DQGMRES: a Quasi – minimal residual algorithm based on incomplete orthogonalization *

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

October, 1993

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 the parent algorithm from which it is derived, DQGMRES does not require restarting. It seems also to be less prone to stagnation. In addition, the algorithm allows flexible preconditioning, i.e., it can accomodate variations in the preconditioner at every step. A number of numerical tests are reported which show that the algorithm often performs better than GMRES and FGMRES.

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

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 the recent years, much of which motivated 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 q_{m-1} is a polynomial of degree $\leq m-1$, x_0 is an initial guess and $r_0 = b - Ax_0$ is the corresponding residual vector.

The GMRES algorithm 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 that seem to do well in practice. These are methods such as the bi-conjugate gradient methods and techniques derived from it such as BiCGSTAB and TFQMR. In this paper we will present a technique 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, we will attempt to get an approximate smallest-residual norm solution, which is obtained by a process that assumes that the Arnoldi vectors are orthogonal.

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. Section 3 and 4 describes the idea of incomplete orthogonalization methods and the DQGMRES variant. Section 5 reports on some numerical experiments. A tentative conclusion is drawn in Section 6.

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 $r_0 = b - Ax_0$. The different versions of Krylov subspace methods arise from different choices of the subspaces K_m and L_m, and from the different ways of building bases for these subspaces. Among these methods, an important sub-class relies on orthogonal bases. We will begin this section by describing the well-known Arnoldi process for generating orthogonal bases of Krylov subspaces. A few variations of these techniques are based on what is referred to as incomplete orthogonalization which will be discussed in Section 3.

2.1 Arnoldi orthogonalization

Arnoldi's algorithm introduced in 1951 builds an orthogonal basis of the Krylov subspace K_m. In exact arithmetic, the basic variant of the algorithm is as follows.

ALGORITHM **2.1 Arnoldi -(Modified Gram-Schmidt)**

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* $i = 1, \ldots, j-1$ *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* $v_{j+1} = w_j / h_{j+1,j}$.

If we denote by V_m the $n \times m$ matrix with column vectors $v_1, \ldots, v_m$ and by $\bar{H}_m$ the $(m+1) \times m$ Hessenberg matrix whose nonzero entries are defined by the algorithm, and H_m the matrix obtained from $\bar{H}_m$ by deleting its last row, 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 will break 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 (2.1)) if and only if the minimal polynomial of v_1 is of degree j. In this situation, the subspace K_j is invariant.

2.2 FOM and GMRES

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. Several methods based on this approach have been developed in the past, one of which we refer to as the Full Orthogonalization Method (FOM) [5]. 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)

$$V_m^T r_0 = V_m^T (\beta v_1) = \beta_1.$$

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

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

and as a result the approximate solution using the above m-dimensional subspaces is given by

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

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 is very similar to that of the FOM algorithm. We start by describing the main idea and then we will discuss some practical variations. To derive GMRES we write again any vector x in $x_0 + K_m$ in the form $x = x_0 + V_m y$, where y is an m-vector and define,

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

Exploiting the relation (5), we can easily show that

$$J(y) = \|V_{m+1}\,(\beta e_1 - \bar{H}_m y)\,\|_2$$

and since the column-vectors of V_{m+1} are orthonormal then

$$J(y) \equiv \|b - A\,(x_0 + V_m y)\,\|_2 = \|\beta e_1 - \bar{H}_m y\|_2. \quad (11)$$

As a result of the above formula, the GMRES approximation from the m-th Krylov subspace can be obtained quite simply as $x_0 + V_m y_m$ where y_m minimizes the function $J(y) = \|\beta e_1 - \bar{H}_m y\|_2$, i.e.,

$$\begin{aligned} x_m &= x_0 + V_m y_m \quad \text{where} \quad (12) \\ y_m &= \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2 \ . \quad (13) \end{aligned}$$

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

$$\Omega_i = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_i & s_i & & \\ & & & -s_i & c_i & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow \text{row } i \\ \leftarrow \text{row } i+1 \\ \\ \\ \\ \end{matrix} , \quad (14)$$

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 $h_{i+1,i}$. Thus if $m = 5$ we would have,

$$\bar{H}_5 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix} \quad , \quad \bar{g}_0 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad . \tag{15}$$

Then we first premultiply $\bar{H}_5$ by Ω_1 with

$$s_1 = \frac{h_{21}}{\sqrt{|h_{11}|^2 + |h_{21}|^2}}, \quad c_1 = \frac{h_{11}}{\sqrt{|h_{11}|^2 + |h_{21}|^2}}$$

to obtain the matrix and right hand side

$$\bar{H}_5^{(1)} = \begin{pmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} & h_{15}^{(1)} \\ 0 & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} & h_{25}^{(1)} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix} \quad , \quad \bar{g}_1 = \begin{pmatrix} c_1\beta \\ -s_1\beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{16}$$

We can now premultiply the above matrix and right hand side again by a rotation matrix Ω_2 to eliminate h_{32}. This process of eliminating the subdiagonal elements is continued this way until we apply the m-th rotation which transforms the problem into,

$$\bar{H}_5^{(5)} = \begin{pmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & 0 \end{pmatrix} \quad , \quad \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \cdot \\ \cdot \\ \gamma_6 \end{pmatrix} \tag{17}$$

The scalars c_i and s_i of the i^{th} rotation Ω_i are defined by

$$s_i = \frac{h_{i+1,i}}{\sqrt{|h_{ii}^{(i-1)}|^2 + |h_{i+1,i}|^2}}, \quad c_i = \frac{h_{ii}^{(i-1)}}{\sqrt{|h_{ii}^{(i-1)}|^2 + |h_{i+1,i}|^2}} \quad . \tag{18}$$

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

$$Q_m = \Omega_m \Omega_{m-1} \ldots \Omega_1, \tag{19}$$

and

$$\bar{R}_m = \bar{H}_m^{(m)} = Q_m \bar{H}_m \tag{20}$$

$$\bar{g}_m = Q_m(\beta e_1) = (\gamma_1, \ldots, \gamma_{m+1})^T \quad . \tag{21}$$

Then, Q_m is unitary and as a result we have

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

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 (17). In addition, it is clear that for the solution y_* the 'residual' $\|\beta e_1 - \bar{H}_m y_*\|$ 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

$$\begin{aligned} b - Ax &\equiv b - A\left(x_0 + V_m y\right) \\ &= r_0 - AV_m y \\ &= \beta v_1 - V_{m+1}\bar{H}_m y \\ &= V_{m+1}\left(\beta e_1 - \bar{H}_m y\right) \\ &= V_{m+1}Q_m^T\ Q_m\left(\beta e_1 - \bar{H}_m y\right) \\ &= V_{m+1}Q_m^T\left(\bar{g}_m - \bar{R}_m y\right)\ . \end{aligned}$$

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}. As a result,

$$b - Ax_m = V_{m+1}Q_m^T(\gamma_{m+1}e_{m+1}) \tag{22}$$

and, as a consequence,

$$\|b - Ax_m\|_2 = |\gamma_{m+1}| \ . \tag{23}$$

We point out that this approach with plane rotations can also be used to solve the linear system (9) for the FOM method. The only difference is that the last rotation Ω_m must be omitted. In particular, a single program can be written to implement both algorithms using a switch for selecting the FOM and GMRES options.

In practice, this procedure is implemented in in a progressive manner, i.e., at each step of the GMRES algorithm. This allows 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 Methods based on incomplete Arnoldi orthogonalization

In the previous algorithms, the dimension m of the Krylov subspace increases by one a each step and this makes the procedure impractical for large m. There are two typical remedies for this. The simplest remedy 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 Orthogonalization

Assume that we select an integer k and that we perform the following 'incomplete' Gram-Schmidt orthogonalization process.

ALGORITHM **3.1 Incomplete Arnoldi Process:**

1. *For* $j = 1, 2, .., m$ *do:*
   (a) *Compute* $w := Av_j$;
   (b) *For* $i = \max\{1, j-k+1\}, \ldots, j$ *do* $\begin{cases} h_{i,j} = (w, v_i), \\ w := w - h_{ij} v_i \end{cases}$
   (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 since $h_{i,j} = 0$ for $i \leq j - k$.

3.2 DIOM

The IOM algorithm [5, 4] is defined similarly to the FOM algorithm except that the Arnoldi vectors obtained are not orthogonal but locally orthogonal. The Hessenberg matrix H_m obtained from the incomplete orthogonalization process has a band structure with a bandwidth of $k+1$. For example when $k = 3$ and $m = 5$ we obtain the following matrix:

$$H_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & & \\ h_{21} & h_{22} & h_{23} & h_{24} & \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \end{pmatrix} \tag{24}$$

We need to keep only the k previous v_i's in the Arnoldi process, and we wish to be able to discard the others. However, if we implement the IOM algorithm, we still face the difficulty that when we compute the solution by formula (8) we will again need all the vectors v_i. One option would be to recompute them at the end, but this will essentially double the cost of the algorithm. This raises the question of developing a formula whereby the approximate solution can be easily computed from the previous approximation x_{m-1} and a small number of vectors that are being updated at each step. A procedure of this type called DIOM (Direct version of IOM) was presented in [5] and we would like describe it briefly for the sake of completeness.

DIOM is derived by writing the LU factorization of H_m as $H_m = L_m U_m$. Assuming no pivoting is used, the matrix L_m is unit lower bidiagonal and U_m is band-upper triangular, with

k diagonals. Thus, for the above matrix, we will have a factorization of the form

$$H_m = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ & l_{32} & 1 & & \\ & & l_{43} & 1 & \\ & & & l_{54} & 1 \end{pmatrix} \times \begin{pmatrix} u_{11} & u_{12} & u_{13} & & \\ & u_{22} & u_{23} & u_{24} & \\ & & u_{33} & u_{34} & u_{35} \\ & & & u_{44} & u_{45} \\ & & & & u_{55} \end{pmatrix} .$$

The approximate solution is then given by,

$$x_m = x_0 + V_m U_m^{-1} L_m^{-1} \beta e_1 .$$

If we define

$$P_m \equiv V_m U_m^{-1} \quad \text{and} \quad z_m = L_m^{-1} \beta e_1 ,$$

then we have,

$$x_m = x_0 + P_m z_m . \tag{25}$$

Because of the structure of U_m we can update P_m very easily via the formula,

$$p_m = \frac{1}{u_{mm}} \left[v_m - \sum_{i=m-k+1}^{m-1} u_{im} p_i \right] .$$

In addition, because of the structure of L_m, we have

$$z_m = \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix}$$

in which

$$\zeta_m = -l_{m,m-1} \zeta_{m-1}.$$

From (25) we have

$$x_m = [P_{m-1}, p_m] \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} = P_{m-1} z_{m-1} + \zeta_m p_m \equiv x_{m-1} + \zeta_m p_m$$

showing that the approximation x_m can be updated at each step from the previous iterate.

We note that a simple formulation of the algorithm which exploits Gaussian Elimination with partial pivoting when solving the Hessenberg system was also developed in [5]. We can also use plane rotations to implementation DIOM, based on a similar observation to the one made earlier regarding the relation between GMRES and FOM.

4 DQGMRES: a Truncated version of GMRES

We can also implement an Incomplete GMRES algorithm which we call Quasi GMRES (QGMRES) for the sake of notational uniformity with other existing algorithms developed in the literature. It is also possible to implement a Direct version called (DQGMRES) using exactly the same arguments as in Section 3 for DIOM.

4.1 QGMRES and DQGMRES

First, we define the QGMRES algorithm, in the simplest terms, by replacing the Arnoldi Algorithm with the Incomplete Orthogonalization Algorithm 3.1. This technique was first described by Brown and Hindmarsh [1] who reported some numerical tests with it in the context of systems of ODE's.

ALGORITHM **4.1 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.

Similarly to IOM, we now need to keep only the k previous vectors so this version of GMRES will potentially save computations but not storage, since when we will need to compute the solution by the formula (12) we must again access all the vectors v_i. Fortunately we can again update the approximate solution in a progressive manner, as was done with DIOM.

The implementation is quite similar to DIOM. We first note that if $\bar{H}_m$ is banded as for example when $m = 5, k = 2$,

$$\bar{H}_5 = \begin{pmatrix} h_{11} & h_{12} & & & \\ h_{21} & h_{22} & h_{23} & & \\ & h_{32} & h_{33} & h_{34} & \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \; g = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{26}$$

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,

$$\bar{H}_5^{(5)} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & & \\ & r_{22} & r_{23} & r_{24} & \\ & & r_{33} & r_{34} & r_{35} \\ & & & r_{44} & r_{45} \\ & & & & r_{55} \\ & & & & 0 \end{pmatrix} \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ . \\ . \\ \gamma_6 \end{pmatrix} \tag{27}$$

The approximate 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. If we define P_m similarly to DIOM,

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

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 **4.2** *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 3.1.*
   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 (18);*
   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. $p_m = \left(v_m - \sum_{i=m-k+1}^{m-1} h_{im} p_i\right) / h_{mm}$
   5. $x_m = x_{m-1} + \gamma_m p_m$
   6. *If* $|\gamma_{m+1}|$ *is small enough then stop.*

The above algorithm does not minimize the norm of the residual vector over $x_0 + K_m$ but attempts to perform an approximate minimization. Indeed, the formula (22) 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 we are only using an incomplete orthogonalization process then the v_is are only locally orthogonal and as a result we will obtain an approximate minimization. In addition, we no longer have (23) which had been derived from the above equation 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

$$\|b - Ax_m\| \leq \sqrt{m+1}|\gamma_{m+1}| \tag{28}$$

provides an actual upper bound of the residual norm in terms of computable quantities. The proof of this inequality is an immediate consequence of (22). If we call $q = (\eta_i)_{i=1,\ldots,m+1}$ the

unit vector $q = Q_m^T e_{m+1}$ then

$$\begin{aligned} \|b - Ax_m\|_2 &= |\gamma_{m+1}| \, \|V_{m+1} q\|_2 = |\gamma_{m+1}| \, \| \sum_{i=1}^{m+1} v_i \eta_i \|_2 \\ &\leq |\gamma_{m+1}| \sum_{i=1}^{m+1} \|v_i\|_2 |\eta_i| \\ &\leq |\gamma_{m+1}| \sqrt{m+1} \end{aligned} \tag{29}$$

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 (22), 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 an easy exercise 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 as a result, we get,

$$z_{m+1} = -s_m z_m + c_m v_{m+1}. \tag{30}$$

The z_i's can be updated at the cost of one extra vector in memory and $3n$ operations at each step. 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 $|\gamma_{m+1}|$.

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,

$$\zeta_{m+1} \leq |s_m| \zeta_m + |c_m| \tag{31}$$

Note that the above relation which costs virtually nothing to update provides an upper bound that is sharper than (29).

An important characteristic of DQGMRES is that it is *flexible* in that it can allow variable preconditioners without requiring additional storage. Specifically, when right preconditioning is used, the preconditioner M is allowed to vary at each step. The idea is similar to that of FGMRES [7]. 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 [7]. 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 4.2 accordingly. We note that DQGMRES is similar in nature to the GMRESR family of algorithms introduced by Van der Vorst and Vuik [9].

Finally, we would like to mention that convergence results identical to those of the QMR algorithm [2, 3] hold. For example, it is easy to prove that the norms of the residuals r_m^Q and r_m^G obtained after m steps of the DQGMRES and GMRES algorithms respectively are related by

$$\|r_m^Q\|_2 \leq \kappa_2(V_{m+1}) \|r_m^G\|_2, \tag{32}$$

assuming that V_{m+1} is of full rank, an assumption that is valid when the degree of the minimal polynomial of r_0 is not less than $m+1$.

4.2 A variation

When comparing DIOM and DQGMRES, we observe that these two algorithms only differ by their last rotation angles. We can choose freely this last rotation angle and this gives us an extra degree of freedom. We may for example think of minimizing the residual norm with respect to this angle. We will now explore this option.

With an arbitrary rotation in the last step of DQGMRES, the residual of the linear system can be expressed as:

$$b - Ax_m = V_{m+1} Q_m^T (\bar{g}_m - \bar{R}_m y) .$$

After we have applied all the previous rotations in step 2.2 of the algorithm 4.2, the variables that will directly influence the selection of the final rotation angle and the residual norm are:

$$h_{mm}, \quad h_{m+1,m}, \quad \gamma_m, \quad v_{m+1}, \quad z_m.$$

If instead of computing the last rotation by equation (18), we use an arbitrary angle of rotation, say θ_m, then it can be shown that the residual norm is given by the following equation:

$$\|r_m\|_2^2 \equiv \|b - Ax_m\|_2^2 = \|z_{m+1}\|_2^2 \gamma_m^2 / \cos^2(\psi_m - \theta_m) , \tag{33}$$

where ψ_m is the rotation angle defined by equation (18). Given this rotation angle θ_m, the vector z can be updated via equation (30) where $c_m = \cos(\theta_m)$ and $s_m = \sin(\theta_m)$. The norm of z_{m+1} can be computed recursively without computing the vector explicitly, since we have

$$\zeta_{m+1}^2 \equiv \|z_{m+1}\|_2^2 = \cos^2(\theta_m) - \eta_m \sin(2\theta_m) + \zeta_m^2 \sin^2(\theta_m) \tag{34}$$

where $\eta_m = z_m^T v_{m+1}$.

The equations (33)–(34) for computing the residual norm have only one free variable, θ_m. Though we cannot compute the exact value of θ_m which minimizes $\|r_m\|_2$ algebraically, we can numerically approximate it. This minimization process can be easily integrated into the algorithm 4.2 to form a new variant of DQGMRES. We will simply use equation (33) to find the

	BiCGSTAB	TFQMR	GMRES(m)	FGMRES(m)	DQGMRES(k)
space	9n	11n	(m+2)n	(2m+2)n	(2k+1)n
time	6n	14n	(m+4)n	(m+1)n	(3k+1)n

Table 1: Complexity of the four iterative solvers, excluding operations with the matrix.

optimal θ_m, then compute the $c_m = \cos(\theta_m)$ and $s_m = \sin(\theta_m)$ instead of using equation (18). At step 2.6 of the algorithm, the exact residual norm can be used.

The main advantage of this new variant of GMRES over DQGMRES or DIOM is that the residual norm is guaranteed not to increase from step to step. Since the residual norm of the previous step can be achieved by letting $\theta_m = \pi/2$, the minimum residual norm achieved at a given step by this method should be no greater than the residual norm of the previous step. This variant requires one more inner-product than the version of DQGMRES which computes the exact residual vector at each step. Compared with Algorithm 4.2, this variant will require an extra vector of storage of size n, and $7n$ additional floating-point operations. In most of the experiments we have performed with this variant, we found that it does not perform significantly better than DQGMRES and that overall the slight gain in performance does not warrant the additional complexity and computational work.

5 Numerical Experiments

For testing purposes we have implemented a version of DQGMRES along with several other iterative linear system solvers. In this section we will present results showing how these methods compare with one another in some matrices arising from real applications. We will be comparing the performance of DQGMRES with BiCGSTAB, TFQMR, GMRES and FGMRES, and will explore some aspects of the flexible preconditioning features of the new algorithm. In our implementation, TFQMR uses slightly more memory than GMRES(8), i.e. the restarted version of GMRES with $m = 8$. Table 1 shows the exact amount of spaces used by each of the five solvers. Note that for both GMRES(m) and DQGMRES(k), we assume $m, k << n$. The time complexity shown in the table is the number of floating-point operations — excluding those for matrix-vector multiplications and the preconditioning operations — required to update the solution averaged over the number of matrix-vector multiplications called (assuming the number of iteration is much larger than m and k). In most real applications, the cost of the matrix-vector multiplications together with the preconditioning operations can be much higher than that related with the other costs, i.e. the operations shown in table 1. For this reason and partly for simplicity, we will use the number of matrix-vector multiplications as the indicator for the time used by each of the iterative solvers that are compared in our discussions later.

We experimented with the four matrices nonsymmetric desribed below, the first three of which are from the Boeing/Harwell sparse matrix collection,

1. **JPWH991** is contributed by J.P. Whelan from Philips LTD. It is a relatively small, only 991 rows and 6027 non-zero elements.

2. **SHERMAN5** is generated by a fully implicit black oil simulator. The matrix size is 3312. It has 20793 non-zero elements.

3. **SHERMAN2** is from a thermal simulation of steam injection on a $6 \times 6 \times 5$ grid. The matrix has 1080 rows and 23094 non-zero elements.

4. **FVOL** is a matrix arising from a volume technique applied to a Navier Stokes equation. It corresponds to the first step in Newton's iteration for solving the Navier-Stokes equation. It has 7160 rows and 112048 non-zero elements.

Roughly speaking the above matrices are listed from 'easiest' to 'hardest' to solve. Thus, the system constructed from the first matrix can be solved by all four methods in a small number of iterations. The others will cause some problems if no preconditioner or a poor preconditioner is applied.

All the right hand-sides are constructed artificially so that the solution to the original linear systems is known. In our tests this known solution is always a vector with all its elements equal to 1. The initial guesses are random vectors, but we always use the same 'pseudo'-random initial guess for the different methods compared on the same linear system.

All experiments have been performed on SPARC workstations. Most of them are on a SPARC-10, a small number of tests are run on SPARC-2 and SPARC-IPX workstations.

We used a number of different preconditioners in our tests, including the standard m-step SOR (Successive Over-Relaxation) and SSOR (Symmetric Successive Over-Relaxation) preconditioners, ILU(0), and ILUT(k, ϵ) [6]. Our version of DQGMRES was implemented with the variable right preconditioning option, to enable us to exploit various types of iterative linear system solvers as preconditioners. Thus, for DQGMRES we also tested using a number of iterative solvers as preconditioners including, BCG, CGNR, etc..

In our experiments, the convergence test used is

$$\|r_i\| \leq rtol \times \|r_0\| + atol \tag{35}$$

where $rtol = 10^{-8}$ and $atol = 10^{-10}$. In all of our experiments, we terminate any iteration when the number of steps exceeds 500.

This first pair of graphs (see figure 1) shows the variation of the residual norm with the number of matrix-vector multiplications. Here the iterative solvers preconditioned only with the diagonal of the matrix. We can see that only the linear system with JPWH991 can be solved in a reasonable number of steps. It is also interesting that for SHERMAN5, TFQMR stagnates in the first 170 steps, while the four GMRES variants stagnate after the first 30 iterations.

Figures 2 and 3 show the convergence process of the iterative solvers with SOR(2) and SSOR as preconditioners. For simplicity, the ω for both SOR(m) and SSOR are chosen to be 1 in our experiments. Only one inner SSOR step is used for preconditioning. The SOR(m) preconditioner has been tested with different m values. What is shown in the figure 2 is the results of using SOR(2). For the linear system with matrix JPWH991, the SOR(2) preconditioning is not as effective as SSOR. However, for SHERMAN5, the two preconditioners have a very similar performance. In figure 2(b), GMRES(8) converges in nearly 490 iterations. In both figure 2(b) and 3(b), DQGMRES(8) did not satisfy our convergence criterion, equation (35), within 500 iterations.

In solving the linear system of JPWH991, SSOR preconditioning reduces the number of iterations required by GMRES(16) and DQGMRES(16) to about 20, which is roughly one third of the number of iterations required if only diagonal preconditioning is applied. For the two 'hard' problems, SOR(m) and SSOR have some effect in helping reducing the residual norm

on one of them, namely FVOL. However, the convergence is still very slow with all iterative methods for both linear systems. When increasing m for SOR(m) and SSOR(m), the number of iterations taken by the iterative solvers decreases for the two 'easy' matrices, JPWH991 and SHERMAN5. However increasing m does not improve the convergence rate for the 'hard' problems SHERMAN2 and FVOL.

Figure 4 shows the results of preconditioning with the Incomplete LU factorization with no fill-in (ILU0). This preconditioner is very effective for SHERMAN2. (see figure 4). It is also slightly more effective on SHERMAN5. In figure 4(b), DQGMRES(8) converges in 308 iterations. For the two other linear systems tested, the effectiveness of the ILU0 preconditioning is comparable with that of SSOR. Next, we test a more sophisticated version of ILU, called ILUT(p, ϵ), discussed in [6]. This is an incomplete LU factorization with a dual dropping strategy, where p is the maximum number of fill-ins allowed per row, ϵ is a relative drop tolerance. In our tests, ILUT(3,10^{-3}) appeared to be sufficiently robust for most of the test matrices (see figure 5), but the result of this preconditioner on the matrix FVOL is still unsatisfactory. After we increase the p to 7 and decrease the ϵ to 10^{-5}, most of the iterative solvers can achieve convergence in no more than 100 iterations (see figure 6).

An appealing feature of the DQGMRES algorithm is that variable preconditioning can be applied at no extra cost. This allows one to choose from a much wider range of preconditioning strategies. The next set of experiments (see figure 7) illustrates the use of some of the common iterative solvers as preconditioners to DQGMRES(16). Such combinations are also commonly referred to as 'inner-outer iterations', where here the outer iteration is DQGMRES(16), and the inner iteration is the iterative preconditioner. In our experiments, each time an inner iterative solvers is called, it will try to solve the linear system given to it using the same convergence test as before (see equation (35)) with *rtol* set to 0.1 and the maximum number of iterations set to 16. In figure 7, the vertical axis of the plot shows the residual normal. The horizontal axis shows the total number of the matrix-vector multiplications used by both the outer iterations (DQGMRES(16)) and the inner iterations. In terms of the number of outer iterations used, this scheme is very competitive with some of the conventional preconditioners such as SOR, SSOR, ILU0, etc. For example when solving JPWH991 using DQGMRES(16) as the outer iteration, GMRES(8) as the inner iteration, only 7 outer iterations are needed to satisfy the convergence criterion (see equation (35)), far fewer than with any one of the conventional preconditioners. However, this may be misleading since the preconditioner in the inner-outer iteration scheme is more expensive than some of the simpler preconditioners, such as SOR(2), SSOR. In our experiments the inner iterations may use up to 16 matrix-vector multiplications. In terms of floating-point operations, one SSOR preconditioning operation costs about as much as two matrix-vector multiplications.

We observe that the inner-outer iteration preconditioning scheme is robust, although it is also expensive compared with conventional preconditioners such as SSOR. One possibility for reducing the number of matrix-vector multiplications is to precondition the inner iterations. We have tested this idea, and some of the results are shown in figure 8. Similarly to figure 7, the horizontal axis is the total number of matrix-vector multiplications. Comparing figure 8 with figure 7, we can see that applying the preconditioner to the inner iteration reduces the total number of matrix-vector multiplications. Comparing figure 8 with figure 2, we notice that this inner-outer iteration scheme with SSOR as preconditioner to the inner iterations requires more matrix-vector multiplications than simply using DQGMRES(16) with SSOR as preconditioner. However, in some other cases, we have observed that this inner-outer iteration scheme can

be as effective as DQGMRES(16) with a more conventional preconditioner. We should point out that these 'inner-outer' iteration schemes bear some similarity with the idea of polynomial preconditioning. The only difference is that here the iteration polynomial changes at each outer step.

To further demonstrate the advantages of the flexibility feature provided by DQGMRES, we experimented with a combination of different types of preconditioners, specifically, we run some experiments consisting of alternating between different preconditioners at each outer loop. Figure 9 shows the results of alternating between SSOR and one of the iterative solvers when solving SHERMAN5. Comparing this with the results of the inner-outer schemes, this option uses fewer matrix-vector multiplications. Comparing it with using only SSOR as a preconditioner, this scheme uses fewer outer (DQGMRES(16)) iterations. For example, when alternating between GMRES(8) and SSOR preconditioners, DQGMRES(16) used 31 iterations, while when using only SSOR as the preconditioner to DQGMRES(16), 65 iterations were required. However, in 31 iterations, the alternating preconditioner scheme used 271 matrix-vector multiplications and called SSOR 15 times, while in its 65 iterations, the simpler preconditioning scheme only used 65 matrix-vector multiplications and 65 SSOR calls. On sequential computers one SSOR step is roughly equivalent to one matrix-vector multiplication, so the alternating preconditioner approach is not as effective. However, the situation may be not be the same for parallel computers.

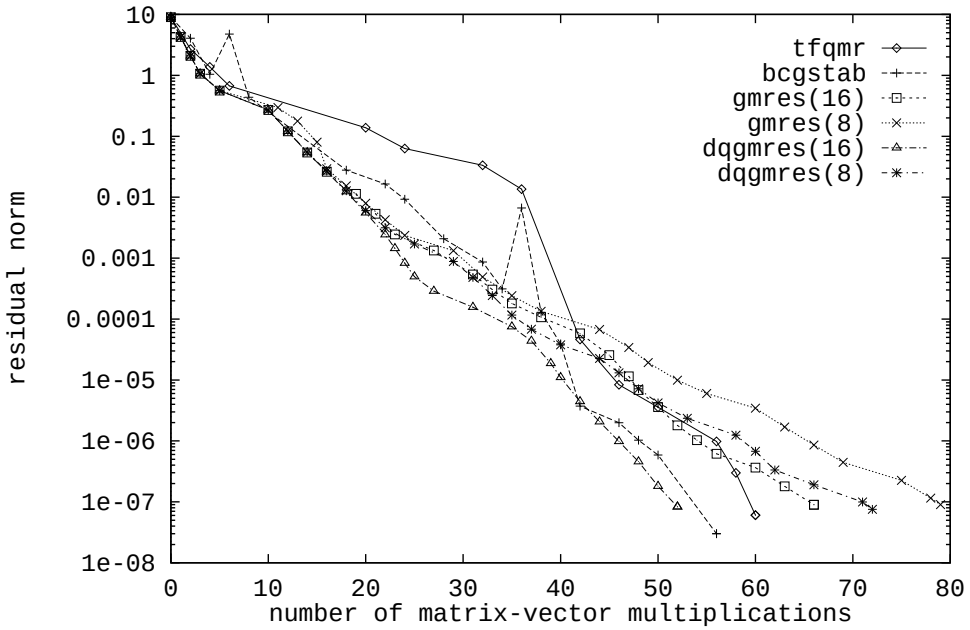
In all the inner-outer iterations tests, we also experimented with using FGMRES as the outer iteration. In figure 7, we plotted the results of using FGMRES as well. The difference between using DQGMRES and FGMRES is small. This was to be expected, since the two methods are closely related. Clearly, the first 16 iterations of DQGMRES(16) and FGMRES(16) should be exactly the same, and the number of the outer iterations in all our experiments are less than or close to 16. Even though the differences are generally small, DQGMRES consistently outperforms FGMRES when the number of outer iterations required for convergence is high.

6 Conclusion

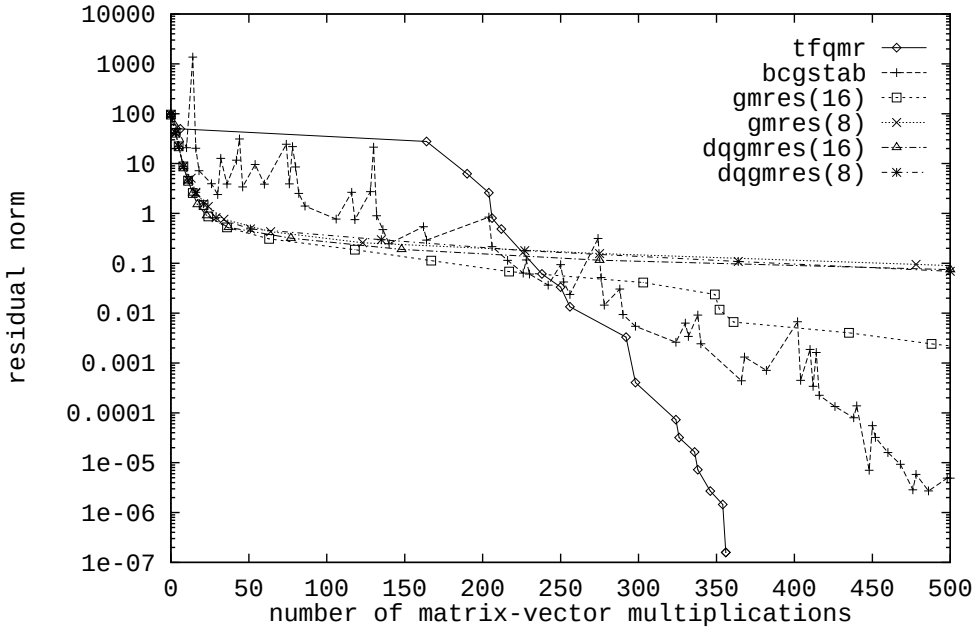
The DQGMRES algorithm compares well with the best iterative techniques available for solving linear systems. One exception is that when the number k against which a given Arnoldi vector is orthogonalized is too small, DQGMRES tends to perform poorly. One advantage of DQGMRES over GMRES, TFQMR and BiCGSTAB, is that it is 'flexible'. It allows the preconditioner to vary at each step – without requiring extra storage as is the case for GMRES [7]. Generally speaking the algorithm often does better than GMRES for the same k, the number of vectors that are mutually orthogonal in the Arnoldi sequence at any given step. However for the same k, DQGMRES requires twice as much storage as GMRES, but the same storage as FGMRES.

Acknowledgement.

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

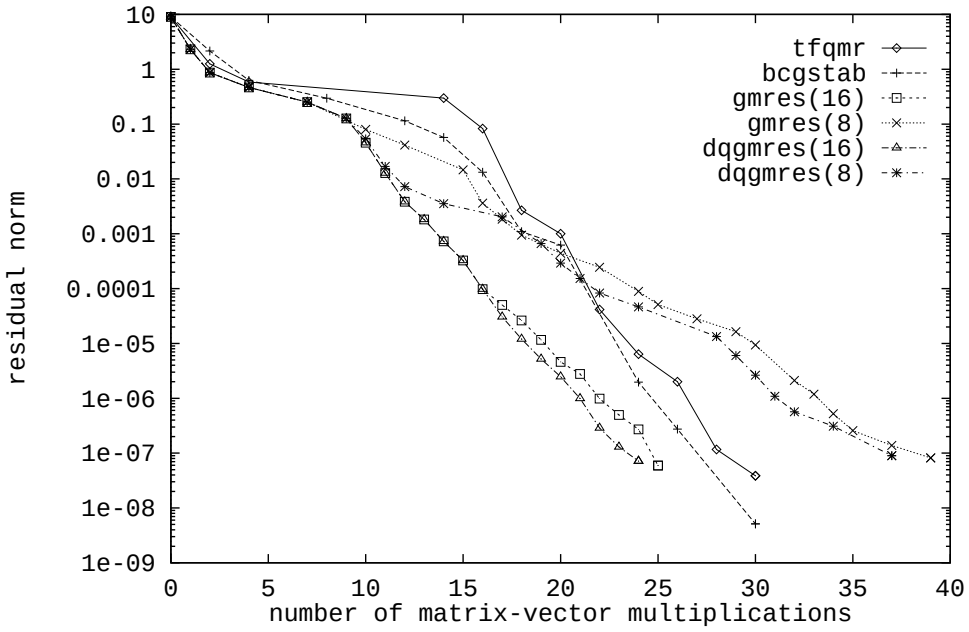


(a) JPWH991

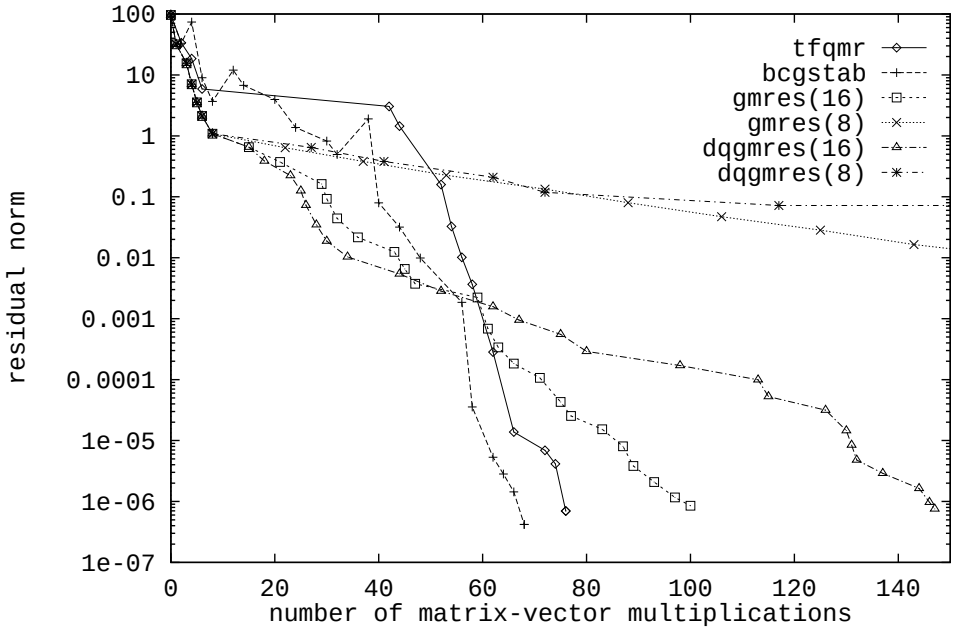


(b) SHERMAN5

Figure 1: Residual norms of the first two linear systems solved with only diagonal preconditioning.

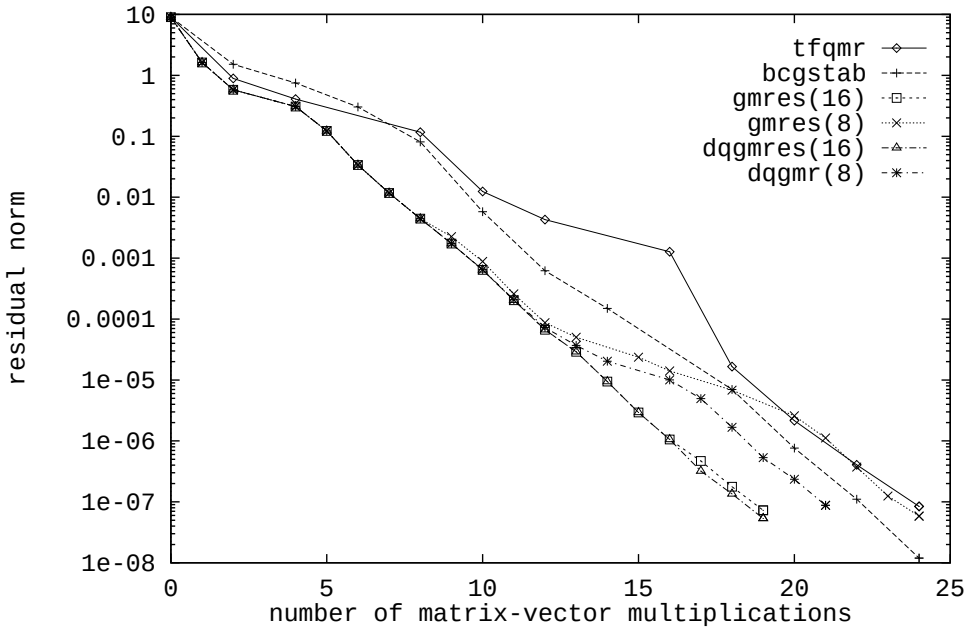


(a) JPWH991

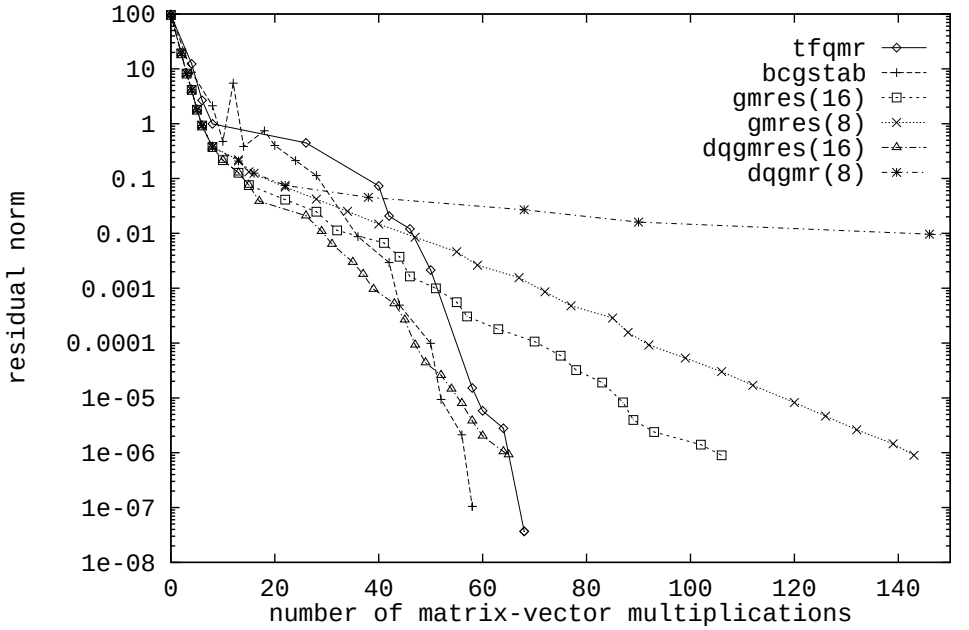


(b) SHERMAN5

Figure 2: Residual norms of two linear systems solved with SOR(2) preconditioning.

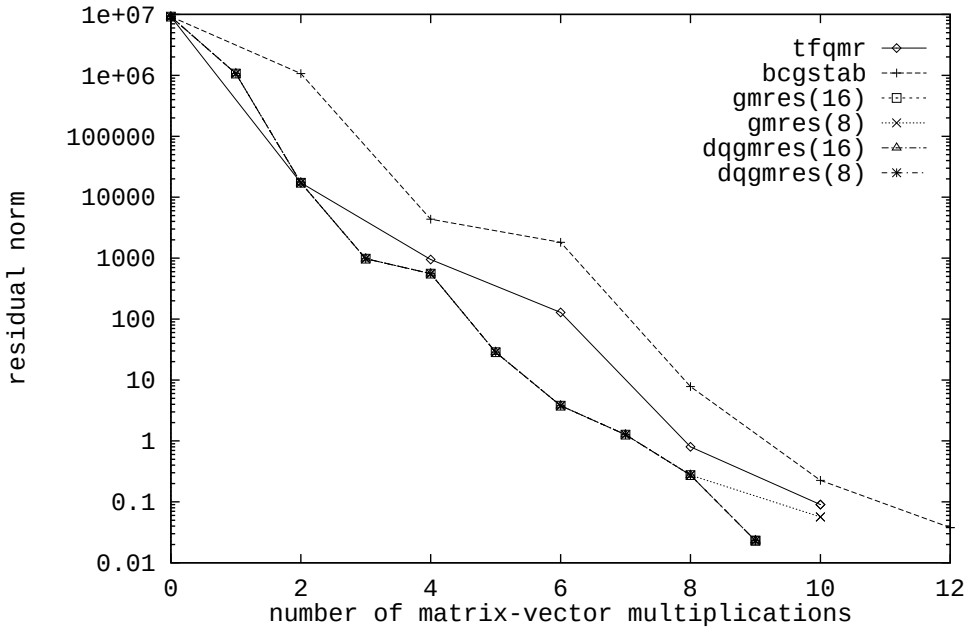


(a) JPWH991

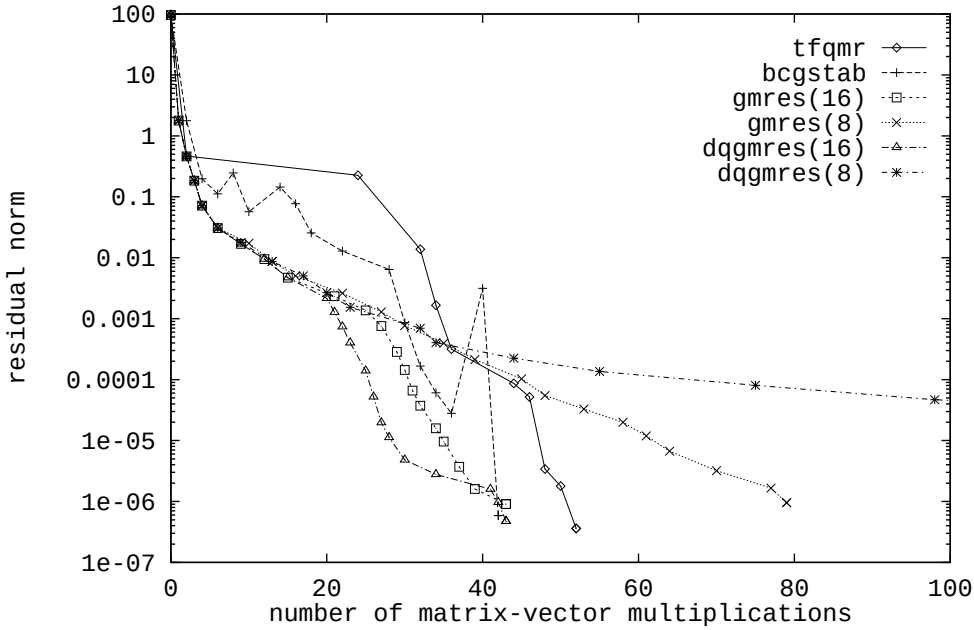


(b) SHERMAN5

Figure 3: Residual norms of two linear systems solved with SSOR preconditioning.

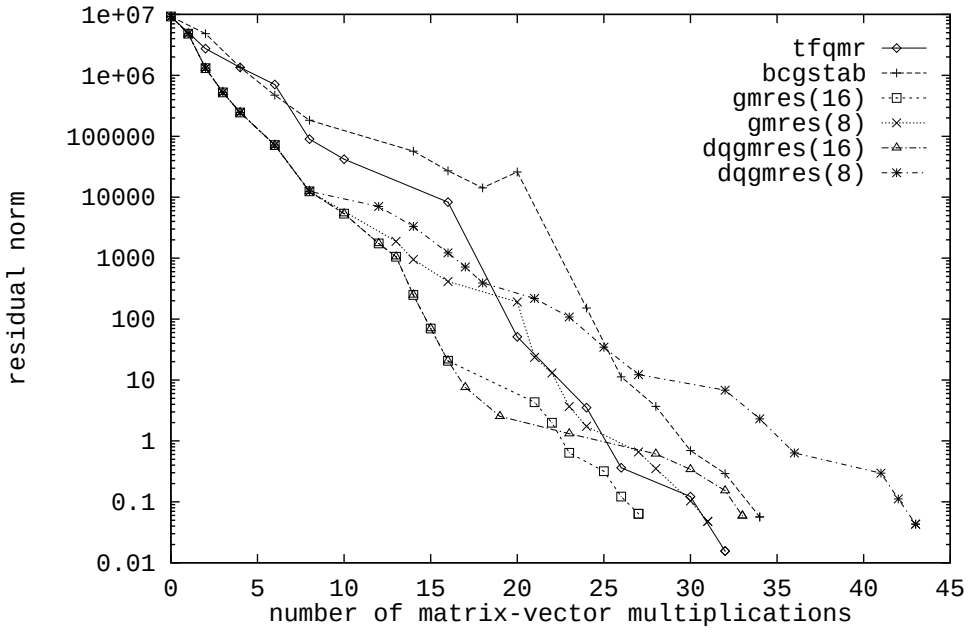


(a) SHERMAN2

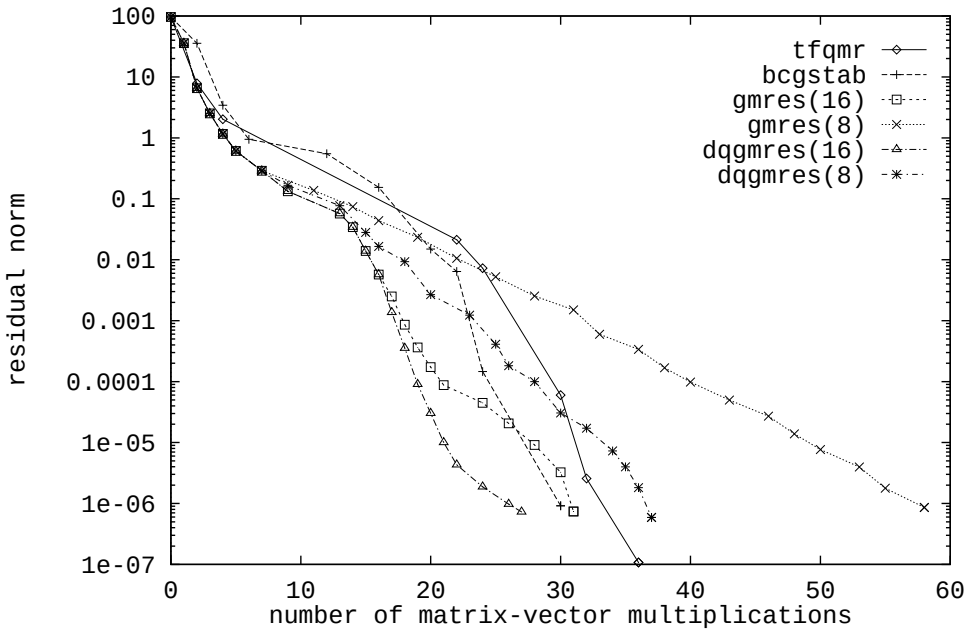


(b) SHERMAN5

Figure 4: Residual norms of two linear systems solved with ILU0 preconditioning.

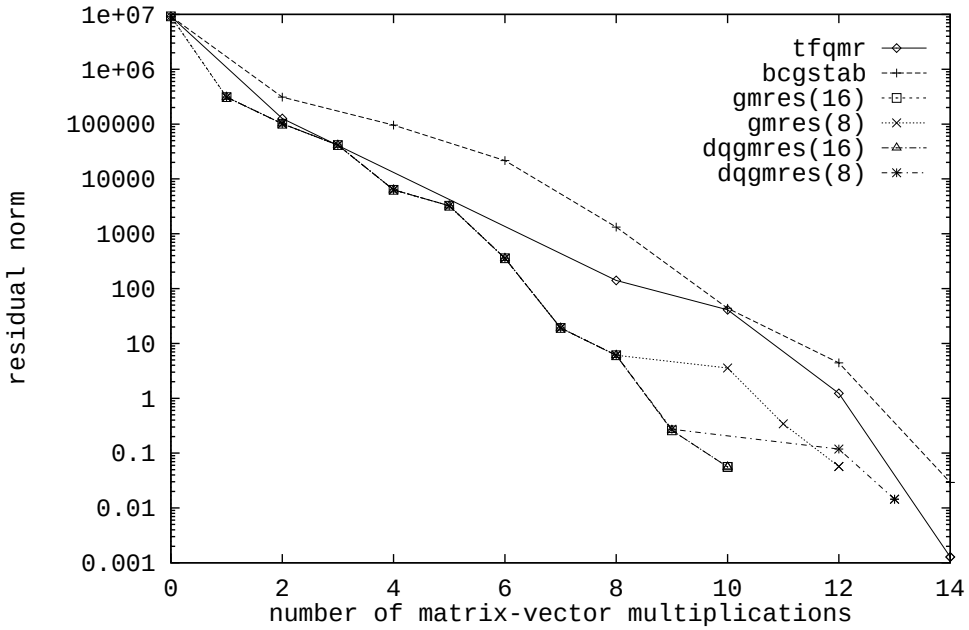


(a) SHERMAN2

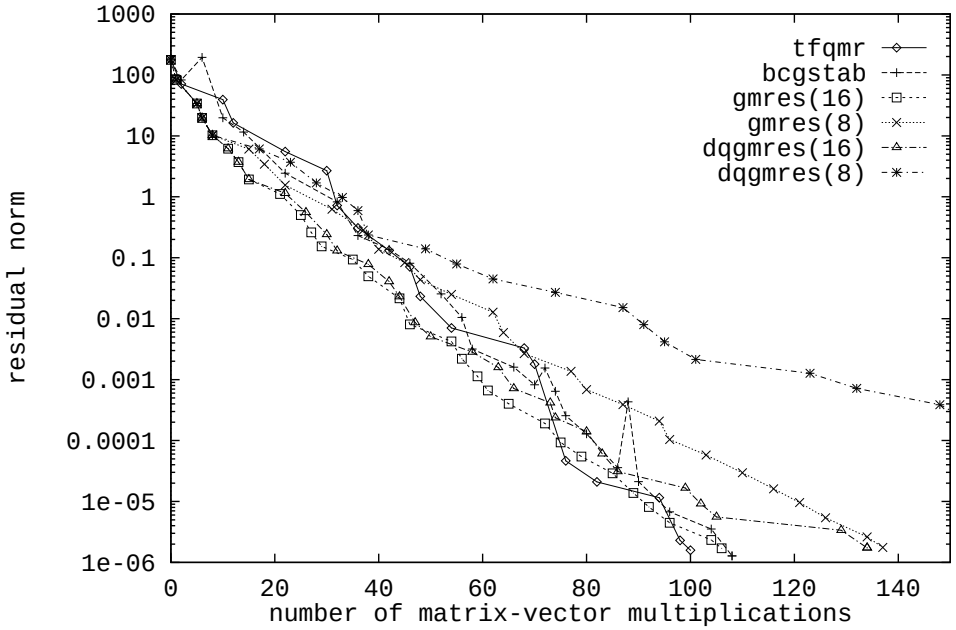


(b) SHERMAN5

Figure 5: Residual norms of two linear systems solved with ILUT(3,10^{-3}) preconditioning.

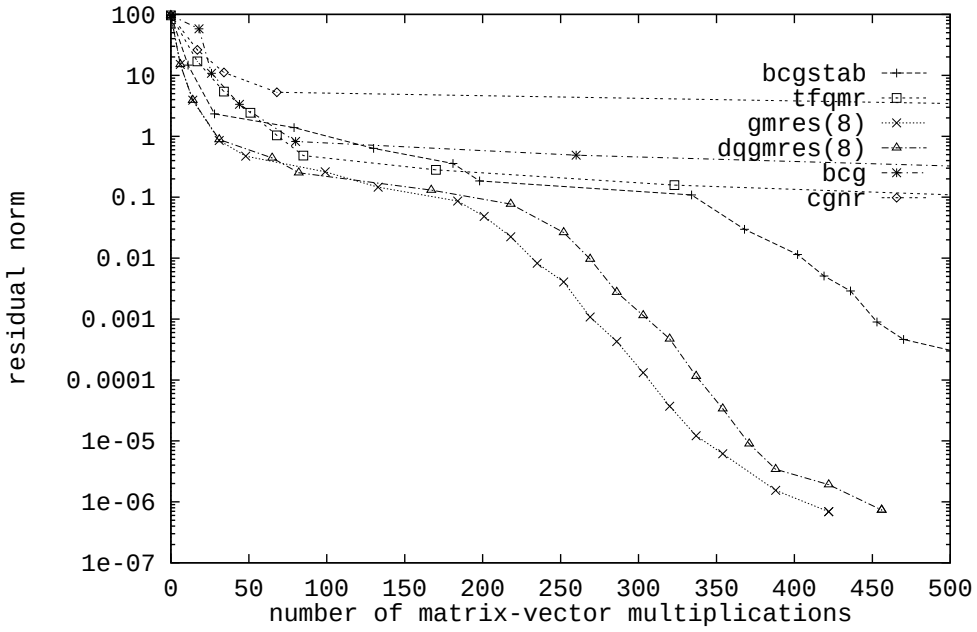


(a) SHERMAN2

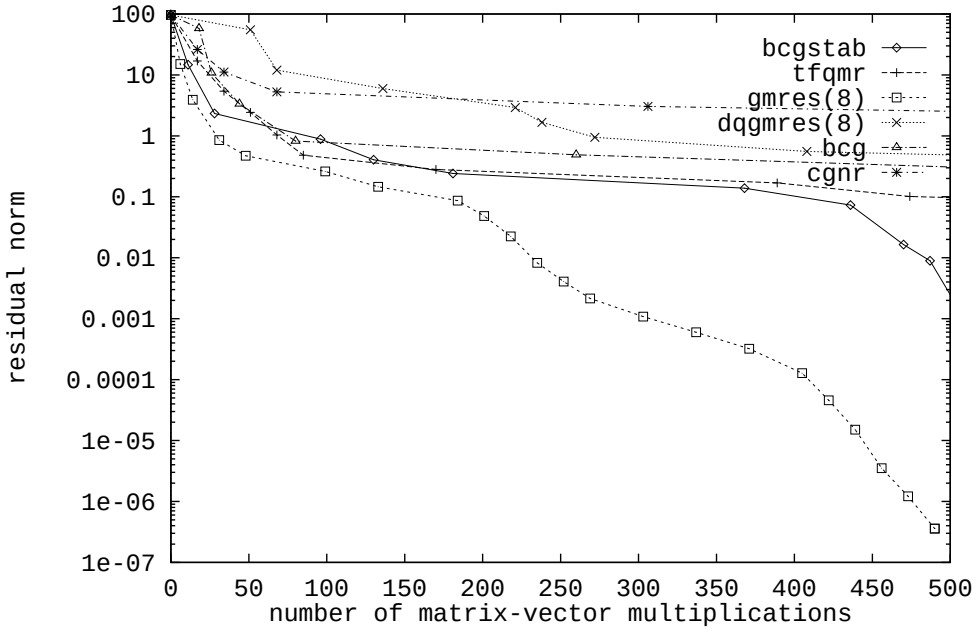


(b) FVOL

Figure 6: Residual norms of two linear systems solved with ILUT(7,10^{-5}) preconditioning.



(a) DQGMRES



(b) FGMRES

Figure 7: Residual norms of SHERMAN5 solved with the inner-outer iteration schemes.

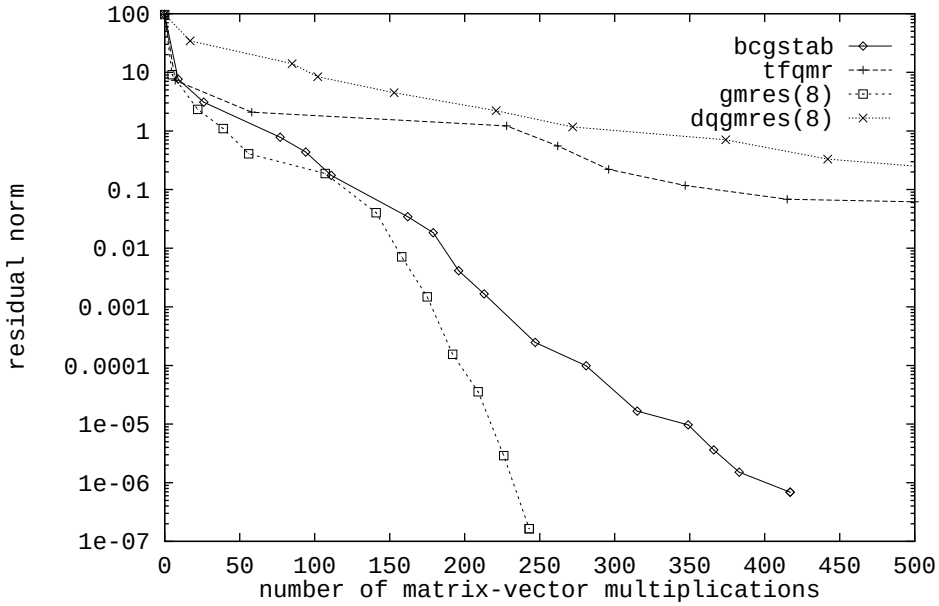


Figure 8: SHERMAN5 solved under the inner-outer iteration schemes with SSOR preconditioning on the inner iterations.

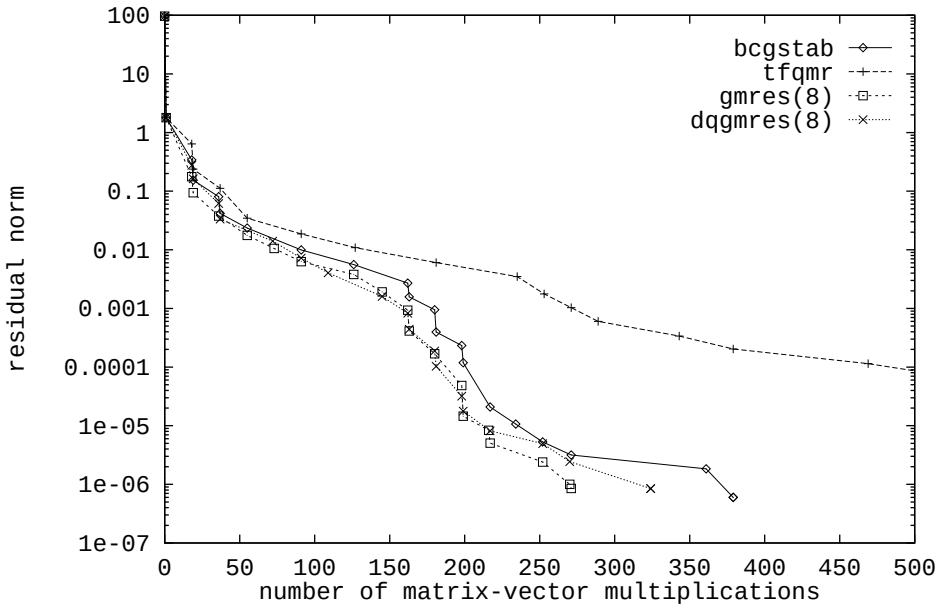


Figure 9: SHERMAN5 solved by the inner-outer iteration schemes with preconditioner alternating between SSOR and an iterative solver.

References

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

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

[3] 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, MIT, Appliexd Mathematics, Cambridge, Massachusetts 02139, 1991.

[4] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation*, 37:105–126, 1981.

[5] 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.

[6] Y. Saad. ILUT: a dual threshold incomplete ILU factorization. Technical Report 92-38, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, 1992.

[7] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Stat. Comput.*, 14:461–469, 1993.

[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] H. A. van der Vorst and C. Vuik. GMRESR: a family of nested gmres methods. Technical Report 91-80, Delft University of Technology, Mathematics and Informatics, Delft, The Netherlands, 1991.