Notes on GMRES Algorithm Organization

Richard J. Hanson

Center for High Performance Software Research, Rice University, Houston, Texas, 77251–1892 USA

David R. Kincaid

Department of Computer Sciences, University of Texas at Austin, Austin, Texas, 78712–0233 USA

Technical Report TR-05-05

Department of Computer Sciences, University of Texas at Austin, March 05, 2005

Abstract

The Generalized Minimum Residual (GMRES) iterative method and variations of it are frequently used for solving systems of linear equations of the form $Ax = b$, where A is a large sparse nonsingular nonsymmetric matrix. We discuss ways to reorganize the algorithm to improve its efficiency.

Key words: Generalized Minimum Residual (GMRES) iterative method, Preconditioned GMRES(m) Algorithm, solving large sparse systems of linear equations, GMRES algorithm reorganization and Matlab code,

1 Introduction

The well-known Generalized Minimum Residual (GMRES) iterative method is often used to solve a system of linear equations

$$Ax = b$$

Email addresses: hanson.rj@worldnet.att.net (Richard J. Hanson), kincaid@cs.utexas.edu (David R. Kincaid).

where A is a large sparse nonsingular nonsymmetric $n \times n$ matrix. The left-right preconditioned system is

$$(M_L^{-1} A M_R^{-1})(M_R x) = M_L^{-1} b$$

where matrices M_L and M_R are nonsingular matrices.

2 Review GMRES.

In this section, we present a brief review of the GMRES procedure for the case $m = 3$. Knowledgeable readers should immediately skip to the next section. In the first phase, the Arnoldi process is used to generate an orthonormal basis set $\{w^{(1)}, w^{(2)}, w^{(3)}\}$ for the Krylov subspace $\mathcal{K}_3(r^{(0)}, A)$ where $r^{(0)} = b - Az^{(0)}$ is the residual vector for the initial value $z^{(0)}$. In the second phase, we form a linear combination

$$z^{(3)} = z^{(0)} + c_1^{(3)} w^{(1)} + c_2^{(3)} w^{(2)} + c_3^{(3)} w^{(3)} = z^{(0)} + W_3 c^{(3)} \tag{1}$$

using matrix and vector notation

$$W_3 = [w^{(1)}, w^{(2)}, w^{(3)}]_{n \times 3}, \qquad c^{(3)} = [c_1^{(3)}, c_2^{(3)}, c_3^{(3)}]_3^T$$

From the Arnoldi process, we have

$$AW_3 = W_4 H_3, \qquad W_4^T W_4 = I \tag{2}$$

and

$$H_3 = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ \sigma_1 & h_{2,2} & h_{2,3} \\ 0 & \sigma_2 & h_{3,3} \\ 0 & 0 & \sigma_3 \end{bmatrix}_{4 \times 3} \tag{3}$$

where $\sigma_0 = ||b - Az^{(0)}||_2^2$ and $\sigma_i = ||w^{(i)}||_2^2$. In the solution phase, we need the least squares solution of

$$H_3 c^{(3)} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ \sigma_1 & h_{2,2} & h_{2,3} \\ 0 & \sigma_2 & h_{3,3} \\ 0 & 0 & \sigma_3 \end{bmatrix} \begin{bmatrix} c_1^{(3)} \\ c_2^{(3)} \\ c_3^{(3)} \end{bmatrix} = \begin{bmatrix} \sigma_0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = q$$

To successively zero out the values of σ_1, σ_2, and σ_3, We use Givens transformation matrices Q_i of the form

$$Q_1 = \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Q_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2 & s_2 & 0 \\ 0 & -s_2 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Q_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c_3 & s_3 \\ 0 & 0 & -s_3 & c_3 \end{bmatrix},$$

where

$$\alpha_i = [\widetilde{h}_{i,i}^2 + \sigma_i^2]^{1/2}, \quad c_i = \widetilde{h}_{i,i}/\alpha_i, \quad s_i = \sigma_i/\alpha_i$$

for $i = 1, 2, 3$. When the Givens matrix is constructed so that when Q_i is applied to matrix H_m it modifies the diagonal entry $\widetilde{h}_{i,i} \leftarrow c_i\widetilde{h}_{ii} + s_i\sigma_i$ and zeros-out the σ_i entry. It also modifies the rows i and $i+1$ from column $i+1$ through n; namely, $\widetilde{h}_{i,j} \leftarrow c_i\widetilde{h}_{i,j} + s_i\sigma_i$ and $\widetilde{h}_{i+1,j} \leftarrow -s_i\widetilde{h}_{i,j} + c_i\widetilde{h}_{i+1,j}$ for $j = i+1, \ldots, n$. Here $\widetilde{h}_{i,j}$ are the appropriately updated elements of the H_m matrix. In our Matlab code, we use the `function rotg.m` to determine the elements s_i and c_i in the Givens rotation matrix.

Now we apply each Givens transformations to both sides of the linear system (3) and obtain

$$QH_3c_3^{(3)} = \begin{bmatrix} \widetilde{h}_{1,1} & \widetilde{h}_{1,2} & \widetilde{h}_{1,3} \\ 0 & \widetilde{h}_{2,2} & \widetilde{h}_{2,3} \\ 0 & 0 & \widetilde{h}_{3,3} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_1^{(3)} \\ c_2^{(3)} \\ c_3^{(3)} \end{bmatrix} = Q_3Q_2Q_1q = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{4}$$

where $Q = Q_3Q_2Q_1$. Consequently, we obtain

$$R_3c_3^{(3)} = QH_3c_3^{(3)} = Qq = y$$

Finally using the first three equations in (4), we can solve this upper triangular system for the coefficients $\widetilde{c}_1^{(3)}, \widetilde{c}_2^{(3)}$, and $\widetilde{c}_3^{(3)}$.

Let $r^{(0)} = W_4q$ where $q = \sigma_0[1, 0, 0, 0]_4^T$ and $\sigma_0 = ||r^{(0)}||_2^2$. Evidently from (1) and (2), we have

$$r^{(3)} = r^{(0)} - AW_3c^{(3)} = r^{(0)} - W_4H_3c^{(3)} = W_4(q - H_3c^{(3)})$$

We have

$$||r^{(3)}||_2^2 = ||W_4(q - H_3c^{(3)})||_2^2 = ||q - H_3c^{(3)}||_2^2$$

using the second equation in (2). Now we have

$$\min ||r^{(3)}||_2^2 = \min ||q - H_3c^{(3)}||_2^2$$

$$= \min ||Q^T(Qq - R_3 c^{(3)})||_2^2 = \min ||Qq - R_3 c^{(3)}||_2^2$$

Here H_3 is factored as $H_3 = Q^T R_3$ where Q is an 4×4 orthogonal matrix ($QQ^T = I$) that is a product of three Givens rotation matrices and R_3 is an 4×3 upper triangular matrix. Let $Qq = y = [y_1, y_2, y_3, y_4]^T$. We can write

$$||Qq - R_3 c^{(3)}||_2^2 = ||\widetilde{y} - \widetilde{R}_3 \widetilde{c}^{(3)}||_2^2 + |y_4|^2$$

where $\widetilde{y}$ and $\widetilde{R}_3$ are the first 3 rows of this vector and matrix. (Remember the last row of R_3 is all zeros.) Choosing $\widetilde{c}^{(3)}$ such that

$$\widetilde{R}_3 \widetilde{c}^{(3)} = \widetilde{y}$$

then $\min ||r^{(3)}||_2^2 = |y_4|^2$. The last element of $y = Qq$ is $y_4 = -\sigma_0 s_1 s_2 s_3$. So we obtain $||r^{(k)}||_2 = |y_4|$ for each iteration without having to compute the residuals. The value of $|y_4|$ gives an indication of whether there is an improvement in the approximation or *stagnation* has occurred. The latter is the case when all the $s_i = 1$ to within the floating-point precision being used.

3 Preconditioned GMRES(m) Algorithm.

The originally published GMRES algorithm found in Saad and Schultz (1986) now appears in many others publications. (See, for example, Golub and van Loan (1996), Greenbaum (1997), or Saad (2003).) We do not include the statement of the traditional algorithm because there are many slightly different versions of the GMRES algorithm (preconditioned and unpreconditioned) in various papers and books. Rather than stating the standard GMRES algorithm, we present it in a more streamlined fashion.

Generalizations and modifications of the GMRES method involving a weighting norm $||x||_C^2 = \langle x, x \rangle_C = \langle Cx, x \rangle = x^T C x$, where C is positive definite symmetric, are given in Chen (1997); Chen, Kincaid, and Young (1999); Kincaid, Young and Chen (2003). We could have easily done that here but it would made the flow of the algorithm more complicated and less likely to be understood.

An important issue with the standard statement of the GMRES algorithm is the occurrence of two applications of the $n \times n$ matrix A times a vector (or a linear operator can be used instead of a matrix). With a small amount of reorganization, the algorithm can be written so that the matrix-vector product is used just once as shown below. The reorganized algorithm uses a single scratch array of size n.

An improved and reorganized version of the GMRES Algorithm is as follows.

Preconditioned GMRES(m) Algorithm

(1) $x \leftarrow z^{(0)}$; (an initial solution approximation)
(2) **for** $j = 0$ **to** m **do**
(3) $x \leftarrow Ax$; (matrix-vector product/overwrite))
(4) **if** $(j = 0)$ $x \leftarrow b - x$;
(5) $[x \leftarrow M_L^{-1}x$; (left preconditioner, solve/overwrite)]
(6) **for** $i = 1$ **to** j **do**
(7) $h_{i,j} \leftarrow t \leftarrow \langle x, w^{(i)} \rangle$; $x \leftarrow x - tw^{(i)}$;
(8) **end for**
(9) $\sigma_j \leftarrow ||x||$;
(10) **if** $(|\sigma_j| \leq \delta$ **or** $j = m)$ **exit loop** j;
(11) $x \leftarrow \sigma_j^{-1}x$; $w^{(j+1)} \leftarrow x$
(12) $[x \leftarrow M_R^{-1}x$; (right preconditioner, solve/overwrite)]
(13) **end for**

In the algorithm above, note that the outer loop starts with the index value $j = 0$. The advantage of one instead of two occurrences of the matrix-vector multiplication is a simplification of the interface to the user's code. As a consequence, there is a test in the outer loop of the algorithm (Step 4) where the inner loop is skipped when $j = 0$ and, in this case, the first normalized basis vector $w^{(1)} = \sigma_0^{-1}(b - Az^{(0)})$ is set in Step 11. Step 7, defines the upper triangular matrix $H = \{h_{i,j}\}$ with the usual row and column values. It is worth mentioning that the test (Step 10) is for $\sigma_j = 0$ and involves comparing against a positive tolerance δ. The loop index j starts at the value $j = 0$ and defines the columns of the matrix $W = [w^{(1)}, w^{(2)}, \ldots, w^{(\widehat{j})}]$, where $\widehat{j} \leq m$ is the value of j at the exit in Step 10. With the test to exit the loop with $j = m$, there is no need to store an unused column; namely, $w^{(m+1)}$.

In using GMRES, the interface to the user's code is one of either forward communication, reverse communication, or embedding the matrix (or an operator) in the innards of the loop (the latter is not recommended). It is almost always necessary to use preconditioners, in which case one considers a related linear system as given in Golub and van Loan (1996, pp. 549–550) Thus, one iterates with a matrix $G = M_L^{-1}AM_R^{-1}$ (or an operator). Note that the use of preconditioning is indicated above in brackets (Steps 5 and 12) with related solve and overwrite steps. In these steps for large systems, we would solve $M_L y = x$ for y and overwrite $x \leftarrow y$ as well as solve $M_R y = x$ for y and overwrite $x \leftarrow y$. It is with the use of preconditioners that we find the reorganization most appealing.

Should stagnation occur with $|y_{m+1}| = \sigma_0$, we suggest that one combine the following options: restart the iterative algorithm using the current approximate solution $z^{(m)}$ as the initial value $z^{(0)}$ with a larger value of m involving more storage or use preconditioners that achieve better conditioning of $G = M_L^{-1}AM_R^{-1}$.

We want to emphasize that the rationale for the reorganized statement of the algorithm is that when the GMRES algorithm is interfaced to the matrix (or an operator) and preconditioner, there is only one place for each application of it. However, the right preconditioner must be applied again at the update step in the solution phase.

4 Solution Phase.

In the solution phase of this algorithm, we compute $c^{(m)}$ the minimizer of $||H_m c^{(m)} - q_m||$ and compute $z^{(m)} = z^{(0)} + W_m c^{(m)}$. Here we define the $(m+1) \times m$ Hessenberg matrix $H_m = \{h_{i,j}\}_{(m+1)\times m}$ and $(m+1)$ vectors $W_m = [w^{(1)}, w^{(2)}, \ldots, w^{(m)}]$. To solve the least squares problem $\min ||H_m c^{(m)} - q_m||$, one transforms the Hessenberg matrix into upper triangular form by using Givens plane rotations. Multiply the Hessenberg matrix H_m and the right hand side q_m by a sequence of rotation matrices Q_i that eliminate the subdiagonal elements σ_i one at a time. We obtain $R_m = QH_m$ and $y_m = Qq_m$ where $Q = Q_m \cdots Q_2 Q_1$. Since Q is unitary $\min ||H_m c^{(m)} - q_m|| = \min ||R_m c^{(m)} - y_m||$. The solution of this least squares problem is obtained by solving the upper triangular system resulting from deleting the last row of zeros in the matrix R_m and the right hand side y_m.

When solving the approximate resulting system with plane rotations, an important point is to have unit strides in the inner loop where the transformations are to be applied. This has a large impact on the efficiency for that part of the algorithm. Our Matlab code uses packed upper triangular storage with unit strides across rows for storing H. The subdiagonals of H are σ_i for $i = 1, 2, \ldots, m$.

The matrix $H = \{h_{i,j}\}_{(m+1)\times m}$ is upper Hessenberg and the subdiagonals are the values σ_j for $j = 1, 2, \ldots, m$. The value of $j = \widehat{j} \leq m$ is when the exit occurs at Step 10 and it defines the dimension of the least squares system that is computed in the solution phase. In this discussion, we assume the $j = m$. Table 1 uses the fact that only the upper triangular part of H occupies space. For $j = 0$, the updated solution is $z^{(0)}$. It is worth pointing out that this requires a nonstandard triangular storage array for H in row-major form. This way the solve step can always use unit stride in the inner loop while solving the least squares system. This organization is independent of the programming language used to implement the algorithm. See the Matlab code and Sections 5–6 for the implementation details.

The least squares system $R_m c^{(m)} = y_m$ that must be solved for the solution update is computed by using plane rotations in the form of Givens transformations matrices. Then a back solve step is performed on the resulting upper

triangular matrix. This description of the solution phase of the algorithm is a refinement of Saad and Schultz (1986, Section 3.2).

Solution Phase Preconditioned GMRES(m) Algorithm

(1) $y \leftarrow [\sigma_0, 0, 0, \ldots, 0]^T_{m+1}$
(2) **for** $i = 1$ to m **do**
(3) Construct Givens matrices Q_i with principal submatrix $\begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix}$
(4) Apply Q_i to H, (rows i and $i+1$ of columns $i+1$ through m)
so last two nonzero entries of column vector i become
$\begin{bmatrix} \widetilde{h}_{i,i} \leftarrow c_i h_{ii} + s_i \sigma_i \\ \sigma_i \leftarrow 0 \end{bmatrix}$
(5) Apply Q_i to y (entries y_i and y_{i+1})
(6) **end for**
(7) $y_m \leftarrow h^{-1}_{m,m} y_m$ (begin back solve)
(8) **for** $i = m-1$ **to** 1 **step** -1 **do**
(9) $y_i \leftarrow h^{-1}_{i,i} \Big(y_i - \sum_{k=i+1}^{j} h_{i,k} y_k \Big)$
(10) **end for**
(11) $x \leftarrow Wy$ (matrix-vector product)
(12) $[x \leftarrow M_R^{-1} x$ (right preconditioner/overwrite)]
(13) $z^{(m)} \leftarrow z^{(0)} + x$ (update approximation)

In Steps 4 and 9, unit strides in the storage of rows assure efficiency of memory access. Step 5 requires two multiplies instead of the usual four because $y_{i+1} = 0$. Loop i (Steps 7-10) back solves the upper triangular system.

5 Testing Code.

We have written Matlab code to illustrate this organization of the GMRES algorithm together with two test programs and the Givens rotation matrix routine:

- `test_x.m`
- `test_y.m`
- `gmres.m`
- `rotg.m`
- `summary.m`

Two Matlab programs `test_x.m` and `test_y.m` exercise routine gmres.m using the Given rotation routine `rotg.m` and resulting plots are produced by routine

`summary.m`. All of these have been placed on a Web site so others can download and run them: `www.cs.utexas.edu/users/kincaid/GMRES`

In the first Matlab sample code `test_x`, we ask the user to input the size of the system n. An $n \times n$ random matrix A is created with a slight diagonal dominance. Next we set the n component vector $x = (1, 1, \ldots, 1)$ and generate the corresponding right hand side vector $b = Ax$ using A and x. Then we request the number of k subdiagonals and k superdiagonals used for computing the ILU factors in the preconditioners. Finally, we ask the user to enter the number m of GMRES iterations to be used in the compute intensive phase of the GMRES(m) algorithm with preconditioner. The initial approximation $z^{(0)}$ used is a random vector and the tolerance is set to $\delta = 10^{-4}$. We stop the iteration if the residual is small enough or we have come to the maximum number of iterations. At the end, summary plots are produced with a bar graph of the solution vector and the logarithmic graph of the `sigma`(i) values, which are an indication of the convergence. The sample data $n = 30$, $k = 15$, $m = 10$ obtains the solution with 10^{-4} accuracy in 24 GMRES(10) iterations.

Users may notice that by selecting different values for k and m, the behavior of the code varies. For example, $n = 30$, $k = 15$, $m = 15$ converges in 4 GMRES(m) iterations and $n = 30$, $k = 5$, $m = 20$ converges in 5 GMRES(m) iterations but $n = 30$, $k = 15$, $m = 5$ does not converge. The number of GMRES(m) iterations is always m but there may be an indefinite number of restarted groups to obtain convergence. For example, 12 groups of GMRES(10) implies that there are 12×10 evaluations of Ax, 12 solution steps, and so on. In general, one may obtain more rapid convergence with large values m but this requires more storage. Also, there is the risk of not converging with too small a value of m.

By zeroing out entries in A, we create a matrix B from A having only k subdiagonals and k subdiagonal. We compute the row pivoted form of the LU factorization of $PB = LU$ where P is a permutation matrix, L is the unit lower triangular matrix, and U is the upper triangular matrix. These factors are used as the preconditioners. The idea here is that B approximates A and the factorization of B is effective and relatively inexpensive to compute. Thus, we have $A \approx B = P^T LU$. In terms of the GRMES algorithm, we use $M_L = P^T L$ and $M_R = U$.

For our second Matlab sample code `test_y`, we use finite differences and solve the following elliptic partial differential equation on the unit square

$$\begin{cases} u_{xx} + u_{yy} + 1 = 0, & (x, y) \in R = [0, 1] \times [0, 1] \\ u(x, y) = 0, & (x, y) \in \text{Boundary}(R) \end{cases}$$

Array	x	$W = [w^{(i)}]$	σ	$H = \{h_{i,j}\}$	$z^{(0)}$
Size	n	mn	$m+1$	$m(m+1)/2$	n

Table 1
Storage requirements.

We generate nonzero elements of the sparse linear system and use the GMRES code Matlab to solve it. The user is asked to enter the number n of inner grid points (in x or y direction) and the number m of GMRES iterations. To reduce the amount of time spent building the sparse matrix from the standard finite differences, the code is written to generate the nonzero coefficients in three groupings. It traverses first the corners, then the edges, and finally the main part of the code does the equations whose entries do not touch the boundary. This results in the generation of large systems more quickly because the movement of large chunks of data is reduced. The preconditioner is derived from the LU factorization of the principle tridiagonal matrix. (Also, it could have been done with the Cholesky decomposition.) Summary plots are produced at the end of the solution phase for both the solution surface and for the logarithm of `sigma`(i). We call this example solving for *Gabriel's pillow* due to the shape of the two-dimensional plot of the solution surface. For example, the sample date $n = 32$, and $m = 16$ solves the $n^2 \times n^2$ system with 10^{-4} accuracy in 6 GMRES(16) iterations.

6 Storage Requirements.

The amount of storage needed for the algorithm should be mentioned. Note that we have eliminated a scratch array of size n. One could make a case for having an additional scratch array. This is particularly important when computing the operator evaluation and for the preconditioning steps. But this extra array is not needed and in the context of solving large partial differential equations this saving may be significant and appreciated.

In the solution phase of the Matlab code, the subdiagonal terms of H are $\sigma_1, \sigma_2, \sigma_3, \ldots$, and they are stored in the array `sigma`. After the solution phase ends, we check for decreasing values of the `sigma` entries , which implies convergence. We use the stagnation stopping test is $|\sigma_0^{(k)} - |y_{m+1}|| < \texttt{eps}^{1/2}\, \sigma_0^{(0)}$ with $\sigma_0^{(k)}$ the current value, $\sigma_0^{(0)}$ the initial value, and `eps` the Matlab constant for the spacing of the floating-point numbers on the computer being used.

References

J-Y. Chen. Iterative Solution of Large Sparse Nonsymmetric Linear Systems. Report CNA–285 (Ph. D. thesis, Dept. of Mathematics), Center for Numerical Analysis, University of Texas at Austin, 1997.

J-Y. Chen, D. R. Kincaid, D. M. Young. Generalizations and modifications of the GMRES iterative method, *Numerical Algorithms* 21 (1999) 119–146.

A. Greenbaum. Iterative Methods for Solving Linear Systems. SIAM, Philadelphia, 1997.

G. Golub and C. van Loan. Matrix Computations, 3rd Edition. John Hopkins University Press, London, 1996.

R. J. Hanson and D. R. Kincaid. Notes on GMRES Algorithm Organization, Technical Report TR-05-05, Computer Sciences Department, University of Texas at Austin, March 2005.

D. R. Kincaid, D. M. Young, and J-Y. Chen. Variations of the GMRES iterative method, *Applied Numerical Mathematics* 45 (2003) 3–10.

Y. Saad. *Iterative Methods for Sparse Linear Systems*, 2nd Edition. SIAM, Philadelphia, 2003.

Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.* **7(3)**, 856–869, 1986.