5 000	12	19	18	20	11	18	18	20
10 000	19	20	19	21	18	20	18	21

Table 9. Number of linear systems solved using extremely large number of matrix–vector multiplications

	$\mathcal{B}$	$\mathcal{G}(20)$	$\mathcal{G}(40)$	$\mathcal{O}(10)$	$\mathcal{Q}(10)$	$\mathcal{Q}(20)$
100	2	3	3	3	3	3
1 000	5	3	5	4	7	7
5 000	16	5	7	8	8	8
10 000	19	6	11	9	10	10

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 of 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 a preconditioner. Table 9 shows the results of allowing the inner iterative solvers to use the same limit for the number of matrix–vector multiplications as the outer iteration. Thus, the iteration is stopped when either this limit (10 000) is reached or when the residual satisfies the criterion $\|r_j\| \leq 10^{-6} \times \|r_0\| + 10^{-12}$. A comparison of Tables 8 and 9 indicates that, in terms of overall robustness, this inner–outer iteration scheme is more effective than any one particular solver used on its own. It is also interesting to note that BiCGSTAB without preconditioning performs significantly better than other iterative schemes, when an extremely large number of matrix–vector products is allowed.

5. Conclusion

Two attractive features of the new algorithm presented in this paper are that, like GMRES, it can only encounter *lucky* break-downs and, like FGMRES, is is *flexible*, i.e., it allows the preconditioner to vary at each step. In contrast with FGMRES(m), however, this flexibility comes at no extra storage penalty.

The extensive numerical experiments with DQGMRES reported in the previous section indicate that, for the problems tested, DQGMRES is as effective as the best known Krylov subspace methods. Compared with some of its closest relatives, DQGMRES(k) often performed better than GMRES($2k$) in the unpreconditioned cases we tried. For the preconditioned cases, GMRES($2k$) typically outperformed DQGMRES(k). Without preconditioning, DQGMRES(k) tended to succeed in solving the same linear systems as ORTHMIN(k). With preconditioning, DQGMRES(k) performed generally better than ORTHMIN(k). Finally, DQGMRES(k) almost always performed better than DIOM(k) in our experiments.

DQGMRES(k) may work well when the matrix is normal or nearly normal. In the case where the matrix is symmetric, DQGMRES(1) can be as effective as GMRES without restart (Proposition 3.4). From a number of experiments not reported in this paper, we also found that DQGMRES(k), for $k > 1$ appears to perform considerably better than the conjugate gradient method for *symmetric* linear systems.

Acknowledgements

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

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. W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9, 17–29, 1951.
2. 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.
3. P. N. Brown and A. C. Hindmarsh. Matrix-free methods for stiff systems of ODEs. *SIAM J. Numer. Anal.*, 23, 610–638, 1986.
4. 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.
5. V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21, 352–362, 1984.
6. R. W. Freund. Quasi-kernel polynomials and their use in Non-Hermitian matrix iterations. *Journal of Computational and Applied Mathematics*, 34, 135–158, 1992.
7. R. W. Freund. Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations. In *Numerical Methods of Approximation Theory*, Braess and Schumack, editors, Birkhauser, Basel, 1992, pp. 77–95.
8. R. W. Freund and N. M. Nachtigal. QMR, a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60, 315–339, 1991.
9. R. W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14(2), 470–482, 1993.
10. Z. Jia. On IGMRES(q), incomplete generalized minimal residual methods for large unsymmetric linear systems. Technical Report 94-047, Department of Mathematics, University of Bielefeld, Sonderforschungsbereich 343, 1994. Last revision March, 1995.
11. N. N. 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.
12. 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.
13. 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.
14. Y. Saad and K. Wu. Design of an iterative solution module for a parallel matrix library (P_SPARSLIB). In *Proceedings of IMACS conference, Georgia, 1994*, W. Schonauer, editor, 1995. Was TR 94-59, Department of Computer Science, University of Minnesota.
15. Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comput.*, 37, 105–126, 1981.
16. Y. Saad. Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 5, 203–228, 1984.

17. Y. 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. Software currently available at <ftp,//ftp.cs.umn.edu/dept/sparse/>.
18. Y. Saad. A flexible inner–outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2), 461–469, 1993.
19. Y. Saad. ILUT, a dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications*, 1, 387–402, 1994.
20. 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*, pp. 149–159. Society of Petroleum Engineers of AIME, 1976.
21. H. A. van der Vorst and C. Vuik. GMRESR, a family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4), 369–386, 1994.
22. 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.
23. H. F. Walker and L. Zhou. A simpler GMRES. *Numerical Linear Algebra with Applications*, 1, 571–581, 1994.