```
function [x, error, iter, flag] = gmres( A, x, b, M, restrt, max_it, tol )

%  -- Iterative template routine --
%     Univ. of Tennessee and Oak Ridge National Laboratory
%     October 1, 1993
%     Details of this algorithm are described in "Templates for the
%     Solution of Linear Systems: Building Blocks for Iterative
%     Methods", Barrett, Berry, Chan, Demmel, Donato, Dongarra,
%     Eijkhout, Pozo, Romine, and van der Vorst, SIAM Publications,
%     1993. (ftp netlib2.cs.utk.edu; cd linalg; get templates.ps).
%
% [x, error, iter, flag] = gmres( A, x, b, M, restrt, max_it, tol )
%
% gmres.m solves the linear system Ax=b
% using the Generalized Minimal residual ( GMRESm ) method with restarts .
%
% input   A        REAL nonsymmetric positive definite matrix
%         x        REAL initial guess vector
%         b        REAL right hand side vector
%         M        REAL preconditioner matrix
%         restrt   INTEGER number of iterations between restarts
%         max_it   INTEGER maximum number of iterations
%         tol      REAL error tolerance
%
% output  x        REAL solution vector
%         error    REAL error norm
%         iter     INTEGER number of iterations performed
%         flag     INTEGER: 0 = solution found to tolerance
%                           1 = no convergence given max_it

   iter = 0;                                         % initialization
   flag = 0;

   bnrm2 = norm( b );
   if  ( bnrm2 == 0.0 ), bnrm2 = 1.0; end

   r = M \ ( b-A*x );
   error = norm( r ) / bnrm2;
   if ( error < tol ) return, end

   [n,n] = size(A);                                  % initialize workspace
   m = restrt;
   V(1:n,1:m+1) = zeros(n,m+1);
   H(1:m+1,1:m) = zeros(m+1,m);
   cs(1:m) = zeros(m,1);
   sn(1:m) = zeros(m,1);
   e1    = zeros(n,1);
   e1(1) = 1.0;

   for iter = 1:max_it,                              % begin iteration

      r = M \ ( b-A*x );
      V(:,1) = r / norm( r );
      s = norm( r )*e1;
```

```
      for i = 1:m,                                   % construct orthonormal
         w = M \ (A*V(:,i));                         % basis using Gram-Schmidt
         for k = 1:i,
           H(k,i)= w'*V(:,k);
           w = w - H(k,i)*V(:,k);
         end
         H(i+1,i) = norm( w );
         V(:,i+1) = w / H(i+1,i);
         for k = 1:i-1,                              % apply Givens rotation
            temp     =  cs(k)*H(k,i) + sn(k)*H(k+1,i);
            H(k+1,i) = -sn(k)*H(k,i) + cs(k)*H(k+1,i);
            H(k,i)   = temp;
         end
         [cs(i),sn(i)] = rotmat( H(i,i), H(i+1,i) ); % form i-th rotation matrix
         temp   = cs(i)*s(i);                        % approximate residual norm
         s(i+1) = -sn(i)*s(i);
         s(i)   = temp;
         H(i,i) = cs(i)*H(i,i) + sn(i)*H(i+1,i);
         H(i+1,i) = 0.0;
         error  = abs(s(i+1)) / bnrm2;
         if ( error <= tol ),                        % update approximation
            y = H(1:i,1:i) \ s(1:i);                 % and exit
            x = x + V(:,1:i)*y;
            break;
         end
      end

      if ( error <= tol ), break, end
      y = H(1:m,1:m) \ s(1:m);
      x = x + V(:,1:m)*y;                            % update approximation
      r = M \ ( b-A*x )                              % compute residual
      s(i+1) = norm(r);
      error = s(i+1) / bnrm2;                        % check convergence
      if ( error <= tol ), break, end;
   end

   if ( error > tol ) flag = 1; end;                 % converged

% END of gmres.m
```