GMRES Usage
M. Behr

http://www.cats.rwth-aachen.de/library/research/technotes

Abstract

This note describes the usage of the GMRES solver using reverse communication protocol. The GMRES control flow is outlined, and an example calling sequence explained.

1. Introduction

In an effort to reduce the cost required to solve a large linear system, we often lower our sights from the exact solution and become content with a solution which approximates the exact one with a prescribed accuracy. Such approximation can often be found at a greatly reduced expense. The iterative methods achieve this aim by constructing a series of guesses approximating the true solution. The conjugate gradient (CG) method has been extremely successful when applied to symmetric linear systems. Discounting the truncation errors, this method is known to produce an exact solution in at most \( N \) iterations, where \( N \) is the order of the linear system. For a review of this and earlier iterative methods see, e.g., [1]. Although several generalizations of the CG algorithm to non-symmetric linear systems exist, the Generalized Minimum Residual (GMRES) method introduced by Saad and Schultz [2] has proven to be a method of choice for many finite element researchers. We first outline a version of the algorithm which was introduced by Saad in [3]. Then we show an example of the calling sequence for the GMRES solver, as implemented in the EWD library.

2. Outline of the Algorithm

Consider the linear equation system:

\[
Ax = b.
\]

The outline of the GMRES algorithm is shown in Box 1. Given the equation system (1), an initial guess \( x_0 \), and a preconditioner matrix \( M \) (to be discussed in the next section), the process leads to an approximate solution vector \( x \) which minimizes the residual of the initial system over the preconditioned Krylov subspace. Each outer iteration involves solution of the linear system projected to a lower-dimensional subspace. The successive outer iterations differ in the quality of the initial guess \( x_0 \) only.

The inner iteration loop constructs the Krylov space and projects the original equation system to this space, producing the matrix \( \bar{H} \) which is by definition upper-Hessenberg. The modified Gramm-Schmidt orthogonalization of the basis vectors is typically the most computationally expensive part of the algorithm for moderate values of \( m \).
for $l = 1, \ldots, n_{\text{outer}}$

\[ r_0 := B - A x_0 \]
\[ \beta := \|r_0\|_2 \]
\[ v_1 = r_0 / \beta \]

for $j = 1, \ldots, m$

\[ z_j := M_j^{-1} v_j \]
\[ w := A z_j \]

for $i = 1, \ldots, j$

\[ h_{i,j} := (w, v_i) \]
\[ w := w - h_{i,j} v_i \]
\[ h_{j+1,j} := \|w\|_2 \]
\[ v_{j+1} := w / h_{j+1,j} \]

\[ \bar{H} := \{h_{i,j}\} \]
\[ y := \text{argmin}_y \|\beta e_1 - \bar{H} y\|_2 \]
\[ x := x_0 + \sum_{i=1}^{m} y_i z_i \]

if $\|\beta e_1 - \bar{H} y\|_2 \leq \varepsilon$ exit
else $x_0 := x$

Box 1: GMRES algorithm: control flow

The solution of the reduced system takes advantage of the upper-Hessenberg form of $\bar{H}$, as shown in Box 2. First the matrix $\bar{H}$ is reduced to an upper-triangular form by a series of Givens rotations. The optimal $y$ is then found through a simple back-substitution on the transformed system. Note that the operations enclosed in frames in Box 2 are omitted from practical implementations.

\[ p := \beta e_1 \]

for $j = 1, \ldots, m$

\[ \gamma := \sqrt{h_{j,j}^2 + h_{j+1,j}^2} \]
\[ c := h_{j,j} / \gamma \]
\[ s := h_{j+1,j} / \gamma \]

\[ \begin{cases} 
  h_{j,j} := c h_{j,j} + s h_{j+1,j} = \gamma \\
  h_{j+1,j} := -s h_{j,j} + c h_{j+1,j} = 0 
\end{cases} \]

\[ p_j := c p_j + h_{j+1,j} T_{j+1,j} \]
\[ p_{j+1} := -s p_j + h_{j+1,j} T_{j+1,j} \]

\[ \begin{bmatrix} 
  y_1 \\
  \vdots \\
  y_m 
\end{bmatrix} := 
\begin{bmatrix} 
  h_{1,1} & \cdots & h_{1,m} \\
  \vdots & \ddots & \vdots \\
  0 & \cdots & h_{m,m} 
\end{bmatrix}^{-1} 
\begin{bmatrix} 
  p_1 \\
  \vdots \\
  p_m 
\end{bmatrix} \]

Box 2: GMRES algorithm: solution of the reduced system
Box 3: GMRES algorithm: modified control flow

It should be noted that the algorithm shown in Box 1 differs from the classical GMRES in two ways. In the classical GMRES the Krylov space is continually expanded until convergence is reached. This leads to excessive memory requirements, and the advantage over a direct method of solution is lost. To alleviate this problem, in a practical GMRES algorithm the dimension of the Krylov space \( m \) is limited a priori, and if this dimension proves insufficient, the algorithm is restarted in the next outer iteration. Alternatively, the orthogonalization process can be truncated to include only a given number of previous basis vectors, but this option is not pursued here. Unfortunately, these modified methods are not guaranteed to converge, and the choice of the restart frequency has to be made carefully, usually by nu-
merical experimentation. The current algorithm also differs from other GMRES derivatives in that it allows variable preconditioning at each inner iteration. This feature enables us to, e.g., freely mix two different but complementary preconditioners, as described in [4].

In a more efficient (but less intuitive) version of the GMRES algorithm, the Givens rotations are computed and applied to the reduced system even as it is being constructed. This modification gives us an option of discontinuing the inner GMRES iterations when proper convergence criterion is met within the inner iteration loop. We use here the fact, that after \( j \) rotations applied to the reduced system, the current value of the norm \( \| \beta e_1 - Hy \|_2 \) is equal to the absolute value \( |p_{j+1}| \), as shown in [2]. The final algorithm adopted for present computations is thus presented in Box 3 and Figure 1.

For equation systems stemming from the increment form of the variational formulation the initial guess \( x_0 \) is normally taken as zero.

### 3. Preconditioning

In contrast to the direct methods of solution, the iterative schemes can be extremely sensitive to the particular numerical properties of the system matrix \( A \), and GMRES algorithm is no exception. The rate of convergence of Krylov subspace methods is influenced by the condition number of the matrix \( A \), as shown in [5]. The analysis contained therein indicates also that the order of convergence will degrade as eccentricity of the ellipse containing the eigenvalues of the linear system decreases. In practical terms, this measure is smaller for problems dominated by advection effects, as indicated by the Reynolds number. In most cases a preconditioning of the original system will be needed to achieve reasonable convergence rates. Preconditioning involves a matrix \( M \) resembling in some sense the matrix \( A \), but whose inverse is easier to compute. Assuming constant preconditioner matrix for the duration of the iterative process (as mentioned earlier, variable preconditioning is also possible), the (right) preconditioned system becomes

\[
AM^{-1} (Mx) = B. \tag{2}
\]

The closer the matrix \( AM^{-1} \) is to identity, the better convergence one may expect from the iterative solver. Once the solution \( Mx \) of (2) is obtained, the solution of the original system, i.e., the \( x \) vector itself, is easily computed.

Scaling of the linear system is a related concept. Here we construct a system

\[
W^{-1/2} AW^{-1/2} (W^{1/2} x) = W^{-1/2} B. \tag{3}
\]

The matrix of the iteratively solved system becomes thus \( W^{-1/2} AW^{-1/2} \). In contrast with the one-sided preconditioning, the scaled matrix retains the symmetry characteristics of the original system.

The simplest preconditioning method uses the diagonal part of the system matrix \( M = \text{diag} A \). Such preconditioning removes the worst effects of the large variations in the magnitude of diagonal entries, which for diagonally dominant matrices is closely related to a
high condition number. A diagonal scaling may be constructed for positive definite matrices with \( W = \text{diag} \ A \) as well. Diagonal preconditioning or scaling become less effective as the Reynolds number increases and off-diagonal entries assume importance.

4. Calling Sequence

The GMRES solver as implemented by the EWD library uses the reverse communication protocol (as employed in the serial version of the code written by Saad). The subroutine is:

```plaintext
subroutine ewdgmres(icode, bg, dg, wkg1, wkg2, z, v, hh, yy, cc, ss, &
   ninner, nouter, ndc, etolr, its, iit, aid)
integer icode, ndc, its, iit, ninner, nouter
real* 8 bg(ndc), dg(ndc), wkg1(ndc), wkg2(ndc)
real* 8 z(ndc,ninner), v(ndc,ninner+1), etolr
real* 8 hh(ninner+1,ninner), yy(ninner+1), cc(ninner), ss(ninner)
character*1 aid
```

where `icode` is the reverse communication switch (explained below), `ndc` is the number of unknowns, and `its` and `iit` are the current time step and nonlinear iteration (for information purposes only; they are included in the convergence report). The `ninner` and `nouter` are the size of the Krylov space \( m \) and the number of outer iterations \( n_{outer} \), respectively. The \( bg \) is the right-hand side \( b \) and the \( dg \) is the initial solution guess \( x_0 \) on input and the final solution \( x \) on exit. The \( wkg1 \) and \( wkg2 \) work vectors are used to communicate to the solver the effects of matrix \( A \) on a vector, as well as the effects of a preconditioner. The arrays \( z, v, hh, yy, cc \) and \( ss \) are used internally to store \( z, v, \bar{H}, y, c \) and \( s \) as seen in Box 3. These have to be allocated and preserved by the calling program. The `etolr` parameter can be used to specify the convergence criterium, and `aid` is a single-character identifier which helps distinguish between convergence data coming from multiple equation systems being solved simultaneously.

An example calling sequence is:

```plaintext
c    initialize the switch
   icode = 0

1 continue
   call ewdgmres(icode, res, del, wkg1, wkg2, zg, vg, hh, yy, cc, ss, &
      ninner, nouter, ndc, 1.0e-8, its, iit, '#')

   if (icode.eq.1) then
      c    preconditioning
         ...
         goto 1
   else if (icode.eq.2) then
```
The ewdgmrres subroutine returns alternately icode = 1 and icode = 2. In the first case, the user is responsible for applying the (left) preconditioner to vector wkg1 and storing the result in vector wkg2. In the second case, the user should apply the system matrix A to vector wkg1 and store the result in vector wkg2. The end of the process is reached when ewdgmrres returns icode = 0. That means that either the residual was reduced below the etolr value, or that the maximum number of outer iterations nouter was reached. At that point, the solution is stored in the vector dg.

History
September 19, 1991     Implemented.
April 28, 1999         Written (discussion of scaling incomplete).
February 23, 2001      Added EWD implementation flowchart.

References
Figure 1. GMRES algorithm: EWD implementation flowchart. Steady-state iteration paths are highlighted.