034702-2-T

CONVERGENCE STUDY OF FEM SYSTEMS WITH PML IMPLEMENTATION

Y. Botros
J. Volakis

2nd Quarterly Report

Compact Software, Inc.
201 McLean Boulevard
Paterson, N.J. 07504

January 1997

34702-2-T = RL-2474

PROJECT INFORMATION

PROJECT TITLE: Accelerated 3D Arbitrary EM Simulation Methods

REPORT TITLE: Convergence Study of FEM Systems with PML Implementation

U-M REPORT No.: 034702-2-T

CONTRACT
START DATE: 1 August 1996
END DATE: 31 July 1997

DATE: January 1997
(2nd Quarterly Report)

SPONSOR: Jason Gerber
Compact Software Inc.
201 McLean Ave.
Paterson, N.J 07504

SPONSOR
CONTRACT No.: 96-0502

U-M PRINCIPAL
INVESTIGATOR: John L. Volakis
EECS Dept.
University of Michigan
1301 Beal Ave
Ann Arbor, MI 48109-2122
Phone: (313) 764-0500 FAX: (313) 747-2106
volakis@umich.edu
http://www-personal.engin.umich.edu/~volakis/

CONTRIBUTORS
TO THIS REPORT: Y. Botros, J. Volakis and J. Gong

Forward

This document represents our 2nd quarterly report on this project. In our previous report we presented new design guidelines and curves for the PML and reported that the PML showed substantial improvement in absorption performance. However at the same time, it also caused some deterioration of the iterative solver convergence. Our efforts over the past quarter were therefore concentrated on improving the convergence of the solver when the PML is used for mesh truncation which typically consumes 90% of the CPU time. Three solvers were examined in this regard and curves are shown in the report which demonstrate solver performance when a preconditioner is used. Most importantly, we re-examined the values of the absorber's parameters in terms of both performance and convergence. As a result of this study, we concluded that certain ranges of the propagation constant (in addition to the phase constant) led to must faster convergence rates. The given curves demonstrate the appropriate values of the PML loss and phase constants.

Having developed a good set of design criteria for the PML absorber, we began their applications (as planned, this is a 3rd quarter task) to more complex microwave structures. One of the geometries considered in this report is a pair of coplanar microstrip transmission lines. Our goal is to use such a geometry to examine the utility of the PML as a mesh truncator. We are looking at feed modeling (some curves are reported) and at coupling reduction issues, particularly when a dielectric well is used to better isolate the co-planar microstrip lines.

Of potential interest to this project is a recent investigation which we are carrying out for a different project dealing with large scale systems for modeling antenna arrays. To handle the large number of unknowns (at the expense of some additional memory requirements), we looked at a recently introduced method for solving sparse systems using a new type of LU solver referred to as CVSS. This solver was originally written for parallel platforms and we recently implemented it for workstations. We have really been impressed with the performance of this new solver. Below are some examples of the run times on an HP workstation rated at 47 Mflop peak speed:

Cases	# of Eqns	# of Nonzero Elements	Time (secs)
1	2,800	37,595	5.34
2	6,448	89,365	22.12
3	21,200	305,845	119.81
4	62,769	650,355	690.10

Cases number 3 and 4 are truly impressive and correspond to large scale microwave circuit computations. Such system sizes refer to real-world microwave circuit applications and therefore CVSS should be considered as a possible alternative to iterative solvers. In the next few months we would like to provide some comparisons of CVSS with the iterative solvers for typical microwave circuits. As planned, our primary focus will be to continue with the application of PML for modeling complex microwave structures.

Convergence Study of FEM Systems with PML Implementation

Youssry Y. Botros and John L. Volakis

Radiation Laboratory

Department of Electrical Engineering

and Computer Science,

The University of Michigan

Ann Arbor, MI 48109-2212

Abstract

When solving Finite Element systems, the solver performance is crucial to the efficiency of the simulation since it often consumes most of the CPU time (90 % or more). This is particularly so when modeling the FEM domains with a perfectly matched layer (PML) absorber. In this report, we address approaches to improving the solver performance. Specifically, the PML absorber parameters are optimally selected for improving the convergence. Using these optimal values, the system matrix is preconditioned using the simple diagonal preconditioner. We also examine the performance of three different iterative solvers interms of their convergence characteristics. In all cases, three dimensional examples representing microwave circuits are utilized.

Contents

1 Introduction

When using the Finite Element Method (FEM), it is necessary to truncate the computational domain by a suitable artificial non reflecting surface. Such termination schemes can be classified into two main categories; Absorbing Boundary Conditions (ABCs) and the Material Absorbers. The latter can be either isotropic or anisotropic. Absorber layers offer several advantages over the ABCs, including ease of implementation, parallelization and conformality. Recently, a superior perfectly matched layer (PML) material absorber was introduced for truncating finite element meshes. This truncation scheme was introduced by Sacks et. al. [1] and a two dimensional parametric extensive study for their design, performance and optimization was performed in [2]. In [3] and [4], the superior performance of the PML was demonstrated and their implementations were extended to circuits, filters and scattering problems. In all of these studies, the PML reflectivity was the main concern and its performance was addressed with respect to the phase shift, attenuation, number of layers and thicknesses. However, the solver convergence issue was not addressed in these investigations.

To make the PML absorbers useful for three dimensional implementations, it is necessary to improve the associated system convergence rate and to determine the optimal selection of the parameters when used in numerical codes. This report has two main issues, the optimization of the PML absorber with respect to both absorption and convergence and the application of the optimal absorber to real microwave circuits. These applications include the computations of the fields in a microwave circuit.

Regarding the PML optimization, we first discuss the effects of the absorber parameters (attenuation and phase shift) on the convergence of the resulting FEM system. Then, we proceed by preconditioning the system matrix so that faster convergence is achieved. Finally, we discuss the different solvers and compare their performance. For the microwave circuits study, we solve the FEM system and we extract the fields in the different parts of the circuit

using the results obtained from the PML parametric design using the appropriate iterative solver.

2 Formulation

Consider a wave incident upon the interface between two media shown in Figure 1. Layer 2 is a uniaxial absorber with $\overline{\overline{\mu}}_r$ and $\overline{\overline{\epsilon}}_r$ representing the relative constitutive parameter tensors for medium 2 of the form

$$\overline{\overline{\mu}}_r = \overline{\overline{\epsilon}}_r = \begin{pmatrix} a_2 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & c_2 \end{pmatrix} \tag{1}$$

According to [1], a zero reflection coefficient along the interface can be obtained by choosing $a_2 = b_2 = 1/c_2 = \alpha - j\beta$ where α represents the phase shift factor and β represents the attenuation coefficient. Because of this property of the interface, the interface is referred as to a Perfectly Matched Layer (PML). Our two and three dimensional study in the previous report [3] focused on understanding and improving the reflectivity of the PML. However, none of these studies discussed the issue relating to the convergence of the iterative solver. In this report, we will carry out such a study In all steps of the convergence study, the discretization (sampling rate) and dimensions of the domain will be kept the same. Also, the performance of the solver will be evaluated using the parameter

$$r = \frac{Number\ of\ Iterations\ before\ Convergence}{FEM\ system\ size} \tag{2}$$

For small values of r, the system is considered as rapidly converging but for r near unity, the system is poorly converging.

3 Convergence as a function of the Absorber Parameters

The phase and loss parameters of the PML play a key role on the convergence of the FEM system. Therefore, there is a need to find certain ranges for these two parameters so that convergence is optimized without deteriorating the PML performance. As an example, we considered the convergence of the FEM system for the microstrip line shown in Figure 2. The FEM system size was approximately 3000 and the Standard BiConjugate Gradient (BCG) routine for non-symmetric matrices was used to solve the system. In studying the effects of the phase factor α, we fixed the value of β at unity. We also fixed $\alpha = 1$ to study the effects of β on the convergence. Below, we discuss the results of the study

3.1 Effect of α on solver convergence

To discuss the effects of varying the phase coefficient α (with β =1), we scanned its value from zero to 5 using a step size of 0.25. At each value of α, we examined the three main quantities during and after completion of the FEM system solution. These are the number of iterations required for convergence and the corresponding iteration ratio r, the fields under the microstrip line and the corresponding value of the Reflection coefficient R and the residue error history.

Figure 3 shows the reflection coefficient curves. They indicate that maximum absorption occurs around $\alpha = 1$ and the corresponding r was approximately 0.15. It is clear that increasing α deteriorates the absorption with no benefit to the convergence rate. We can therefore conclude that the maximum absorption occurs when the value of α is around unity. Also, at this value, the convergence rate is best when compared to the other values of α. It should be noted that $\alpha = 1$ represents the free space case and as deduced from the graphs,

it also offer the best absorption and convergence.

3.2 Effect of β On the convergence

In this part, a parametric study is performed on β with $\alpha = 1$. The β range is scanned from zero to 5 with .25 increments. The number of convergence iterations, field under the microstrip line and the residue value at every iteration are calculated during and after the completion of the solution to extract the convergence parameters as well as the reflection coefficient.

Figure 4 displays R in dB and the ratio r as a function of β for $\alpha = 1$. As shown, the value of β around unity is optimum interms of the convergence rate and maximum absorption. We also observe that the convergence rate deteriorates with increase in β. This is related to the abrupt changes in the field values among the adjacent segments when the attenuation coefficient is increased. Thus, the iterative solver finds it difficult to extract the fields in all the segments and hence more iterations are needed unless a much higher sampling rate is employed. From Figures 3 and 4, we can conclude that the optimal values for achieving both convergence and minimal reflection coefficient are $\alpha = \beta = 1$.

4 Preconditioning the FEM system

Using the optimal values for α and β, we now proceed to discuss the effects of the preconditioner on the solver convergence. To improve the system condition, we implemented and tested a diagonal preconditioner. There are two reasons for selecting this type of preconditioner. Among them is its simplicity minimal computational overhead. Also, the performance of this preconditioner is superior when compared to more complicated preconditioner such as the block preconditioners [5] and [6]. The significance of the preconditioner is illustrated in Figure 5, where the number of iterations ratio dropped from 0.155 to 0.95 when the diagonal preconditioner is applied to the FEM system before starting the BCG solver. In the next section, we will use the diagonal preconditioners with each of the implemented solvers. In general, the diagonal preconditioner saves from 30 percent to 60 percent of the total number of iterations for all of the iterative solvers. This percentage is highly dependent on many factors and parameters. The element shape employed in the computational domain, the sampling or discretization rate, the layer (absorber) parameters, etc . . ., all affect the performance.

5 Iterative Solvers

During this quarter, we looked at the performance of three following solvers

- The BiConjugate Gradient (BCG).
- The Quasi-Minimal Residual (QMR).
- The Generalized Minimal Residues (GMRES)

These are implemented and tested for the three dimensional example in Figure 2. The systems are preconditioned using the diagonal preconditioner. The results in Figure 6 demonstrates that r is approximately 0.095 for both the BCG and QMR algorithms and 0.045 when GMRES is used. Also, the error history is smoother for the GMRES when compared with the QMR. Below, we make a brief comparison of the attributes for the three solvers.

5.1 The BiConjugate Gradient (BCG) Solver

This solver can be applied to nonsymmetric matrices. It requires matrix-vector products with the coefficient matrix and its transpose. The two matrix-vector products and the preconditioning steps are independent. Thus, these operations can be done in parallel, or their communication stages can be packaged. For a matrix of order N, the BCG storage requirement is given by the matrix itself plus $10N$. It is observed here that the convergence behavior may be quite irregular and the method can breakdown. The breakdown or near breakdown situations can be sometimes avoided by a restart at the iteration step immediately before the breakdown (near breakdown step). Another method is to switch to a more robust (but more expensive) method such as GMRES.

5.2 The Quasi Minimal Residual (QMR) Solver

The QMR system is also applicable for nonsymmetric matrices. It is designed to avoid the irregular behavior of the BCG and possible breakdowns. Its improvement per iteration step is similar to the BCG. However, when the BCG starts to diverge, QMR may still produce better residue behavior. QMR requires one matrix-vector product with the coefficients of the matrix and its transpose. As in BCG, these two matrix-vector products (as well as the preconditioners) are independent. Therefore, parallelization is easily achievable. The storage requirements for QMR are those needed for the matrix itself plus $16N$. As a whole, QMR produces similar performance to the BCG with lower breakdown possibilities.

5.3 The Generalized Minimal Residual (GMRES) Solver

The GMRES system is applied to nonsymmetric systems and leads to the smallest residual for a fixed number of iteration steps. However, each of these iteration steps become increasingly expensive. To limit the increasing storage requirements and work per iteration step, restarting is necessary. Depending on the system matrix [A] and the excitation vector {b}, the number of restarts should be chosen carefully. The proper choice of this number requires *a priori* experience and it is fully dependent on the system parameters and sampling rate. Regarding the number of operations, the GMRES requires only one matrix-vector product but the number of inner products increases linearly with the iteration step. For storage, the matrix is needed plus $(i+1)N$ where i is the iteration step starting from one up to the total number of restarts.

6 Three Dimensional Example

In this section, we apply the optimal parameters, values and techniques to a practical three dimensional example representing actual microwave circuits and structures. We first start by evaluating the fields under the conductor microstrip of the circuit shown in Figure 2. Finally, we consider a multiconductor configuration to study the coupling between two adjacent transmission lines. We also look at methods of reducing the coupling among for complex multiport microwave circuits.

6.1 Microstrip Lines and Feed Probes

The configuration is shown in Figure 2 and to truncate the microstrip line, we used the optimal values of the PML absorber parameters ($\alpha = \beta = 1$) with a sampling rate on the order of 15 samples per wavelength. A simple diagonal preconditioner was employed and the BCG algorithm was used to solve the resulting linear system. To evaluate the efficiency of the PML, we examine the fields under the microstrip line. Figure 7 shows the field under the microstrip for the following cases:

1. The parameters are optimally selected.

2. With a non optimal PML parameter selection.

It is clear that the optimal selection of the parameters gives the best reflection from the line termination. Figures 8 shows the field under the microstrip line when the number of probes modeling the feed is increased from 1 to 5. It is clear that the field increases linearly with the number of probes. This is expected because the input power to the transmission line is increased. However, the input impedence remains constant and levels off when the number of feeding probes increases to 4.

6.2 Coupling Study

We have already begun this study of the multiconductor geometry shown in Figure 9. Analysis of coupling between two transmission lines in conjunction with the FEM method is now being developed. As shown in Figure 9, the left transmission line is excited while both back ports are matched. The scattering parameters of this multiport will be evaluated after solving the resulting FEM system with the domain truncated using the optimal PML discussed in this paper.

References

[1] Z.S. Sacks, D.M. Kingsland, R. Lee and J.F. Lee, "A perfectly matched anisotropic absorber for use as an absorbing boundary condition," *IEEE Trans. Antennas and Propagation*, December 1994.

[2] S. Legault, T.B.A. Senior and J.L. Volakis, "Design of Planar Absorbing Layers for Domain Truncation in FEM Applications," *Electromagnetics J.*, vol 16, no.4, July-August 1996.

[3] Y.Y. Botros, S.R. Legault, J. Gong, J.L. Volakis and T.B.A. Senior, "Perfectly Matched Absorbers (PML):Theory, Analysis and Implementation Radiation Laboratory Report (No. 034702-1), The university of Michigan, Ann Arbor, October 1996.

[4] J. Gong, S.R. Legault, Y.Y. Botros, J.L. Volakis and P. Petre, "Application and Design Guidelines of the PML Absorber for Finite Element Simulations of Microwave Packages", Microwave Theory and Techniques Symp., San Fransisco, June 1996.

[5] R. Barret et. al., *Templetes for the Solution of Linear Systems: Building Blocks for Iterative Solvers*, siam , 1994.

[6] A. Chaterjee, J L. Volakis and D. Windheiser,"Parallel Computation of 3D electromagnetic Scattering using Finite Elements," Int. J. Numerical Modeling: Electr. Net. Dv. and Fields, Vol. 7, pp 329-342, 1994.

List of Figures

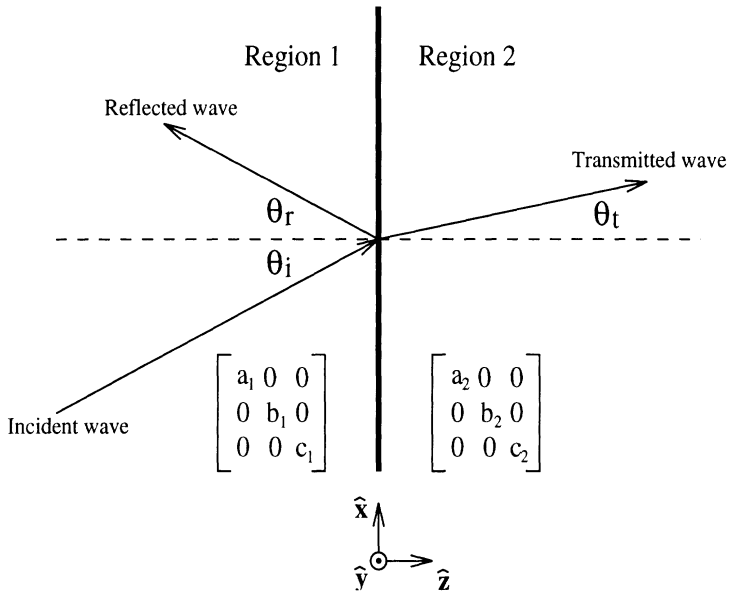

Figure 1: Plane wave incidence on an interface between two diagonally anisotropic half-spaces.

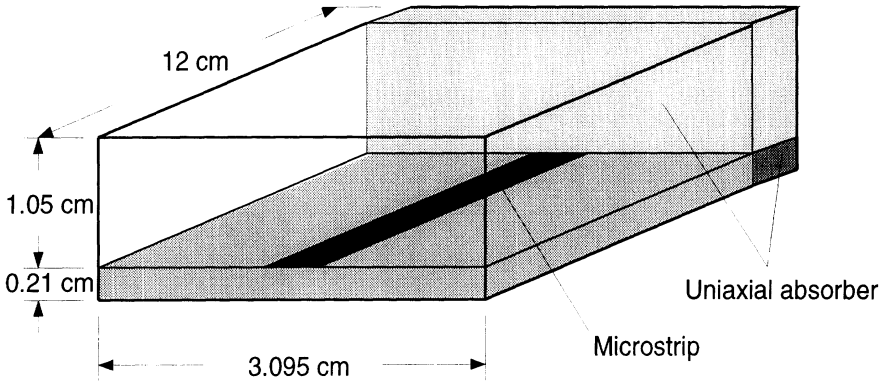

Figure 2: Microstrip Line

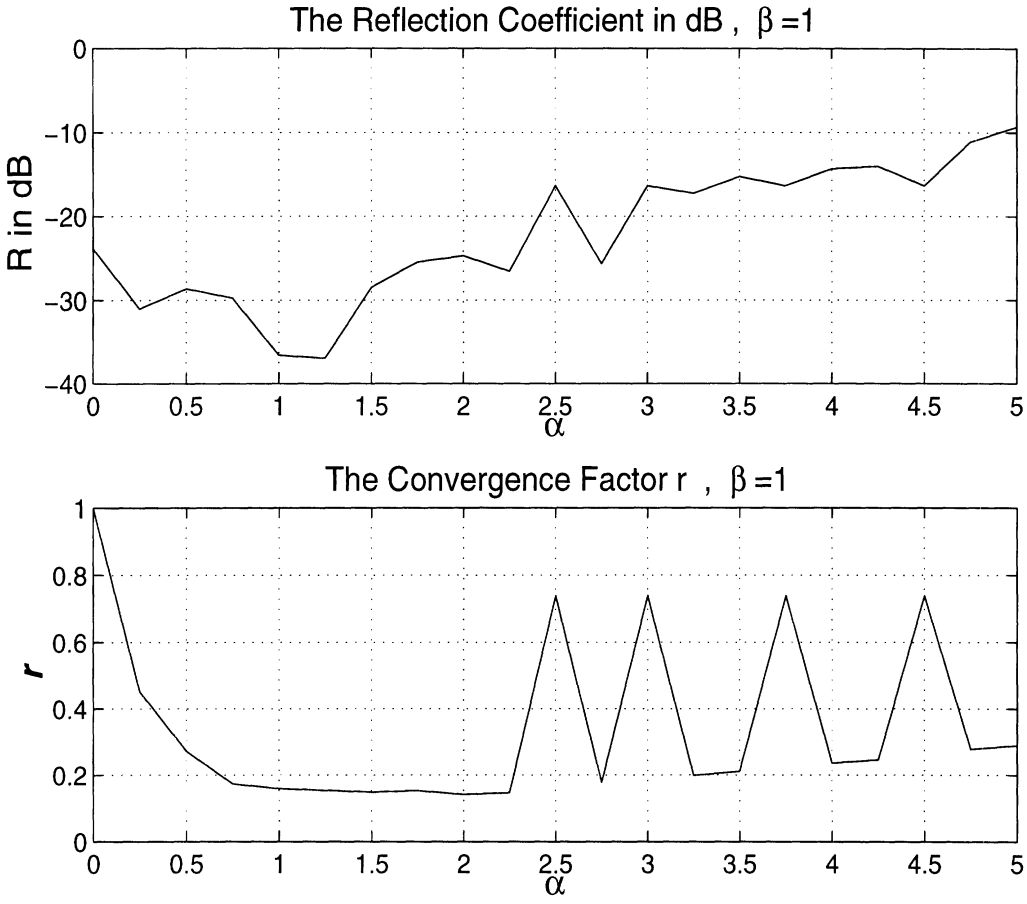

Figure 3: Effect of α on both convergence and absorption

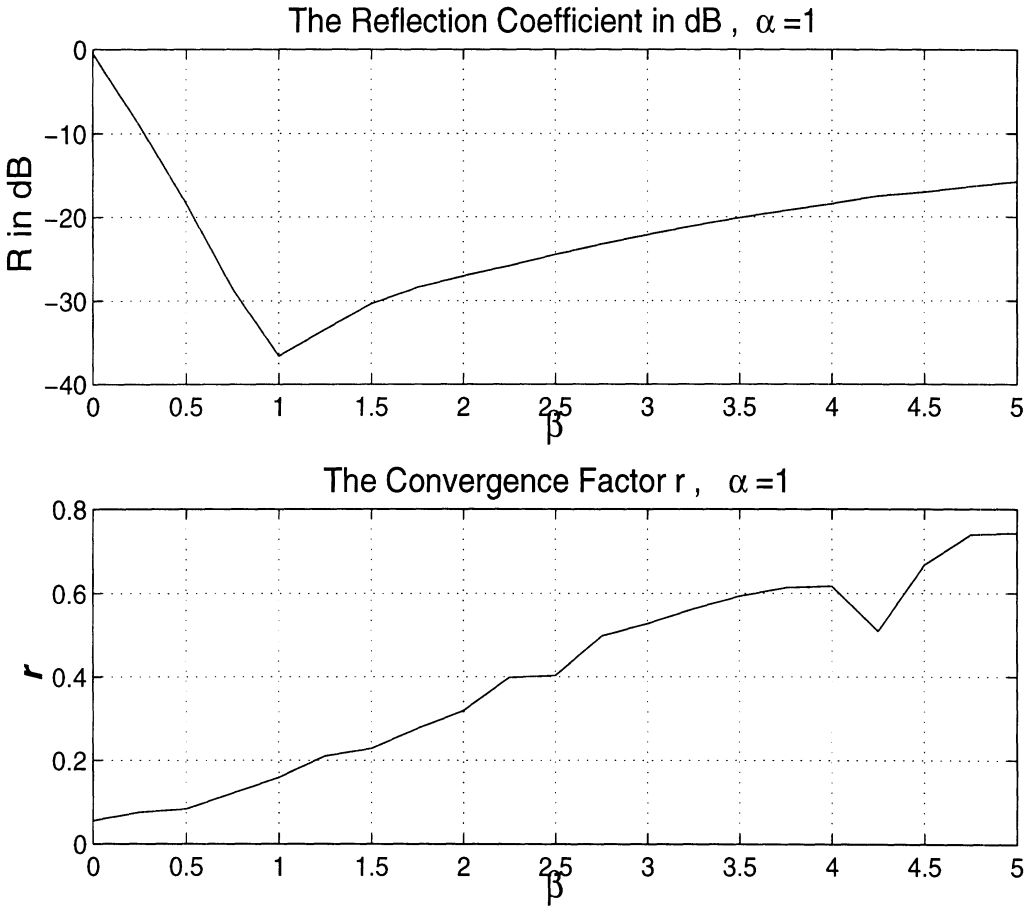

Figure 4: Effect of β on both convergence and absorption

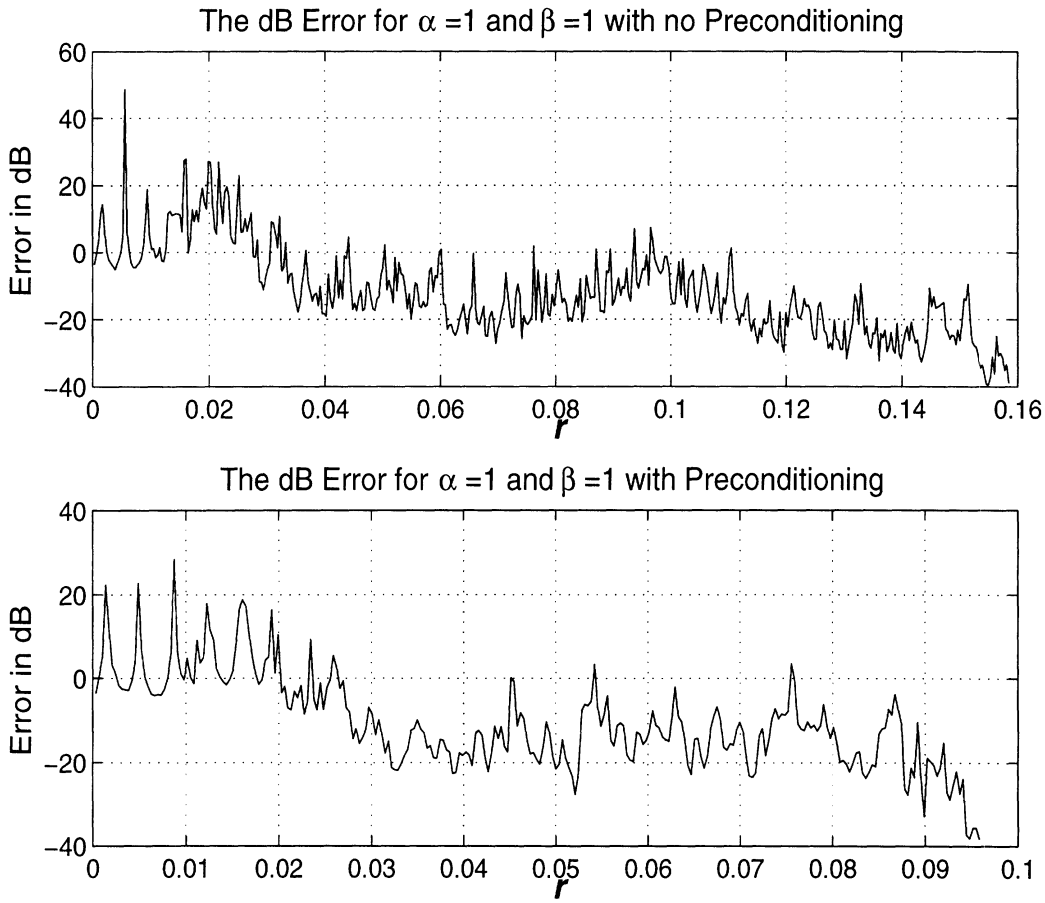

Figure 5: Preconditioning study

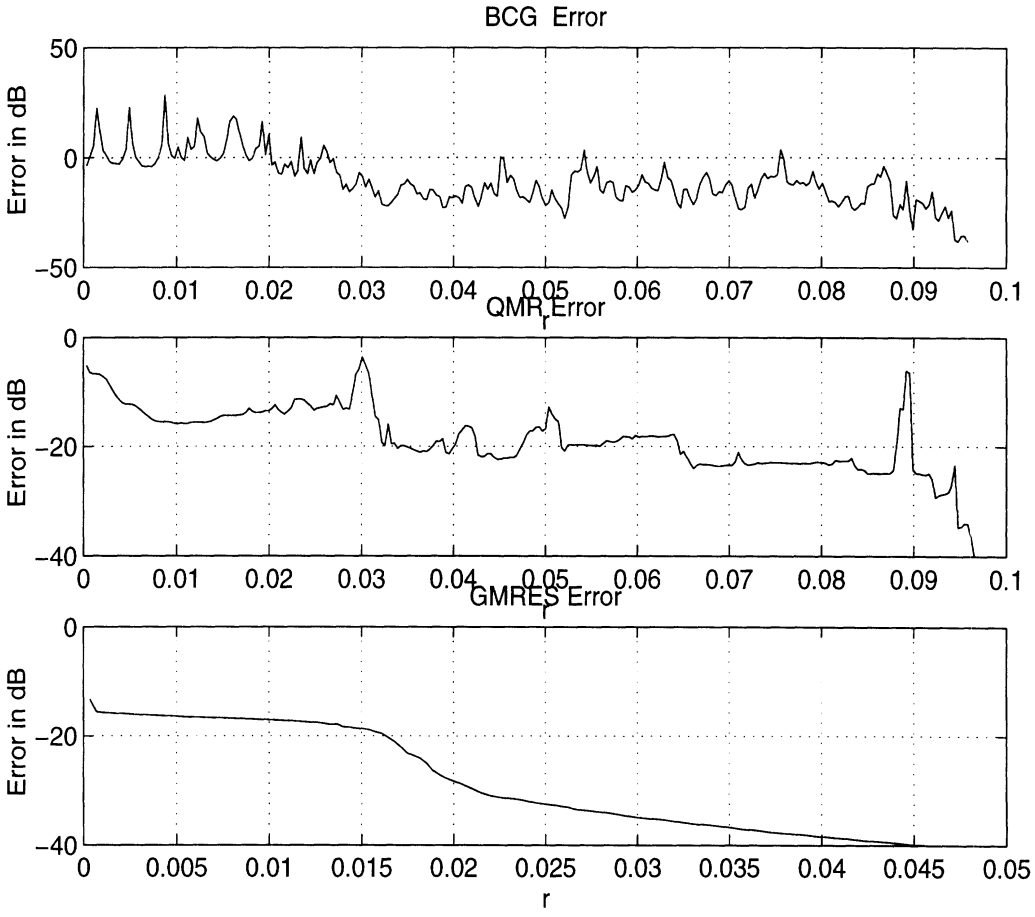

Figure 6: Comparison Between the Convergence of three Different Solvers

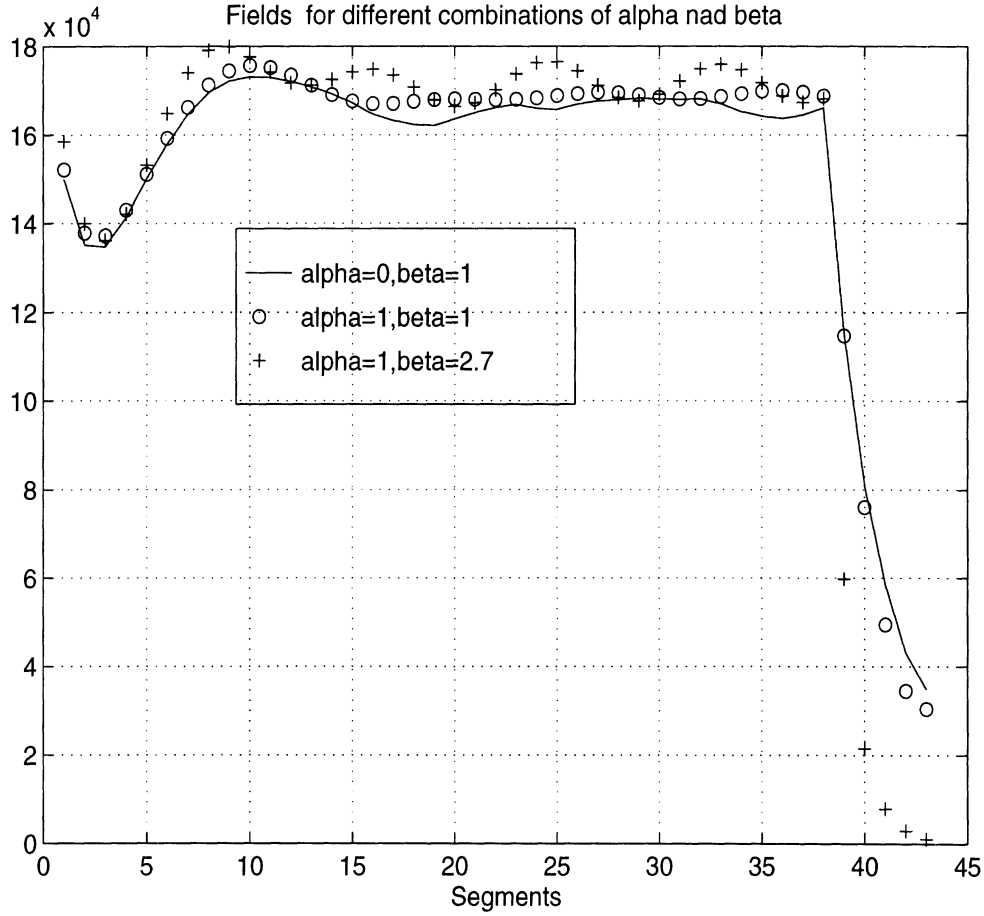

Figure 7: Fields under the microstrip line shown in Figure 2 for different choices of *alpha* and β. Note that the case of $alpha = \beta = 1$ is optimal

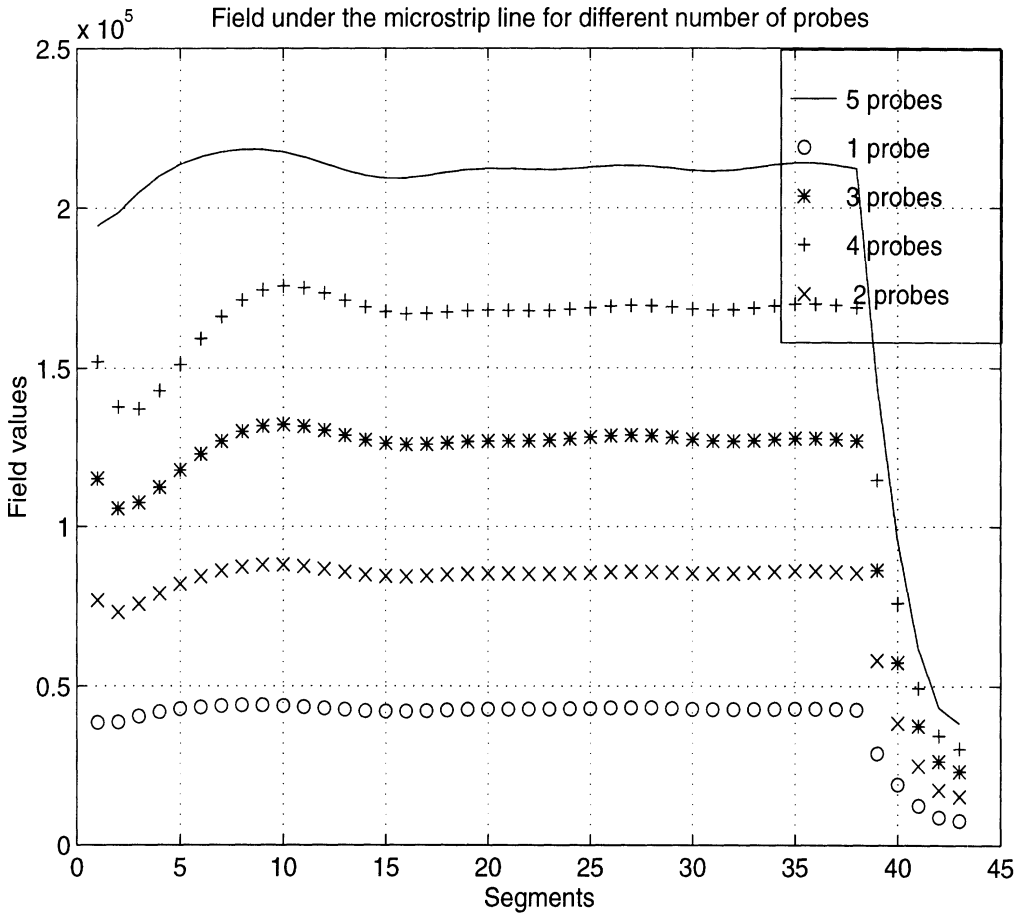

Figure 8: Fields under the microstrip line when different number of probes are used to model the feed

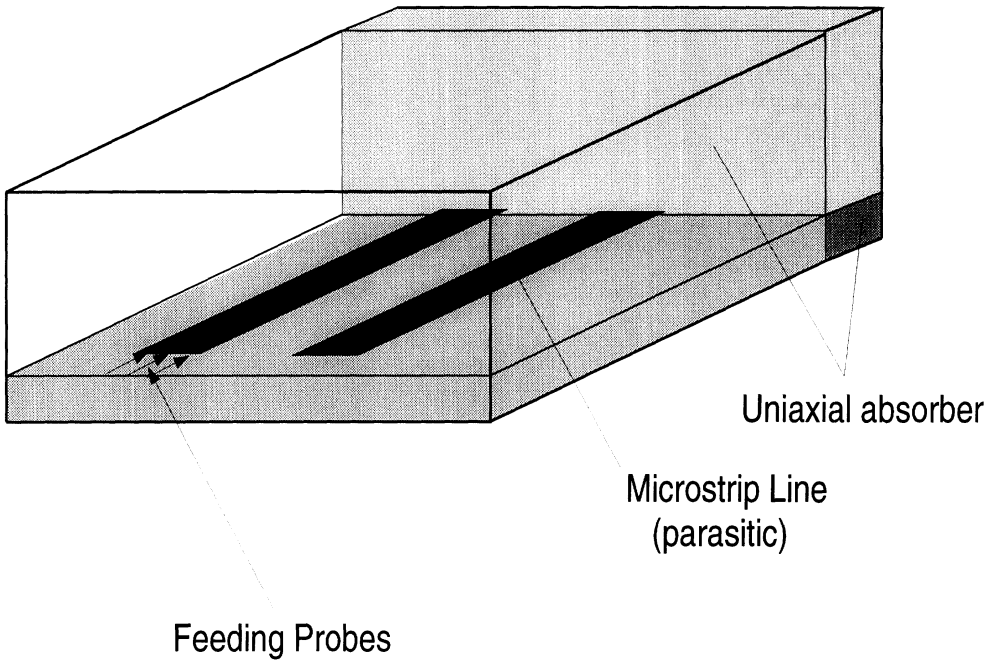

Figure 9: Geometry of the Circuit for Coupling Studies

7 Appendix A: The Solver Input Data

For each of the following solvers, we will input the following four vectors produced by the FEM code. These are:

1. rvec is a vector specifying the row indices of the non-zero elements of the overall FEM matrix. That is, rvec gives the i index of the matrix entry A(i,j).

2. cvec is a vector specifying the corresponding column indices of the non-zero elements of the overall FEM matrix. That is, gives the j index of the matrix entry A(i,j)

3. Cvec is a vector containing the entries of the non-zero elements of the FEM matrix.

4. b is a complex Excitation Vector.

Example: If [A] is give by

$$A = \begin{bmatrix} 15 & 0 & 20 & 0 \\ 0 & 16 & 0 & 40 \\ 0 & 0 & 12 & -4 \\ 10 & 14 & 0 & -10 \end{bmatrix} \tag{3}$$

then rvec={1,1,2,2,3,3,4,4,4}

cvec={1,3,2,4,3,4,1,2,4}

Cvec={15,20,16,40,12,-4,10,14,-10}

8 Appendix B: BCG Subroutine

```
function [x,iter,flag,ebcg,nfbcg] =bicg(rvec,cvec,Cvec,b)
% Input vectors:
% 1- rvec--- row indices of the non-zero entries of the matrix A.
% 2- cvec--- column indices of the non-zero entries of the matrix A.
% 3- Cvec--- Values of the non-zero entries of the matrix A.
% 4- b --- excitation (feed) vector

A=sparse( rvec, cvec,Cvec);
flops(0)
tol=10^(-2);
FEMsize=length(b);
FEMsize
max_it=FEMsize;
iii=ones(1:FEMsize);
mmm=1:FEMsize;
M=sparse(mmm,mmm,iii);
x=zeros(FEMsize,1); %initialization of the iterations.
Ad=diag(A);
M=sparse(mmm,mmm,Ad); %diagonal preconditioning.

% bicg.m solves the linear system Ax=b using the
% BiConjugate Gradient Method with preconditioning.
%
% input   A        is the input Sparse FEM Matrix for the solver.
%         M        is the diagonal preconditioner matrix
```

```
%         x        initial guess vector (put as zeros).
%         b        right hand side vector (excitation vector).
%         max_it   maximum number of iterations  and we set it here equal to the tota
%         tol      error tolerance (the error is defined as (norm(A*x-b)/norm(b))
%
% output  x        solution vector
%         ebcg     error norm history (the error in each iteration).
%         iter     INTEGER number of iterations performed.
%     nfbcg  Number of flops used in the solver.
%     flag     INTEGER: 0 = solution found to tolerance.
%                           1 = no convergence given max_it.
%                          -1 = breakdown.

  iter = 0;                                % initialization
  flag = 0;

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

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

  r_tld  = r;
```

```
  for iter = 1:max_it                    % begin iteration
iter
error
     z = M \ r;
     z_tld = conj(M') \ r_tld;
     rho   = (conj( z')*r_tld );
     if ( rho == 0.0 ),
        break
     end

     if ( iter > 1 ),                    % direction vectors
        beta = rho / rho_1;
        p    = z  + beta*p;
        p_tld = z_tld + beta*p_tld;
     else
        p = z;
        p_tld = z_tld;
     end

     q = A*p;                            % compute residual pair
     q_tld = conj(A')*p_tld;
     alpha = rho / (conj(p_tld')*q );

     x = x + alpha*p;                    % update approximation
     r = r - alpha*q;
```

```
      r_tld = r_tld - alpha*q_tld;


      error = norm( r ) / bnrm2;          % check convergence
    if ( error <= tol ), break, end


      rho_1 = rho;
e(iter)=error;
  end


  if ( error <= tol ),                    % converged
     flag =  0;
  elseif ( rho == 0.0 ),                  % breakdown
     flag = -1;
  else
     flag = 1;                            % no convergence
  end


nfbcg=flops;
ebcg=e;
% END bicg.m
```

9 Appendix C: QMR Subroutine

```
function [x, error, eqmr,nfqmr] = qmr(rvec,cvec,Cvec,b)
% Input vectors:
% 1- rvec--- row indices of the non-zero entries of the matrix A.
% 2- cvec--- column indices of the non-zero entries of the matrix A.
% 3- Cvec--- values of the non-zero entries of the matrix A.
% 4- b --- excitation (feed) vector

A=sparse( rvec, cvec,Cvec);
flops(0)
tol=10^(-2);
FEMsize=length(b);
FEMsize
max_it=FEMsize;
iii=ones(1:FEMsize);
mmm=1:FEMsize;
M=sparse(mmm,mmm,iii);
x=zeros(FEMsize,1); %initialization of the iterations.
Ad=diag(A);
M=sparse(mmm,mmm,Ad); %diagonal preconditioning.

% bicg.m solves the linear system Ax=b using the
% BiConjugate Gradient Method with preconditioning.
%
% input   A        is the input Sparse FEM Matrix for the solver.
```

```
%         M         is the diagonal preconditioner matrix
%         x         initial guess vector (put as zeros).
%         b         right hand side vector (excitation vector).
%         max_it    maximum number of iterations and we set it here equal to the total
%         tol       error tolerance (the error is defined as (norm(A*x-b)/norm(b))
%
% output  x         solution vector
%         ebcg      error norm history (the error in each iteration).
%         iter      INTEGER number of iterations performed.
%      nfbcg  Number of flops used in the solver.
%      flag      INTEGER: 0 = solution found to tolerance.
%                         1 = no convergence given max_it.
%
%   breakdown:
%                        -1: rho
%                        -2: beta
%                        -3: gamma
%                        -4: delta
%                        -5: ep
%                        -6: xi


   iter = 0;                                  % initialization
   flag = 0;
```

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

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

    [M1,M2] = lu( M );

    v_tld = r;
    y = M1 \ v_tld;
    rho = norm( y );

    w_tld = r;
    z = conj(M2') \ w_tld;
    xi = norm( z );

    gamma =  1.0;
    eta = -1.0;
    theta =  0.0;
xold=x;
    for iter = 1:max_it,                    % begin iteration

iter
```

```
 if ( rho == 0.0 | xi == 0.0 ), break, end


 v = v_tld / rho;
 y = y / rho;


 w = w_tld / xi;
 z = z / xi;


 delta = conj(z')*y;
 if ( delta == 0.0 ), break, end


 y_tld = M2 \ y;
 z_tld = conj(M1')\ z;


 if ( iter > 1 ),                          % direction vector
    p = y_tld - ( xi*delta / ep )*p;
    q = z_tld - ( rho*delta / ep )*q;
       else
    p = y_tld;
    q = z_tld;
 end


 p_tld = A*p;
 ep = conj(q')*p_tld;
 if ( ep == 0.0 ), break, end
```

```
beta = ep / delta;
if ( beta == 0.0 ), break, end

v_tld = p_tld - beta*v;
y =  M1 \ v_tld;

rho_1 = rho;
rho = norm( y );
w_tld = ( conj(A')*q ) - ( beta*w );
z =  conj(M2') \ w_tld;

xi = norm( z );

gamma_1 = gamma;
theta_1 = theta;

theta = rho / ( gamma_1*beta );
gamma = 1.0 / sqrt( 1.0 + (theta^2) );
if ( gamma == 0.0 ), break, end

eta = -eta*rho_1*(gamma^2) / ( beta*(gamma_1^2) );

if ( iter > 1 ),                          % compute adjustment
   d = eta*p + (( theta_1*gamma )^2)*d;
```

```
        s = eta*p_tld + (( theta_1*gamma )^2)*s;
      else
        d = eta*p;
        s = eta*p_tld;
      end

      x = x + d;                                % update approximation

      r = r - s;                                % update residual

      error2=norm(x-xold)/norm(x)
      error = norm( r ) / bnrm2;                % check convergence
      eqmr(iter)=error;
      if ( error2 <= tol ), break, end
xold=x;
   end

   if ( error <= tol ),                         % converged
      flag =  0;
   elseif ( rho == 0.0 ),                       % breakdown
      flag = -1;
   elseif ( beta == 0.0 ),
      flag = -2;
   elseif ( gamma == 0.0 ),
      flag = -3;
```

```
    elseif ( delta == 0.0 ),
       flag = -4;
    elseif ( ep == 0.0 ),
       flag = -5;
    elseif ( xi == 0.0 ),
       flag = -6;
    else                                          % no convergence
       flag = 1;
    end
nfqmr=flops
%  END qmr.m
```

10 Appendix D: GMRES Subroutine

```
function [x, error, egmres,nfgmres,iter]= gmres(rvec,cvec,Cvec,b)
% Input vectors:

% 1- rvec--- row indices of the non-zero entries of the matrix A.
% 2- cvec--- column indices of the non-zero entries of the matrix A.
% 3- Cvec--- values of the non-zero entries of the matrix A.
% 4- b --- excitation (feed) vector

A=sparse( rvec, cvec,Cvec);
flops(0)
tol=10^(-2);
FEMsize=length(b);
FEMsize
max_it=FEMsize;
iii=ones(1:FEMsize);
mmm=1:FEMsize;
M=sparse(mmm,mmm,iii);
x=zeros(FEMsize,1); %initialization of the iterations.
Ad=diag(A);
M=sparse(mmm,mmm,Ad); %diagonal preconditioning.

% bicg.m solves the linear system Ax=b using the
% BiConjugate Gradient Method with preconditioning.
%
```

```
% input   A        is the input Sparse FEM Matrix for the solver.
%         M        is the diagonal preconditioner matrix
%         x        initial guess vector (put as zeros).
%         b        right hand side vector (excitation vector).
%         max_it   maximum number of iterations (we set it here equal) to the
total FEM size.
%         tol      error tolerance (the error is defined as (norm(A*x-b)/norm(b)).
%
% output  x        solution vector
%         ebcg     error norm history (the error in each iteration).
%         iter     INTEGER number of iterations performed.
%     nfbcg  Number of flops used in the solver.
%     flag     INTEGER: 0 = solution found to tolerance.
%                           1 = no convergence given max_it.
%                          -1 = breakdown.



   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
iter
error
      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
egmres(iter)=error;
      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
nfgmres=flops;
% END of gmres.m
```