Efficient Harmonic Balance Analysis of Microwave Circuits Using Iterative Matrix Solvers and Newton Updating

Shunmin Wang

Master of Science

North Carolina State University

1999

Abstract

Several methods are introduced to improve the nonlinear equation solver arising from harmonic balance analysis of microwave circuits. The nonlinear equation solver is based on Newton's method. Two direct methods (QR and LU decomposition) and one iterative method (GMRES with left and right preconditioner) are investigated for matrix solve. These with the Broyden's Jacobian update method and the modified Broyden's Jacobian update method for iterative nonlinear solve are compared for sparse and dense soliton circuits. These are compared for the harmonic balance solution of both a moderately and strongly driven soliton circuit and a grid amplifier quasi-optical power combining circuit. After thresholding of the Jacobian these circuits result in a range of sparsities of the Jacobian from very sparse to dense.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations and Objectives of This Study

Steady state (harmonic balance) analysis of microwave circuits is the dominant approach to the simulation of microwave circuits. Development efforts are currently being directed at extending the approach to accommodate a very large number of tones, improve robustness, improve analysis speed and the development of matrix-free methods to handle very large problems without explicit storage of the Jacobian.

The analysis time and memory needed are mainly determined by the nonlinear equation solver. The Jacobian density increases with the nonlinearity of the circuit, that is, the Jacobian matrix has both dense and sparse structures for different circuits with various drive conditions. It can then be expected that different nonlinear equation solving methods are to be preferred for different circuits.

The aim of this work was to incorporate advanced numerical techniques, specifically, iterative matrix solve, in a simulator for the steady-state and transient moldeling of microwave spatial power combiners. Several developments of a preexisting nonlinear equation solver, the NNES package(Nonmonotonic Nonlinear Equation Solver) [2], were required as follows:

1. LU decomposition (for dense systems)

2. GMRES preconditioning method (for sparse systems)

3. Modified Broyden's Jacobian update method (a nonlinear solver method)

1.2 Thesis Overview

Chapter 2 reviews the nonlinear equation solver used in Harmonic Balance analysis.

Chapter 3 presents new methods and time analysis.

In Chapter 4 the results of new methods applied to moderately and strongly driven sparse and dense systems.

In the last Chapter the conclusions and the directions for future study are presented.

Chapter 2

Literature Review

2.1 Role of Nonlinear Equation Solver in Harmonic Balance Analysis

2.1.1 Summary of Transim

Transim is a general purpose circuit simulator developed at North Carolina State University. It provides netlist parsing, a library with many linear microwave elements, and output routines to display voltages, currents, state variables and operations among them.

In *Transim* the harmonic equations are formulated with the minimum number of unknowns starting from a modified nodal admittance matrix of the linear part of the circuit.

The system equation formulation begins with the partitioned network of Figure 2.1, with the nonlinear elements replaced by variable voltage or current sources. For each nonlinear element one terminal is taken as reference

Figure 2.1: Network with nonlinear elements.

and the element is replaced by a set of sources connected to the reference terminal. Clearly, the state of the element can be determined considering only the current of the sources, or the voltages across the sources, or some combination of the voltages and currents.

Using the *state variable* concept, let $\mathbf{X}$ be the *state variable vector*, and let $U_{NL,fi}(\mathbf{X})$ be the vector with the voltages of all the current sources due

to the nonlinear elements at frequency f_i (port voltages of the nonlinear elements). The error function is formulated by comparing the port voltages of the nonlinear elements $U_{NL,fi}(\mathbf{X})$ with the port voltages at the linear circuit $U_{L,fi}(\mathbf{X})$. Let us list the final equations in the frequency domain as follows. The reader is referred to [1] for a complete description.

$$F(\mathbf{X}) = \begin{bmatrix} S_{sv,f0} + \mathbf{M}_{sv,f0} I_{NL,f0}(\mathbf{X}) - U_{NL,f0}(\mathbf{X}) \\ S_{sv,f1} + \mathbf{M}_{sv,f1} I_{NL,f1}(\mathbf{X}) - U_{NL,f1}(\mathbf{X}) \\ \vdots \\ S_{sv,fm} + \mathbf{M}_{sv,fm} I_{NL,fm}(\mathbf{X}) - U_{NL,fm}(\mathbf{X}) \end{bmatrix} = \mathbf{0} \tag{2.1}$$

After solving the above nonlinear equation, the value of the state variable vector is known, so that the voltages (and the current of the ideal voltage sources) for the entire network can be found.

Figure 2.2 shows the flow diagram of Transim related to the harmonic balance analysis. Many options used by the NNES [2] are set in the input file netlist, most of them have default values. Note that the nonlinear equation solver knows nothing about harmonic balance. This fact allows us to integrate any general solver into the program.

Figure 2.2: General flow diagram of TRANSIM

2.1.2 Time and Memory Needed by Nonlinear Equation Solver

In order to solve the nonlinear equation, we need the Jacobian matrix, which requires the computer memory of N^2 words. The Jacobian need to be factored if we use a direct method to solve linear equations. the factorization time is $O(N^3)$, where N is the number of unknowns. When N is large, it will need huge memory and time. So the nonlinear equation solver has significant influence to HB analysis.

2.2 General Methods Used in Nonlinear Equation Solver

1. Bisection Method

2. Newton's Method (also known as Newton-Raphson method)
3. Secant Method
4. Muller's Method

Among these methods, Newton's method has the best convergence properties if the initial estimated solution is close to real solution. It is the most commonly used approach in HB simulation.

2.3 Newton-based Method

2.3.1 Introduction

We intend to solve nonlinear equations

$$F(x) = 0 \tag{2.2}$$

with a faster method, where $F \in R^n, x \in R^n$,

$$boundli \leq x(i) \leq boundui, i = 1, 2, ..., n \tag{2.3}$$

F is assumed to be continuously differentiable with Jacobian matrix J $\in R^{n*n}$,

$$J(i, j) = \frac{\partial F(i)}{\partial x(j)} \tag{2.4}$$

Newton's method is based upon a truncated Taylor series,

$$F(x) = F(x_k) + J(x_k)(x - x_k) \tag{2.5}$$

By defining Newton step

$$s = x - x_k, s \in R^n \tag{2.6}$$

We get the following iterative sequence, where k denotes the k^{th} iteration.

$$J_k s_k = -F(x_k) \tag{2.7}$$

$$x_{k+1} = x_k + s_k \tag{2.8}$$

The iteration will stop when meet either step size criterion or function value criterion or both, according to the user defined parameter.

2.3.2 Methods Used in Solving Linear Equations

Two types of methods can be used to solve linear equation 2.7. One is direct method which is based on Gauss elimination, one is iterative method which use successive approximation to obtain an accurate solution of a linear system, according to some criteria. Iterative method has been developed historically to provide us with ways to solve large sparse problems, and it can find a reasonable approximate solution quickly (in far fewer elemental operations than the direct methods) for well conditioned systems.

The direct method can get exact solution of linear equations (assume no computation accuracy lost), and iterative method can only get inexact solution of linear equations(only reach some accuracy criterion). It is also called *Inexact-Newton's Method* when iterative methods are used in Newton's method. Inexact-Newton's method has some advantages: we can get Newton step with low accuracy when the solution is far from the real solution and establish high accuracy only for the final nonlinear iterations, thus it can use less time while the the final solution is remain exact. In general, GMRES is the best method among iterative methods in circuit problem solver.

1. Direct method

   Gauss Elimination(LU factorization)

   QR factorization

   $$J_k = Q_k R_k \tag{2.9}$$

   where $Q_k Q_k^T = I, R_k$ is an upper triangular matrix.

   So equation 2.7 becomes to

   $$Q_k(R_k s_k) = -F(x_k) \tag{2.10}$$

   That is,

   $$z_k = -Q_k^T F(x_k) \tag{2.11}$$

   $$R_k s_k = z_k \tag{2.12}$$

   The Householder transformation is considered to be desirable among the Givens transformation and modified Gram-Schmidt method due to its computational efficiency and robust numerical stability. By using the Householder transformation, Equation 2.9 needs $O(\frac{10}{3}n^3)$ operations. Solving 2.10 needs $O(3n^2)$ operations.

2. Iterative method

   Jacobian method

   Gauss-Seidel method

   Successive Over-Relaxation(SOR) method

   Symmetric SOR (SSOR) method

   Conjugate Gradient (CG) method

   MINRES and SYMMLQ method

   Generalized Minimal Residual (GMRES) method

   Bi-Conjugate Gradient (BiCG) method

   Quasi-Minimal Residual (QMR) method

   Squared CG(CGS) method

2.3.3 Methods Used in Jacobian Update

The Jacobian matrix J_{k+1} can be evaluated explicitly in $k+1^{th}$ iteration, but it is expensive. Several quasi-Newton update methods have been proposed to estimate the J_{k+1} by a matrix B_{k+1}. Broyden's method is one of them. It uses equation 2.13 to get B_{k+1}. For QR factored B_k 2.13 needs $O(29n^2)$ operations to get QR factored B_{k+1} by using rank one update method. The Newton's method with Broyden Jacobian update method is shown as follows.

set $x_0 = 0$;
(s1). evaluate $B_0 = J_0$;
for k=0,1,...until satisfy stop criterion do
begin
$B_k s_k = -F(x_k)$; Obtain s_k by using QR decomposition
$x_{k+1} = x_k + s_k$; Obtain x_{k+1}
Evaluate $F(x_{k+1})$;
If meet stop criterion then stop
If the results have not been improved much during several iterations, then update $x_0 = x_k$, goto (s1)
$Y_k = F(x_{k+1}) - F(x_k)$;
get B_{k+1} from equation 2.13
end

$$B_{k+1} = B_k + \frac{(Y_k - B_k s_k)s_k^T}{s_k^T s_k} = B_k + \frac{F(x_{k+1})s_k^T}{s_k^T s_k} \text{ [3]} \tag{2.13}$$

The reason why we need to explicitly reevaluate Jacobian is that the estimated Jacobian can become inaccurate after several iterations.

The disadvantage of using Broyden's method instead of using explicitly evaluated Jacobian at each iteration is that the quadratic convergence of Newton's method is lost, being replaced by a convergence called superlinear. In most applications, the reduction to superlinear convergence is a more than acceptable trade-off for the decrease in the amount of computation.

2.4 Summary of NNES

NNES (Nonmonotonic Nonlinear Equation Solver) is written in FORTRAN and it provides Newton and quasi-Newton methods using QR decomposition(using the Householder transformation). The major drawback of Newton's method is that it is not globally convergent so, in practice, it fails too often to be a useful general method. Two global methods, Line Search method and Trust Region method, are introduced in NNES. According to how to select S_k obtained from equation 2.7, NNES has four different methods:

1. Plain Newton's method
   Directly use s_k obtained from equation 2.7

2. Absolute Newton's method
   If x_{k+1} in equation 2.8 does not satisfy equation 2.3, the amount of the violation is reflected to obtain the shorter s_k .

3. Line Search method
   Use Armijo criterion for step acceptation.
   Define an objective function

   $$f = \frac{1}{2}F(x_k)^T F(x_k) \tag{2.14}$$

   If

   $$f(x_k + \lambda d_k) \leq f(x_k) + \alpha\lambda \bigtriangledown f(x_k)^T d_k \tag{2.15}$$

accept λd_k as Newton step s_k, where $d_k = -B_k^{-1}F(x_k)$ is Newton direction.
Else modify λ , where $\alpha \in (0,1)$, is a constant , $\lambda \in (0,1)$, is the relaxation factor .

4. Trust Region method
   The trust region [4] method attempts to combine the better global convergence properties of searches in the steepest descent direction with the local quadratic convergence of the Newton method. It is similar to the well-known Levenberg-Marquardt method for nonlinear least squares.

NNES can handle singular Jacobians or ill conditioned Jacobians.

The Jacobian matrix can be evaluated explicitly in one of four ways: analytically; forward differences; backward differences and central differences. Since all of the methods are expensive, NNES also uses 2 quasi-Newton update methods to get approximate Jacobian: Broyden's method and Lee & Lee's method. Both of the methods can update QR factored and unfactored Jacobian matrix.

Chapter 3

Technical Contribution

Several free software packages used here are listed in Table 3.1.

3.1 LU decomposition

$$J_k = L_k U_k \tag{3.1}$$

where L_k is a lower triangular matrix, U_k is an upper triangular matrix, So equation 2.7 becomes to

$$L_k(U_k s_k) = -F(x_k) \tag{3.2}$$

That is :

$$L_k z_k = -F(x_k) \tag{3.3}$$

$$U_k s_k = z_k \tag{3.4}$$

Here we apply *partial pivoting* strategy (row interchanges) in LU decomposition, that is, choose pivot element as the largest element in magnitude of the

Sparse	Sparse matrix library
NNES	Nonmonotonic Nonlinear Equation solver
Adol-c	Automatic differentiation package
SLATEC	Common mathematical library [1]
Gnuplot	Plotting utility

Table 3.1: Software packages used in Transim

leftmost possible column in the remaining submatrix. Equation 3.1 requires $O(\frac{2}{3}n^3)$ operations and $O(n^2)$ comparisons. Solving 3.2 requires $O(2n^2)$ operations.

The advantages of using LU factorization are :

1. LU factorization uses approximately one fourth of the operations QR factorization used.

2. It is faster to solve LU factored linear equation than to solve QR factored linear equation.

There are also two drawbacks of using LU factorization:

1. The ability of handling ill-conditioned Jacobian is weak.

2. Update LU factored Jacobian is difficult.

For the first drawback, it is not a big problem for circuit simulation. From our experiments, whenever QR method converges, LU method converge. In addition, we can use scaling factor to reduce condition number.

For the second drawback, we introduced modified Broyden's method.

3.2 Modified Broyden Jacobian Update Method

Some methods can be used to get QR factorization of B_{k+1} directly by updating QR factored B_k, avoiding re-factor B_{k+1}, such that reduce the operation from $O(n^3)$ to $O(n^2)$ flops.

But it is very difficult to update LU factored Jacobian matrix because of pivoting requirements and when we tamper with a positive definite matrix the result may not be positive definite [5]. Here we introduce modified Broyden method which allows us to obtain Newton step without the information of current Jacobian matrix.

set $x_0 = 0$;

(s1). evaluate $B_0 = J_0$;

[1]SLATEC is the acronym for the Sandia, Los Alamos, Air Force Weapons Laboratory Technical Exchange Committee

$B_0 s_0 = -F(x_0)$; Obtain s_0 by using LU decomposition or GMRES solver

for k=0,1,2,..until satisfy stop criterion do

begin

$x_{k+1} = x_k + s_k$; Obtain x_{k+1}

Evaluate $F(x_{k+1})$;

If meet stop criterion then stop

If the results have not been improved much during several iterations, then update $x_0 = x_k$, go to (s1)

get s_{k+1} from the equation 3.8. Note: $B_0^{-1}F(x_{k+1})$ is obtained from LU or GMRES method.

end

By using equation 2.13 and Sherman-Morrison Formula,

$$(B + uv^T)^{-1} = (I - \frac{(B^{-1}u)v^T}{1 + v^T B^{-1}u})B^{-1} \tag{3.5}$$

Let $u_k = \frac{F(x_{k+1})}{\|s_k\|_2}, v_k = \frac{s_k}{\|s_k\|_2}$, we can deduce :

$$B_k^{-1} = \prod_{j=0}^{k-1}(I + \frac{s_{j+1}s_j^T}{\|s_j\|_2^2})B_0^{-1} \text{ [7]} \tag{3.6}$$

By using $s_{k+1} = -B_{k+1}^{-1}F(x_{k+1})$, we can deduce:

$$s_{k+1} = \frac{-\|s_k\|_2^2 B_k^{-1}F(x_{k+1})}{\|s_k\|_2^2 + s_k^T B_k^{-1}F(x_{k+1})} \tag{3.7}$$

where

$$B_k^{-1} = \prod_{j=0}^{k-1}(I + s_{j+1}s_j^T/\|s_j\|_2^2)B_0^{-1} \tag{3.8}$$

for $k \geq 1$, with End Conditions

$$s_0 = -B_0^{-1}F(x_0) \tag{3.9}$$

We can see, we only need B_0 to get Newton step, thus during all the iterations we only need to factorize B_0 once until the program need to get next new

B_0. So actually we do not need to factorize or update B_{k+1}, and this is a time-saving step.

The operations for calculating s_{k+1} are about $O(2n^2 + 6kn)$ by using the following algorithm:

solve z from $B_0 z = -F(x_{k+1})$
for $j = 0, k-1$
$z = z + s_{j+1}(s_j^T z)/\|s_j\|_2^2$
$s_{k+1} = \frac{z}{1+s_k^T z/\|s_k\|_2^2}$

3.3 Line Search Strategy

Using $\bigtriangledown f(x_k) = J^T(x_k)F(x_k)$ we deduce
$\bigtriangledown f(x_k)^T d_k = F^T(x_k)J(x_k)d_k = F^T(x_k)(-F(x_k)) = -2f(x_k)$
substitute it to equation 2.15, we get

$$f(x_k + \lambda d_k) \leq (1 - 2\alpha\lambda)f(x_k) \tag{3.10}$$

Because Newton step is no longer d_k, the formula for Newton direction in 3.8 has to be changed.

Let $u_k = \frac{F(x_{k+1})}{\|s_k\|_2}, v_k = \frac{s_k}{\|s_k\|_2}$, $d_k = -B_k^{-1}F(x_k)$, and $s_k = -\lambda_k d_k$, we can deduce:

$$d_{k+1} = \frac{(-\|s_k\|_2^2 + (1-\lambda_k)s_k s_k^T)B_k^{-1}F(x_{k+1})}{\|s_k\|_2^2 + \lambda_k s_k B_k^{-1}F(x_{k+1})} \tag{3.11}$$

where

$$B_k^{-1} = \prod_{j=0}^{k-1}(I + (\frac{\lambda_j}{\lambda_j + 1}s_{j+1} + (\lambda_j - 1)s_j)s_j^T/\|s_j\|_2^2)B_0^{-1} \tag{3.12}$$

3.4 GMRES method

3.4.1 Introduction

We intend to solve

$$Ax = b \tag{3.13}$$

where $A = (a_{i,j})$, is a Jacobian matrix, and $x = (x_i), b = (b_i)$, i, j =1, 2, ..., n.

GMRES was proposed in 1986 as a Krylov subspace method for nonsymmetric systems. We use the most popular form of GMRES, which is based on the modified Gram-Schmidt procedure and uses restarts to control storage requirements [9]. GMRES will stop when $\frac{\|b-Ax\|}{\|b\|} < \epsilon$ for right preconditioning, where ϵ is the error tolerance for linear equation. In practice, ϵ is in the range of [0.01,0.5].

3.4.2 Preconditioner

The convergence rate of GMRES method (and other iterative methods) depends greatly on the spectrum of matrix A. We know condition number $\kappa(A) = \|A\|\|A^{-1}\|$ and the 2-norm $\kappa(A)$ is the ratio of the largest singular value of A to the smallest. Error for the GMRES is a function of $\kappa(A)$, so we hope $\kappa(A)$ is as small as possible, that is, as close to 1 as possible, because the smallest κ is 1. A preconditioner M is a matrix which can transform 3.13 into one that is equivalent in the sense that it has the same solution, but has more favorable spectrum. In fact, the iterative method may even fail to converge without a preconditioner or with a bad preconditioner. A good preconditioner improves the convergence, sufficiently to overcome the extra cost of constructing and applying the preconditioner.

By applying preconditioner, equation 3.13 changed to

$$(AM^{-1})(Mx) = b \tag{3.14}$$

with right preconditioning , or

$$(M^{-1}A)x = M^{-1}b \tag{3.15}$$

with left preconditioning

A : use dense format

M : use sparse coordinate format (COO) [8]

No matter which preconditioner we use, we must calculate $M^{-1}b$, so M should be chosen in such a way that it is as close as possible to A such that $\kappa(M^{-1}A) \approx 1$ and M^{-1} can be evaluated or approximated by some computationally fast technique.

ILU(0)

Level 0 Incomplete LU (ILU) Factorization Preconditioner.

Apply ILU to A, we get M

$$A = M + E, \quad M = LD^{-1}U \tag{3.16}$$

with D diagonal, L strict lower triangular, U strict upper triangular and E error matrices

During the completed factorization process($\|E\| = 0$) the additional nonzeros will be generated such that L, D and U have less sparsity than A. Level 0 means the additional nonzeros will be discarded so that L + D + U has at least the same sparsity structure as A. Obviously, the less the value of the norm of E, the more accuracy M approximate A, the more density the M and also the more computational effort needed to factorize M.

ILUS(0)

Level 0 Incomplete Sparse LU (ILUS) Factorization Preconditioner.

$$A = S + E_1, \quad S = M + E, \quad M = LD^{-1}U \tag{3.17}$$

This preconditioner is similar to ILU(0) except M is obtained from sparse matrix S which is extracted from A by following some criterions. First consider the Relative Value Criterion, that is, if $|A(i,j)| \leq \alpha * min(|A(i,i)|, |A(j,j)|)$ discard S(i,j), where $0 \leq \alpha \leq 1E-3$. This kind of preconditioner takes the advantage of A is diagonal dominant.

Table 3.2 shows $\kappa(M^{-1}A)$ that corresponds to different α in a soliton lines circuit with 4 state variables during the first iteration of Newton's method. Figure 3.1 gives the structures of extracted matrix S corresponds to Table 3.2. When $\alpha = 0$, S=A and the sparsity of A is 34%. When $\alpha = 1E-3$, the sparsity of S is reduced to 3% while $\kappa(M^{-1}A) = 1$. Therefore using S to construct M will save a lot of factorization time.

Table 3.3 shows condition numbers of incomplete and complete LU that correspond to different α in a soliton lines circuit with 4 state variables during the sixteen iteration of Newton's method. For this particular problem, α = 1E-4 is good because it reduced nonzeros at least one third and $\kappa(M^{-1}A)$ is still close to 1. We noticed $M^{-1}A$ in (f) is ill-conditioned. We also noticed even when nonzeros = 7163, S approximate A pretty well($\kappa(S^{-1}A) = 1.23$). But with incomplete LU, M is still far away from A($\kappa(M^{-1}A) = 1557.98$). What makes the big difference? Fill-ins. ILU(0) dropped some fill-ins with large magnitude. Figure 3.2 and figure 3.3 give the structures of extracted

No.	α	Nonzeros	$\kappa(M^{-1}A)$
(a)	0	21834	1.00
(b)	1E-5	1988	1.00
(c)	1E-4	1988	1.00
(d)	1E-3	1956	1.00
(e)	1E-2	1862	1.03
(f)	1E-1	1478	83.52

Table 3.2: Condition numbers in the 1st iteration

No.	α	Nonzeros in S and ILU of S	$\kappa(M^{-1}A)$	Nonzeros in LU of S	$\kappa(S^{-1}A)$
(a)	0	63504	1.00	63504	1.00
(b)	1E-5	30981	1.13	41841	1.00
(c)	1E-4	20534	1.68	33693	1.00
(d)	1E-3	12804	58.68	21507	1.02
(e)	1E-2	7163	1557.98	11982	1.23
(f)	1E-1	2824	7E+19	5321	6.29

Table 3.3: Condition numbers in the 16th iteration

matrix S and complete LU of S correspond to Table 3.3. There are several methods to partially alleviate the disadvantage. The first method is to use higher level ILU. This makes difficult to predict the nonzeros and allocate memory. The second method is to use reordering methods, for example, Cuthill-McKee and minimum degree algorithms. Reordering methods are originally devised for symmetric positive definite systems to reduce fill-in in the factorization. With proper modification, they can also be used in non-symmetric systems. But experiments show reordering methods are ineffective to diagonally dominant matrices [17].

Since the construction of efficient general purpose preconditioner is not possible, we use the combination of several criterions to choose S.

1. Band Criterion
   All nonzeros in the diagonal zone with width of $BANDWD * n$ will be

Figure 3.1: Preconditioner structures in the 1st iteration

Figure 3.2: Preconditioner structures in the 16th iteration

Figure 3.3: Complete LU structures of S in the 16th iteration

retrived. $BANDWD = [0.0, 1.0]$.

2. Relative Value Criterion
   If $|A(i,j)| \leq \alpha * min(|A(i,i)|, |A(j,j)|)$ then discard S(i,j). $\alpha = [0.0, 1.0E-3]$.

3. Absolute Value Criterion
   If $|A(i,j)| \leq LLIMIT$ then discard S(i,j). $LLIMIT = [0.0, 1.0]$.

4. Zero Row Summation Criterion
   If ZSJRSM=true then diagonal elements of S are adjusted so that A-S has zero row summations. For many matrices, this requirement produces a better condition number.

The choice of criterions was problem and architecture-dependent and based on the numerical experiments. For example, when $BANDWD = 0.3, \alpha = 1E-3$, $LLIMIT = 1E-3$ and $ZSJRSM = true$, there are 17819 nonzeros in S and $\kappa(M^{-1}A) = 1.30$. It is better than just using $\alpha = 1E-4$ alone, in that case there are 20534 nonzeros and $\kappa(M^{-1}A) = 1.68$. When $BANDWD = 0, \alpha = 0$ and $LLIMIT = 0$, ILUS(0) reduces to ILU(0).

Block Jacobian

Approximation of the inverse for A.
M^* = Approximation of A^{-1}

$$M = (M^*)^{-1} \tag{3.18}$$

3.4.3 Implementation

Preconditioner GMRES Algorithm:

$x^{(0)}$ is an initial solution

for j=1, 2, ..., ITMAX // ITMAX is the maximum number of restarts

Solve r from $Mr = b - Ax^{(0)}$

$v^{(1)} = r/\|r\|_2$

$s := \|r\|_2 e_1$

for i=1, 2, ..., m // m is dimensions of Krylov subspace

Solve w from $Mw = Av^{(i)}$ // for left preconditioner
OR from $w = Av^{(i)}$ // for right preconditioner

for k=1,...,i
$h_{k,i} = (w, v^{(k)})$
$w = w - h_{k,i}v^{(k)}$

end

$h_{i+1,i} = \|w\|_2$

$v^{(i+1)} = w/h_{i+1,i}$

Construct J_i such that
$[J_i(h_{1,i}, h_{2,i}, ..., h_{i+1,i}, 0, ..., 0)^T]_{i+1} = 0$
$s := J_i s$

UPDATE($\tilde{x}, i$)

end

$UPDATE(\tilde{x}, m)$

if convergent, quit

end

UPDATE($\tilde{x}, i$) Algorithm:

$\tilde{x} = x^{(0)} + y_1 v^{(1)} + y_2 v^{(2)} + ... + y_i v^{(i)}$

$s^{(i+1)} = \|b - A\tilde{x}\|_2$

if $s^{(i+1)}$ small enough then quit

else $x^{(0)} = \tilde{x}$

Where H is a $m * m$ upper Hessenberg matrix,

and $(H)_{i,j} = h_{i,j}$ if $i \leq j$
0 otherwise

$\tilde{s} = (s_1, s_2, ..., s_i)^T$

$Hy = \tilde{s}$

From above algorithms we can see no matter which preconditioner we select, GMRES must solve 2 equations

$$Mr = b - Ax \tag{3.19}$$

$$Mw = Av \tag{3.20}$$

for left preconditioning

$$w = Av \tag{3.21}$$

for right preconditioning
where r, w, v$\in R^n$. So we introduce 2 subroutines MATVEC and MSOLVE.

MATVEC : The routine performs the matrix vector multiply y $= A * x$ given A and x. A can be either dense format or sparse format.

MSOLVE : The routine solves a linear system Mz = r for z given r with the preconditioning matrix M. M use sparse format.

We also noticed that the Jacobian matrix A is used only by MATVEC provided we have M.

3.5 Time Analysis

Table 3.4 lists CPU time estimate when line search method is not conducted. When $d > 54\%$, we have $\frac{7}{3}d^2n^3 > \frac{2}{3}n^3$. This implies that when the density of preconditioner is greater than 54%, by no means the GMRES method will faster than the LU method. Figure 3.4 lists the relationship between CPU time and Jacobian density in our single tone problem with 47 diodes, 20 frequencies and 1833 system dimensions.

For GMRES method, the total time is related to the density and pattern of Jacobian matrix, so here we only compare QR and LU methods.

Time spent in QR method:

I_qF	(function evaluation)
$+ E_qJ$	(explicitly Jacobian evaluation)
$+ \frac{10}{3}n^3tE_q$	(QR factorization)

Figure 3.4: CPU time spent in QR, LU and ILU decomposition for the circuit of Figure 4.2

$$
\begin{aligned}
&+ 29n^2t(I_q - E_q) && \text{(update factored Jacobian)}\\
&+ 3n^2tI_q && \text{(solve factored Jacobian)}
\end{aligned}
$$

Time spent in LU method:

$$
\begin{aligned}
&I_uF && \text{(function evaluation)}\\
&+ E_uJ && \text{(explicitly Jacobian evaluation)}\\
&+ \tfrac{2}{3}n^3tE_u && \text{(LU factorization)}\\
&+ (2n^2 + 3Kn)tI_u && \text{(update Newton step)}
\end{aligned}
$$

So the ratio of time spent in LU and QR method is,

$$R = \frac{LU\_Time}{QR\_Time} \approx \frac{(\frac{2}{3}n^3t + J)E_u + (2n^2t + 3Knt + F)I_u}{(\frac{10}{3}n^3t + J)E_q + (32n^2t + F)I_q} \tag{3.22}$$

In general, K<30, $F << J <$ factorization time. For large n, Jacobian factorization/get preconditioner represents a major contribution to the overall CPU time. From 3.22 we see the ratio is greatly depend on E_u and E_q.

[1]t is the CPU time spent in one FLOP (Flopping Point Operation, including addition/subtraction and multiplication/division), and we omitted O(n^2t) FLOPs here.

[2]Partial pivoting requires $O(n^2)$ additional comparisons.

[3]NZ is the total number of nonzeros in the preconditioner and d is the density of the preconditioner.

[4]Related to the condition number of preconditioned matrix and error tolerance for GMRES iteration.

[5]k is the number of iterations since last explicitly Jacobian evaluation. K is the number of iterations between two explicitly Jacobian evaluations.Typically K=[3,20]

No.	Item	QR with Broyden method	LU with modified Broyden method	GMRES with modified Broyden method
1	Evaluate functions	F		
2	Evaluate Jacobian matrix	J		
3.1	Factorize Jacobian matrix	$\frac{10}{3}n^3t$ [1]	$\frac{2}{3}n^3t$ [2]	-
3.2	Get preconditioner	-	-	$\frac{7(nz)^2}{3n}t = \frac{7}{3}d^2n^3t$ [3]
4.1	Solve factored linear equation	$3n^2t$	$2n^2t$	-
4.2	Solve preconditioned GMRES	-	-	S_gt [4]
5	Get Newton step in Jacobian updated iteration	$(29n^2 + 3n^2)t$	$(2n^2 + 6kn)t$ [5]	$(S_g + 6kn)t$
6.1	Total time per iteration when Jacobian evaluated	item 1 + 2 +3.1 + 4.1	item 1 + 2 +3.1 + 4.1	item 1 + 2 + 3.2 + 4.2
6.2	Total time per iteration when Jacobian updated	item 1+5		
7.1	Number of total iterations	I_q	I_u	I_g
7.2	Number of Jacobian evaluations	E_q	E_u	E_g

Table 3.4: Time Estimate

Chapter 4

Results

All the results of tests are performed on a SUN UltraSPARC workstation with 512 MB central memory, starting from zero harmonics. The tolerance is set to 10^{-08}. The netlists we used in this chapter are shown in Appendix D.

4.1 Sparse Jacobian Matrix Case

4.1.1 Fullgrid [10]

The layout for the system we consider here is shown in Figure 4.1. Table 4.1 shows the test results.

4.1.2 Soliton

The layout for the system we consider here is shown in Figure 4.2. GMRES denotes GMRES method using ILU preconditioner. GMRES-C denotes GMRES method using cut-off ILU preconditioner, with BANDWD = 0.5, LLIMIT=$1.0 * 10^{-5}$. See Appendix A for explanation. GMRES-D denotes GMRES method using cut-off DILU preconditioner, with α =5.0E-5. Table 4.2 and Figure 4.3 show the test results.

Figure 4.1: Layout for 2x2 quasi-optical grid amplifier with bias inductors

Number of frequencies	16
Oversample	2
Number of state variables	32
System dimension n	992
Maximum Jacobian Density	26%
Average Jacobian Density	26%

No.	Item	QR with Broyden method	LU with Modified Broyden method	GMRES with Modified Broyden method
1	Evaluate functions (second)	0.19-0.47		
2	Evaluate Jacobian matrix (s)	17.34-18.45		
3.1	Factorize Jacobian matrix (s)	115.38	32.12	x
3.2	Get preconditioner (s)	x	x	10.51
4.1	Solve factored linear equation (s)	0.41-0.47	0.09-0.10	x
4.2	Solve preconditioned GMRES (s)	x	x	0.13-0.18
5	Get Newton step in Jacobian updated iteration (s)	2.23-2.27	0.10-0.11	0.20-0.21
6.1	Total time per iteration when Jacobian evaluated (s)	143.67	50.45	36.70
6.2	Total time per iteration when Jacobian updated (s)	7.64-7.85	0.29-0.31	0.40-0.42
7.1	Number of total iteration	12	12	12
7.2	Number of function evaluation	13	13	13
7.3	Number of jacobian evaluation	1	1	1
8	Simulation time (second)	224.75	54.52	41.78

Table 4.1: Comparison of different methods used in fullgrid circuit

Figure 4.2: Layout for Soliton Lines Circuit

Figure 4.3: Soliton Line Circuit Analysis: Sparse Case

Number of frequencies		32			
Oversample		2			
Number of state variables		4	10	20	30
System dimension n		252	630	1260	1890
Maximum Jacobian Density		100%	100%	100%	100%
Average Jacobian Density		67%	89%	78%	67%
Maximum Preconditioner Density (GMRES-C)		57%	70%	66%	51%
Maximum Preconditioner Density (GMRES-D)		37%	57%	56%	20%
Average Preconditioner Density (GMRES-C)		38%	58%	47%	35%
Average Preconditioner Density (GMRES-D)		20%	45%	34%	12%
Number of Total Iterations	QR	11	82	33	14
	LU	16	28	26	20
	GMRES	16	29	26	20
	GMRES-C	16	29	26	20
	GMRES-D	16	29	26	21
Number of Jacobian Evaluation	QR	2	19	5	2
	LU	2	6	3	2
	GMRES	2	6	3	2
	GMRES-C	2	6	3	2
	GMRES-D	2	6	3	2
Factorization or ILU Time (second)	QR	1.45-1.48	28.05-28.12	239.40-239.63	871.10-871.90
	LU	0.40-0.41	7.86-7.87	66.29-66.43	236.73-238.12
	GMRES	0.23-1.52	3.67-26.01	30.96-214.79	106.19-736.88
	GMRES-C	0.04-0.47	0.62-12.67	5.10-94.60	17.63-130.44
	GMRES-D	0.01-0.20	0.03-9.07	0.20-76.08	0.63-28.37
Total Time (second)	QR	29	1332	1974	2434
	LU	29	256	430	708
	GMRES	31	415	937	1531
	GMRES-C	29	303	506	569
	GMRES-D	29	280	420	312

Table 4.2: Comparison of different methods used in sparse case of Soliton Lines Circuit

Number of frequencies		32			
Oversample		2			
Number of state variables		4	10	20	30
System dimension n		252	630	1260	1890
Maximum Jacobian Density		100%	100%	100%	100%
Average Jacobian Density		97%	98%	99%	98%
Number of Total Iteration	QR	136	140	171	108
	LU	92	137	202	154
	GMRES	926	137	202	154
Number of Jacobian Evaluation	QR	12	14	17	23
	LU	20	34	52	38
	GMRES	20	34	52	38
Factorization or ILU Time (second)	QR	1.47-1.50	28.13-28.58	239.45-241.75	870.60-872.53
	LU	0.41-0.42	7.86-7.95	66.45-66.85	235.06-237.21
	GMRES	0.06-1.57	0.88-26.31	7.30-239.84	106.53-737.00
Total Time (second)	QR	216	1403	9300	28627
	LU	286	1737	8057	14863
	GMRES	325	2578	20573	45782

Table 4.3: Comparison of different methods used in Dense case of Soliton Lines Circuit

4.2 Dense Jacobian Matrix Case

4.2.1 Soliton

Table 4.3 and Figure 4.4 show the test results.

Figure 4.4: Soliton Line Circuit Analysis : Dense Case

Chapter 5

Conclusions and Future Research

5.1 Conclusions

Which method is the best depends on the system dimension, Jacobian density and Jacobian pattern. In general, when the Jacobian is a sparse matrix, GMRES is the best method. When the system dimension is small and Jacobian is not sparse, QR is the best method. Otherwise, LU decomposition is the best method.

5.2 Future Research

1. Find the relation between the error tolerance in GMRES and the objective function value.

2. Using Matrix Free Method to decrease the memory needed by GMRES.

   From the GMRES algorithm we know Jacobian matrix is used only in Jacobian-vector product and constructing preconditioner M. So if we can get M and Jacobian-vector product without Jacobian matrix, many memory will be saved. Using truncated Taylor series

$$F(X_k + \delta V) = F(X_k) + J(X_k)(X_k + \delta V - X_k) \tag{5.1}$$

   so

$$J(X_k)V = (F(X_k + \delta V) - F(X_k))/\delta \tag{5.2}$$

The difficulties are selecting appropriate δ and constructing M without Jacobian.

3. For strongly nonlinear circuits, use Jacobian inversion method.

Bibliography

[1] C. E. Christoffersen, M. B. Steer, M. A. Summers, Harmonic Balance Analysis for Systems with Circuit-Field Iterations. *Proc. 1998 IEEE Int. Microwave Symp.*,Vol. 2, 1131-1134, 1998.

[2] R. S. Bain, *NNES User's Manual*, 1993.

[3] C. G. Broyden, A class of methods for solving nonlinear simultaneous equations, *Math. Comput.*, 19, 577-593, 1965.

[4] M. J. D. Powell, A hybrid method for nonlinear equations, *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, Editor, Gordon and Breach, 1988.

[5] G. H. Golub, Charles F. Van Loan, *Matrix computations*, Baltimore : Johns Hopkins University Press, 1996.

[6] J. E. Dennis, Jr., Robert B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Englewood Cliffs, N.J. : Prentice-Hall, 1983.

[7] C. T. Kelley, *Iterative methods for linear and nonlinear equations*, Philadelphia : Society for Industrial and Applied Mathematics, 1995.

[8] Christoph W. Ueberhuber, *Numerical computation : methods, software, and analysis*, Berlin ; New York : Springer, 1997.

[9] Richard Barrett, Micheal Berry, Tony F. Chan, James Demmel, June M. Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, Henk Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.*

[10] M. A. Summers, Simulation of a quasi-optical grid amplifier, M.S. Thesis, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, North Carolina, USA, 1997.

[11] W. H. Press, B. P. Flannery, *Numerical recipes in C: the art of scientific computing*, Cambridge [Cambridgeshire] ; New York : Cambridge University Press, 1992.

[12] V. Rizzoli, F. Mastri, F. Sgallari, G. Spaletta, Harmonic-Balance Simulation of Strongly Nonlinear very Large-Size Microwave Circuits by Inexact Newton Methods, *Proc. 1996 IEEE Int. Microwave Symp.*, 1996.

[13] I. Moret, On the Convergence of Inexact Quasi-Newton Methods, *Int. J. of Computer Math.*, Vol. 28, 117-137, 1987.

[14] W. D. McQuain, C. J. Ribbens, L. T. Watson, R. C. Melville, Preconditioned Iterative Methods for Sparse Linear Algebra problems Arising in Circuit Simulation, *Computers Math. Applic.*, Vol. 27, No. 8, 25-45, 1994.

[15] C. E. Christoffersen, State variable harmonic balance simulation of a quasi-optical power combining system, M.S. Thesis, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, North Carolina, USA, 1997.

[16] R. Telichevesky, K. Kundert, I. Elfadel, J. White, Fast Simulation Algorithms for RF Circuits, *IEEE 1996 Custom Integrated Circuits Conference.*, 437-444,1996.

[17] M. Benzi, D. B. Szyld, A. V. Duin, Orderings for Incomplete Factorization Preconditioning of Nonsymmetric Problems, *SIAM J. SCI. COMPUT.*, Vol.20, No.5, 1652-1670, 1999.

Appendix A

User's Guide

A.1 Environment

1. Need g77 - GNU project F77 Compiler and gcc - GNU project C and C++ Compiler

2. Suppose current working directory is TRANSIM

3. Suppose the netlist file is NETLIST in TRANSIM directory

4. Suppose the erl Locker is ERL

5. Run the shell command >unlimit stacksize

6. Check out source files from ERL to TRANSIM by using the following commands

   (a) >add erl

   (b) >add rcstools

   (c) >add gnu

   (d) >cd TRANSIM

   (e) >tco Makefile

   (f) >make createdirs

A.2 Steps of using QR, LU or GMRES methods

1. Set up parameters in NETLIST
   Table A.1 shows the possible optionals of different methods, where x indicates it's possible and o indicates it's the default value.
   Explanation of parameters:
   (a) Method

   Method = 1: Plain Newton's method

   Method = 2: Absolute Newton's method

   Method = 3: Newton's method with Line Searches method

   Method = 4: Newton's method with Trust Region method

   (b) JUPDM

   Determines whether the Jacobian is to be evaluated explicitly or to be updated via a quasi-Newton method.

   JUPDM = 0 Jacobian evaluated explicitly (see jactyp for different options)

   JUPDM = 1: Broyden update

   JUPDM = 2: LEE and LEE update

   JUPDM = 3: LU Decomposition + Modified Broyden's update(Sherman-Morrison Method)

   JUPDM = 4: GMRES method + Modified Broyden's update(Sherman-Morrison Method)

   (c) QNUPDM

   Determines how the quasi-Newton update is done.

   QNUPDM = 0: update unfactored Jacobian

   QNUPDM = 1: update factored Jacobian
   QNUPDM does not make sense for LU and GMRES method.

2. Set up parameters used by GMRES in setup.f at directory TRANSIM/nnes_local

	Method				JUPDM					QNUPDM	
	1	2	3	4	0	1	2	3	4	0	1
QR	x	x	o	x	o	x	o	x	x	x	o
LU	x	x	o		x			o			
GMRES	x	x	o						o		

Table A.1: Options used by different methods

(a) PRETYP

Preconditioner type used in GMRES method
= 2 : Jacobian Inverse
= other : ILU, you need to set up BANDWD and LLIMIT
default : 1

(b) BANDWD

Width of diagonal zone in retriving Jacobian matrix used for preconditioner
value : [0.0,1.0]
default value 0.0 : do not use band property
The larger the value, the slower the preconditioner calculation speed, the fewer the iterations needed in solving GMRES

(c) LLIMIT

Low limit absolute value of off-diagonal zone elements treated as non-zero elements in retriving Jacobian matrix used for preconditioner
value : [0.0,1.0]
default value 0.0 : use whole Jacobian matrix

(d) ALPHAP

Low limit relative parameter of off-diagonal zone elements treated as non-zero elements in retriving Jacobian matrix used for preconditioner
value : [0.0,1.0E-3]
default value 0.0 : use whole Jacobian matrix

(e) ZSJRSM

Whether the retrived matrix and the original matrix has the same row sums
value : true, false
default value true: Row sums of S - A = 0, S is the retrived sparse matrix from A. That is, all the discarded elements in A are added to the corresponded diagonal element in S.

Generally, when adjusting BANDWD, LLIMIT and ALPHAP, we must keep the target percentage (none-zeros after adjusting/non-zeros before adjusting) greater than a lower bound (50%, for example) in order to get convergence.

(f) TOLLOW
Lower limit of convergence criterion TOL in GMRES . TOLLOW <= TOLUP
value : [0.001,0.1]
default value : 0.01
The lower the value,the more the time spent in each iteration, the more the total iterations needed .

(g) TOLUP
Upper limit of convergence criterion TOL in GMRES .
value : [0.001,0.1]
default value : 0.01
TOL = BNRM if TOLLOW <= BNRM <= TOLUP
The lower the value,the more the time spent in each iteration, the more the total iterations needed.

(h) MAXL
Maximum dimension of Krylov subspace.
Value : [1,50]
Default value : 40

(i) ITMAX
MAXIMUM number of iterations in GMRES.
The actual maximum number of iterations = MAXL*(ITMAX/MAXL

+1)
if ITMAX < 0,the actual maximum number = MAXL
Default : 40

3. Create tran and run tran
   > cd TRANSIM

   (a) For QR method
   > make

   (b) For LU or GMRES method
   > make tran_LUGMRES

   > tran NETLIST

A.3 Memory Needed

A.3.1 QR Method

1. Double precision JAC(N,N): Store Jacobian Matrix
2. Double precision A(N,N): Store Scaled Jacobian Matrix
3. Double precision H(N,N): Store the finite difference Jacobian matrix in order to compare to the analytical Jacobian matrix if CHECKJ (CHECK Jacobian) is true
4. Double precision PLEE(N,N): Store P matrix used for Lee & Lee update method

For a matrix of size $4000 * 4000$ 1.1-1.4 will need $4 * 4000 * 4000 * 4$ bytes, about 244MB.

A.3.2 LU Method

1. Double precision JAC(N,N): Store Jacobian Matrix and Scaled Jacobian Matrix

2. Double precision H(N,N): Store the finite difference Jacobian matrix in order to compare to the analytical Jacobian matrix if CHECKJ (CHECK Jacobian) is true

For a matrix of size $4000 * 4000$ 2.1-2.2 will need $2 * 4000 * 4000 * 4$ bytes, about 122MB.

A.3.3 GMRES Method

1. Double precision JAC(N,N): Store Jacobian Matrix and Scaled Jacobian Matrix

2. Double precision H(N,N): Store the finite difference Jacobian matrix in order to compare to the analytical Jacobian matrix if CHECKJ (CHECK Jacobian) is true

3. Integer IA(NELT):

4. Integer JA(NELT):

5. Double precision A_JAC(NELT):
   Sparse preconditioner matrix is stored in the SLAP Triad format. In this format only the non-zeros are stored. They may appear in ANY order. The user supplies three arrays of length NELT, where NELT is the number of non-zeros in the matrix: (IA(NELT), JA(NELT), A_JAC(NELT)). For each non-zero the user puts the row and column index of that matrix element in the IA and JA arrays. The value of the non-zero matrix element is placed in the corresponding location of the A array.

6. Integer IWORK($40 + 5 * N$ + NELT): Temporary memory

7. Double Precision RWORK($1 + N * (50 + 7) + 50 * (50 + 3) + NELT$): Temporary memory

For a matrix of size $4000 * 4000$ 3.1-3.7 will need at most $5 * 4000 * 4000 * 4 + (40 + 5 * N + N * N) * 4 + (1 + N * (50 + 7) + 50 * (50 + 3) + N * N) * 4$ bytes, about 428MB, depending on the number of non-zeros in preconditioner matrix.

If NELT $= 0.75 * N * N$, it will need 352MB memory.

If NELT $= 0.25 * N * N$, it will need 199MB memory.

GMRES use AUTOMATIC ARRAY whose size is determined by dummy parameter NELT, and NELT varies from iteration to iteration when Jacobian matrix is evaluated explicitly. In G77, automatic array is stored in stack whose size can not exceed some limit, eg., 8MB, in one process. In order to use large automatic array in G77, we must set large stacksize when login to UNIX. If the system use CSH we may add "unlimit stacksize" to csh login file. Otherwise the "Segmentation fault" will occur. See description of sysdef, limit and unlimit command.

Appendix B

Programming Guide

B.1 TRANSIM

Figure B.1 shows the PAD [1] of TRANSIM.

B.2 NNES

Figure B.2 shows the PAD of original NNES.
Figure B.3 shows the PAD of modified NNES.

[1] Problem Analysis Diagram

Figure B.1: Dataflow of Transim(Related to NNES)

Figure B.2: Dataflow of Original NNES

Figure B.3: Dataflow of Modified NNES

Appendix C

Source Codes

C.1 Deleted files from original NNES

1. nstpun.f Find Newton step using unfactored Jacobian —>replaced
2. nstpfa.f Find Newton step using factored Jacobian —>replaced
3. broyun.f Update the unfactored Jacobian using Broyden's method —> replaced
4. broyfa.f Update the factored Jacobian using Broyden's method —> replaced

5. The above 4 subroutines are replaced by brokel.f brokli.f brogmr.f brogli.f

6. delcau.f Establishes an initial trust region
7. dogleg.f Find a trust region step using the (double) dogleg method.
8. trstup.f Check trust region
9. llun.f Update the unfactored Jacobian using Lee & Lee's method
10. llfa.f Update the factored Jacobian using Lee & Lee's method
11. qrdcom.f QR decomposition —> replaced by dgecom.f

12. qrsolv.f QR solver —> replaced by dgeslm.f
13. rsolv.f R solver —> replaced by dgeslm.f
14. deufls.f Conducts a line search —> replaced by bdeufl.f
15. onenrm.f Find 1-NORM
16. cholde.f Find the cholesky decomposition of a matrix
17. rtrmul.f Find $R\hat{R}$ for QR-Decomposed Jacobian
18. qrupda.f Update QR decomposition using a series of givens rotations
19. qform.f Form $Q\hat{}$from the householder matrices
20. jacrot.f Jacobi (or givens) rotation
21. condno.f Estimate the condition number of a QR-decomposed matrix
22. ataov.f Finds the product of the transpose of the matrix A and matrix A
23. chsolv.f Solve $(LL\hat{})S$=RHS
24. ltsolv.f Solve $L\hat{Y}$=B
25. lsolv.f Solve LB=RHS

C.2 Added source files to NNES

1. dgefam.f
2. dgecom.f
3. dgeslm.f
4. brokel.f
5. brokli.f
6. brogmr.f

7. brogli.f
8. bdeufl.f
9. gmres.f
10. dgmres.f
11. dhels.f
12. dheqr.f
13. dorth.f
14. dpigmr.f
15. drlcal.f
16. dxlcal.f
17. isdgmr.f
18. dscal.f
19. daxpy.f
20. ddot.f
21. dnrm2.f
22. dcopy.f
23. d1mach.f
24. xermsg.f
25. xersve.f
26. xerhlt.f
27. xercnt.f
28. xerprn.f
29. j4save.f

30. fdump.f
31. i1mach.f
32. xgetua.f
33. dslugm.f
34. ds2y.f
35. dchkw.f
36. dsilus.f
37. matvec.f
38. msolve.f
39. dslui2.f
40. qs2i1d.f
41. dsds.f
42. svd.f
43. pythag.f
44. rg.f
45. balanc.f
46. elmhes.f
47. hqr.f
48. hqr2.f
49. balbak.f
50. eltran.f
51. cdiv.f
52. dsort.f

53. dgemv.f
54. lsame.f
55. xerbla.f

C.3 Modified source files

1. svHB_interface_local.c (instead of svHB_interface.c)
2. setup.f
3. nnes.f
4. nnes_svHB.f
5. initch.f
6. nestop.f
7. avmul.f
8. bdeufl.f
9. brogli.f
10. brogmr.f
11. brokel.f
12. brokli.f
13. dgecom.f
14. dgmres.f
15. dpigmr.f
16. dslugm.f
17. dsmv.f
18. gmres.f

19. isdgmr.f
20. matvec.f
21. msolve.f

Appendix D

Netlist Files

The netlist files we listed here are for GMRES method only. The differences among QR, LU and GMRES methods are the values of "method" and "jupdm" parameters. See "Userguide".
Soliton04.net denotes soliton lines circuit with 4 diodes, and soliton10.net denotes soliton lines circuit with 10 diodes,... . For sparse cases, we may change "mv" parameters: soliton04.net , mv =<-6.,4., ... ; soliton10.net, mv=<-6.,4., ... ; soliton20.net, mv=<-6.,3.2., ... ; soliton30.net, mv=<-6.,1., ...

1. fullgrid.net

```
fullgrid_masummers.net

.options iterationdump = "off" dumptables = "on" nonlinear = "on"
+ dump = "off" verbose ="on" writenet = "off" dumpnet = "off"
+ tree = "off"  tr_debug = "on"
+ type = "svhb" freq=6e9 web ="set"

.options x = \ 5.4e9 5.4e9 .1e9 \

.hb spts = 16 fundamental =x deriv=0
+max_iter = 500 level =2 oversample= 2 tolerance = 1e-8
+method=3  jupdm = 4 verb_level=2 step = 0

*Excitation Sources

iacfile:iin 1 0  101 0  5 0  105 0    201  0 301 0
+  205 0  305 0  401 0  501 0   405 0  505 0
+  601 0    701 0
```

```
+  605 0 705 0
+ 0 0 0 0 0 0 0 0
+ 0 0 0 0 0 0 0 0
+ 0 0 0 0 0 0 0 0
+ ifilename = "fullgrid_cmp.i" efilename = "horngrid.e"

* Input bias source.

vdc:vindc 3000 0 v=4.5

* Output bias Source
vdc:vcc 21 0 v=4.25

* Net element Interface to Emag model

net:ngrid 1 101 5 105 201 301 205 305
+ 401 501 405 505 601 701 605 705
+"ind11" "ind12" "ind13" "ind14" "ind15" "ind16"
+"ind17" "ind18"
+"ind19" "ind110"  "ind111" "ind112"
+"LR1" "LR2" "LR3" "LR4"
+"ind21" "ind22" "ind23" "ind24" "ind25" "ind26"
+"ind27" "ind28"
+"ind29" "ind210"  "ind211" "ind212"
+  filename ="fullgrid_cmp.yp" matrixtype = "nodal"

************************************************************
*  Bias inductors
************************************************************

ind:ind1 "ind11" "ind21"  l=11e-9
ind:ind2 "ind12" "ind22"  l=11e-9
ind:ind3 "ind13" "ind23"  l=11e-9
ind:ind4 "ind14" "ind24"  l=11e-9
ind:ind5 "ind15" "ind25"  l=11e-9
ind:ind6 "ind16" "ind26"  l=11e-9
ind:ind7 "ind17" "ind27"  l=11e-9
ind:ind8 "ind18" "ind28"  l=11e-9
ind:ind9 "ind19" "ind29"  l=11e-9
ind:ind10 "ind110" "ind210"  l=11e-9
ind:ind11 "ind111" "ind211"  l=11e-9
ind:ind12 "ind112" "ind212"  l=11e-9
res:rind1 "ind21" 0 r=1e7
res:rind2 "ind22" 0 r=1e7
res:rind3 "ind23" 0 r=1e7
res:rind4 "ind24" 0 r=1e7
res:rind5 "ind25" 0 r=1e7
res:rind6 "ind26" 0 r=1e7
res:rind7 "ind27" 0 r=1e7
res:rind8 "ind28" 0 r=1e7
res:rind9 "ind29" 0 r=1e7
res:rind10 "ind210" 0 r=1e7
res:rind11 "ind211" 0 r=1e7
res:rind12 "ind212" 0 r=1e7
```

```
* Input Bias Network

* Input bias decoupling for device 1 Unit Cell 1
res:rindc 3000 16 r=50
ind:lindc 16 1 l=1
* Input bias decoupling for device 2 Unit Cell 1
res:rindc2 3000 1016 r=50
ind:lindc2 1016 101 l=1
* Input bias decoupling for device 1 Unit Cell 2
res:rindc3 3000 2016 r=50
ind:lindc3 2016 201 l=1
* Input bias decoupling for device 2 Unit Cell 2
res:rindc4 3000 3016 r=50
ind:lindc4 3016 301 l=1
* Input bias decoupling for device 1 Unit Cell 3
res:rindc5 3000 4016 r=50
ind:lindc5 4016 401 l=1
* Input bias decoupling for device 2 Unit Cell 3
res:rindc6 3000 5016 r=50
ind:lindc6 5016 501 l=1
* Input bias decoupling for device 1 Unit Cell 4
res:rindc7 3000 6016 r=50
ind:lindc7 6016 601 l=1
* Input bias decoupling for device 2 Unit Cell 4
res:rindc8 3000 7016 r=50
ind:lindc8 7016 701 l=1

* Output Bias Network

*Thermal resistor  Equivalent resistor for 8 50
* ohm resistors in parallel

res:rth 21 20 r=6.25

*Decoupling network for device 1 Unit Cell 1
ind:lcc 20 4 l=1e-3
*Decoupling network for device 2 Unit Cell 1
ind:lcc2 20 104 l=1e-3
*Decoupling network for device 1 Unit Cell 2
ind:lcc3 20 204 l=1e-3
*Decoupling network for device 2 Unit Cell 2
ind:lcc4 20 304 l=1e-3
*Decoupling network for device 1 Unit Cell 3
ind:lcc5 20 404 l=1e-3
*Decoupling network for device 2 Unit Cell 3
ind:lcc6 20 504 l=1e-3
*Decoupling network for device 1 Unit Cell 4
ind:lcc7 20 604 l=1e-3
*Decoupling network for device 2 Unit Cell 4
ind:lcc8 20 704 l=1e-3

* Connection to DC Ground
```

```
*Unit Cell 1
ind:lgnd1 "LR1" 0 l=1e-9
*Unit Cell 2
ind:lgnd2 "LR2" 0 l=1e-9
*Unit Cell 3
ind:lgnd3 "LR3" 0 l=1e-9
*Unit Cell 4
ind:lgnd4 "LR4" 0 l=1e-9

**************************************************
* Device 1 for unit cell 1
* Input node = 1
* Output node = 5
* Common node = LR1
**************************************************
vct_ssig:a11 2 54 3  gamma=-2.3 beta=8 kf=0.0025
vct:b11 2 4 15  gamma=1.1 beta=1 kf=1
+ poly 1 2 15 5.979287283058695e-4 -2.334033466242205e-02
+ 6.282215276145409e-02
+ -4.319540162756818e-02 9.169340904889009e-03
ind:ssig11 54 0 l=1e-3
cap:ssig11 54 4 c=1e-3
ind:lin11 1 2 l=1.0238e-9
cap:cin11 2 15 c= 0.32713e-12
net:nin11 2 3 filename = "netin.yp" matrixtype = "nodal"
ind:le11 15 55 l=0.1e-9
res:re11 55 "LR1" r=7
cap:ce11 55 "LR1" c=1e-3
net:ne11 3 15 filename = "freqcomp.yp" matrixtype = "nodal"
res:rout11 24 15 r=80
cap:cdum11 24 4 c=1
cap:cout11 4 15 c=0.64225e-12
ind:lout11 4 5 l=1.2361e-9
cap:coutsub11 5 "LR1" c=0.12864e-12

*End of device 1 in unit cell 1

**************************************************
* Device 2 for Unit Cell 1
* Input node  101
* Output node 105
* Common node LR1
**************************************************
vct_ssig:a21 102 1054 103  gamma=-2.3 beta=8 kf=0.0025
vct:b21 102 104 1015  gamma=1.1 beta=1 kf=1
+ poly 1 2 15 5.979287283058695e-4 -2.334033466242205e-02
+6.282215276145409e-02
+ -4.319540162756818e-02 9.169340904889009e-03
ind:ssig21 1054 0 l=1e-3
cap:ssig21 1054 104 c=1e-3
ind:lin21 101 102 l=1.0238e-9
cap:cin21 102 1015 c= 0.32713e-12
net:nin21 102 103 filename = "netin.yp" matrixtype = "nodal"
ind:le21 1015 1055 l=0.1e-9
```

```
res:re21 1055 "LR1" r=7
cap:ce21 1055 "LR1" c=1e-3
net:ne21 103 1015 filename = "freqcomp.yp" matrixtype = "nodal"
res:rout21 1024 1015 r=80
cap:cdum21 1024 104 c=1
cap:cout21 104 1015 c=0.64225e-12
ind:lout21 104 105 l=1.2361e-9
cap:coutsub21 105 "LR1"  c=0.12864e-12

************************************************************
* End of Device 2 Unit Cell 1
************************************************************

************************************************************
* Device 1 for unit cell 2
* Input node = 201
* Output node = 205
* Common node = LR2
************************************************************
vct_ssig:a12 202 2054 203  gamma=-2.3 beta=8 kf=0.0025
vct:b12 202 204 2015  gamma=1.1 beta=1 kf=1
+ poly 1 2 15 5.979287283058695e-4 -2.334033466242205e-02
+6.282215276145409e-02
+ -4.319540162756818e-02 9.169340904889009e-03
ind:ssig12 2054 0 l=1e-3
cap:ssig12 2054 204 c=1e-3
ind:lin12 201 202 l=1.0238e-9
cap:cin12 202 2015 c= 0.32713e-12
net:nin12 202 203 filename = "netin.yp" matrixtype = "nodal"
ind:le12 2015 2055 l=0.1e-9
res:re12 2055 "LR2" r=7
cap:ce12 2055 "LR2" c=1e-3
net:ne12 203 2015 filename = "freqcomp.yp" matrixtype = "nodal"
res:rout12 2024 2015 r=80
cap:cdum12 2024 204 c=1
cap:cout12 204 2015 c=0.64225e-12
ind:lout12 204 205 l=1.2361e-9
cap:coutsub12 205 "LR2" c=0.12864e-12

************************************************************
*End of device 1 in unit cell 2
************************************************************

************************************************************
* Device 2 for unit cell 2
* Input node = 301
* Output node = 305
* Common node = LR2
************************************************************
vct_ssig:a22 302 304 303  gamma=-2.3 beta=8 kf=0.0025
vct:b22 302 304 3015  gamma=1.1 beta=1 kf=1
+ poly 1 2 15 5.979287283058695e-4 -2.334033466242205e-02
+6.282215276145409e-02
+ -4.319540162756818e-02 9.169340904889009e-03
```

```
ind:ssig22 3054 0 l=1e-3
cap:ssig22 3054 304 c=1e-3
ind:lin22 301 302 l=1.0238e-9
cap:cin22 302 3015 c= 0.32713e-12
net:nin22 302 303 filename = "netin.yp" matrixtype = "nodal"
ind:le22 3015 3055 l=0.1e-9
res:re22 3055 "LR2" r=7
cap:ce22 3055 "LR2" c=1e-3
net:ne22 303 3015 filename = "freqcomp.yp" matrixtype = "nodal"
res:rout22 3024 3015 r=80
cap:cdum22 3024 304 c=1
cap:cout22 304 3015 c=0.64225e-12
ind:lout22 304 305 l=1.2361e-9
cap:coutsub22 305 "LR2" c=0.12864e-12

**************************************************************
*End of device 2 in unit cell 2
**************************************************************

**************************************************************
* Device 1 for unit cell 3
* Input node = 401
* Output node = 405
* Common node = LR3
**************************************************************
vct_ssig:a13 402 4054 403  gamma=-2.3 beta=8 kf=0.0025
vct:b13 402 404 4015  gamma=1.1 beta=1 kf=1
+ poly 1 2 15 5.979287283058695e-4 -2.334033466242205e-02
+6.282215276145409e-02
+ -4.319540162756818e-02 9.169340904889009e-03
ind:ssig13 4054 0 l=1e-3
cap:ssig13 4054 404 c=1e-3
ind:lin13 401 402 l=1.0238e-9
cap:cin13 402 4015 c= 0.32713e-12
net:nin13 402 403 filename = "netin.yp" matrixtype = "nodal"
ind:le13 4015 4055 l=0.1e-9
res:re13 4055 "LR3" r=7
cap:ce13 4055 "LR3" c=1e-3
net:ne13 403 4015 filename = "freqcomp.yp" matrixtype = "nodal"
res:rout13 4024 4015 r=80
cap:cdum13 4024 404 c=1
cap:cout13 404 4015 c=0.64225e-12
ind:lout13 404 405 l=1.2361e-9
cap:coutsub13 405 "LR3" c=0.12864e-12

**************************************************************
*End of device 1 in unit cell 3
**************************************************************

**************************************************************
* Device 2 for unit cell 3
* Input node = 501
* Output node = 505
* Common node = LR3
```

```
*********************************************************
vct_ssig:a23 502 5054 503  gamma=-2.3 beta=8 kf=0.0025
vct:b23 502 504 5015  gamma=1.1 beta=1 kf=1
+ poly 1 2 15 5.979287283058695e-4 -2.334033466242205e-02
+6.282215276145409e-02
+ -4.319540162756818e-02 9.169340904889009e-03
ind:ssig23 5054 0 l=1e-3
cap:ssig23 5054 504 c=1e-3
ind:lin23 501 502 l=1.0238e-9
cap:cin23 502 5015 c= 0.32713e-12
net:nin23 502 503 filename = "netin.yp" matrixtype = "nodal"
ind:le23 5015 5055 l=0.1e-9
res:re23 5055 "LR3" r=7
cap:ce23 5055 "LR3" c=1e-3
net:ne23 503 5015 filename = "freqcomp.yp" matrixtype = "nodal"
res:rout23 5024 5015 r=80
cap:cdum23 5024 504 c=1
cap:cout23 504 5015 c=0.64225e-12
ind:lout23 504 505 l=1.2361e-9
cap:coutsub23 505 "LR3" c=0.12864e-12

*********************************************************
*End of device 2 in unit cell 3
*********************************************************

*********************************************************
* Device 1 for unit cell 4
* Input node = 601
* Output node = 605
* Common node = LR4
*********************************************************
vct_ssig:a14 602 6054 603  gamma=-2.3 beta=8 kf=0.0025
vct:b14 602 604 6015  gamma=1.1 beta=1 kf=1
+ poly 1 2 15 5.979287283058695e-4 -2.334033466242205e-02
+6.282215276145409e-02
+ -4.319540162756818e-02 9.169340904889009e-03
ind:ssig14 6054 0 l=1e-3
cap:ssig14 6054 604 c=1e-3
ind:lin14 601 602 l=1.0238e-9
cap:cin14 602 6015 c= 0.32713e-12
net:nin14 602 603 filename = "netin.yp" matrixtype = "nodal"
ind:le14 6015 6055 l=0.1e-9
res:re14 6055 "LR4" r=7
cap:ce14 6055 "LR4" c=1e-3
net:ne14 603 6015 filename = "freqcomp.yp" matrixtype = "nodal"
res:rout14 6024 6015 r=80
cap:cdum14 6024 604 c=1
cap:cout14 604 6015 c=0.64225e-12
ind:lout14 604 605 l=1.2361e-9
cap:coutsub14 605 "LR4" c=0.12864e-12

*********************************************************
*End of device 1 in unit cell 4
*********************************************************
```

```
**********************************************************
* Device 2 for unit cell 4
* Input node = 701
* Output node = 705
* Common node = LR4
**********************************************************
vct_ssig:a24 702 7054 703  gamma=-2.3 beta=8 kf=0.0025
vct:b24 702 704 7015  gamma=1.1 beta=1 kf=1
+ poly 1 2 15 5.979287283058695e-4 -2.334033466242205e-02
+ 6.282215276145409e-02
+ -4.319540162756818e-02 9.169340904889009e-03
ind:ssig24 7054 0 l=1e-3
cap:ssig24 7054 704 c=1e-3
ind:lin24 701 702 l=1.0238e-9
cap:cin24 702 7015 c= 0.32713e-12
net:nin24 702 703 filename = "netin.yp" matrixtype = "nodal"
ind:le24 7015 7055 l=0.1e-9
res:re24 7055 "LR4" r=7
cap:ce24 7055 "LR4" c=1e-3
net:ne24 703 7015 filename = "freqcomp.yp" matrixtype = "nodal"
res:rout24 7024 7015 r=80
cap:cdum24 7024 704 c=1
cap:cout24 704 7015 c=0.64225e-12
ind:lout24 704 705 l=1.2361e-9
cap:coutsub24 705 "LR4" c=0.12864e-12

**********************************************************
*End of device 2 in unit cell 4
**********************************************************

* * Write out port voltages
* .out variable "xxx"
* + term 1 v term "LR1" v sub term 101 v
* +term "LR1" v sub -1 mult
* + term 5 v  term "LR1" v sub -1 mult
* +term 105 v term "LR1" v sub
* + term 201 v term "LR2" v sub term 301 v
* +term "LR2" v sub -1 mult
* + term  205 v term "LR2" v sub -1 mult
* +term 305 v term "LR2" v sub
* + term 401 v term "LR3" v sub term 501 v
* +term "LR3" v sub -1 mult
* + term 405 v term "LR3" v sub -1 mult
* + term  505 v term "LR3" v sub
* + term 601 v term "LR4" v sub  term 701 v
* +term "LR4" v sub -1 mult
* + term 605 v term "LR4" v sub -1 mult
* +term 705 v term "LR4" v sub
* + term "ind11" v term "ind21" v sub term "ind12" v
* + term "ind22" v sub
* + term "ind13" v term "ind23" v sub term "ind14" v
* +term "ind24" v sub
* + term "ind15" v term "ind25" v sub  term "ind16" v
```

```
* +term "ind26" v sub
* + term "ind17" v term "ind27" v sub  term "ind18" v
* +term "ind28" v sub
* + term "ind19" v term "ind29" v sub  term "ind110" v
* + term "ind210" v sub
* + term "ind111" v term "ind211" v sub  term "ind112" v
* +term "ind212" v sub
* + 1 cat  "xxx" append
* .out end write "xxx" push in "fullgridcmp.v"
.end
```

2. soliton04.net : dense case

```
spice.net
.options type = "svhb"
.hb iterationdump = "off" dumptables = "off" nonlinear = "on"
+ dump = "on" writenet = "off" dumpnet = "off" tree = "off"
+ mattype = "MNA" spts = 32 fundamental = 9.e9
+ max_iter = 5000 chord=1
+ oversample=4
+ jupdm= 4 verb_level=2 steps = 0 oversample=2 deriv = 0
+ method=3
*+ gminstep=0
*+ block=4
*
mvac:ain   202 0 mf=<0.0,9.e9,18.e9,27.e9,36.e9,45.e9,54.e9,63.e9,72.e9,
+ 81.e9,90.e9,99.e9,108.e9,117.e9,126.e9,135.e9,144.e9,153.e9,162.e9,
+ 171.e9,180.e9,189.e9,198.e9,207.e9,216.e9,225.e9,234.e9,243.e9,252.e9,
+ 261.e9,270.e9,279.e9,288.e9,297.e9,306.e9,315.e9,324.e9,333.e9,342.e9,
+ 351.e9,360.e9,369.e9,378.e9,387.e9,396.e9,405.e9,414.e9,423.e9,432.e9,
+ 441.e9,450.e9,459.e9,468.e9,477.e9,486.e9,495.e9,504.e9,513.e9,522.e9,
+ 531.e9,540.e9,549.e9,558.e9,567.e9>
+ mv=<-6.,12.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.>
+ mphase=<0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.>
*+ mv=<-6.,4.94975,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
*
diode:D1 1 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=0. vb= -16. area=271.64
diode:D2 2 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=0. vb=-16. area=258.63
diode:D3 3 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=246.24
diode:D4 4 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=234.45
*
*res:r1 1 101 r=.6328153
*res:r2 2 102 r=.6646545
*res:r3 3 103 r=.6980956
*res:r4 4 104 r=.7332192
```

```
*
tlinpy:T0 201 1 z0mag=75.00 length=1027.5 k=7 width=30. f=1.e10
tlinpy:T1  1  2  z0mag=75.00 length=978.57 k=7 width=30. f=1.e10
tlinpy:T2  2  3 z0mag=75.00 length=931.69 k=7 width=30. f=1.e10
tlinpy:T3  3  4 z0mag=75.00 length=887.06 k=7 width=30. f=1.e10
tlinpy:T4  4  5  z0mag=75.00 length=844.57 k=7 width=30. f=1.e10
*res:rl   5  0  R=75. nonlinear=1 poly 1 2 0 0. 1.
res:rl   5  0  R=75.
res:rs  202 201 R=75.
.end
```

3. soliton10.net : dense case

```
spice.net
.options type = "svhb"
.hb iterationdump = "off" dumptables = "off" nonlinear = "on"
+ dump = "on" writenet = "off" dumpnet = "off" tree = "off" mattype = "MNA"
+ spts = 32 fundamental = 9.e9
+ max_iter = 5000 chord=1
+ oversample=4
+jupdm= 4 verb_level=2 steps = 0 oversample=2
+method=3
*+ gminStepFlag=1
*+ block=2
*
mvac:ain   202 0 mf=<0.0,9.e9,18.e9,27.e9,36.e9,45.e9,54.e9,63.e9,72.e9,
+ 81.e9,90.e9,99.e9,108.e9,117.e9,126.e9,135.e9,144.e9,153.e9,162.e9,
+ 171.e9,180.e9,189.e9,198.e9,207.e9,216.e9,225.e9,234.e9,243.e9,252.e9,
+ 261.e9,270.e9,279.e9,288.e9,297.e9,306.e9,315.e9,324.e9,333.e9,342.e9,
+ 351.e9,360.e9,369.e9,378.e9,387.e9,396.e9,405.e9,414.e9,423.e9,432.e9,
+ 441.e9,450.e9,459.e9,468.e9,477.e9,486.e9,495.e9,504.e9,513.e9,522.e9,
+ 531.e9,540.e9,549.e9,558.e9,567.e9>
+ mv=<-6.,9.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.>
+ mphase=<0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.>
*+ mv=<-6.,6.131,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
*+ mv=<-6.,4.94975,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
*
diode:D1 1 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=0. vb= -16. area=271.64
diode:D2 2 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=0. vb=-16. area=258.63
diode:D3 3 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=246.24
diode:D4 4 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=234.45
diode:D5 5 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=223.21
diode:D6 6 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=212.52
diode:D7 7 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=202.34
```

```
diode:D8 8 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=192.65
diode:D9 9 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=183.42
diode:D10 10 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=174.63
*
tlinpy:T0 201 1 z0mag=75.00 length=501.29 k=7 a=5.990686e-5 f=1.e10
tlinpy:T1  1  2  z0mag=75.00 length=978.57 k=7 a=5.990686e-05 f=1.e10
tlinpy:T2  2  3 z0mag=75.00 length=931.69 k=7 a=5.990686e-05 f=1.e10
tlinpy:T3  3  4 z0mag=75.00 length=887.06 k=7 a=5.990686e-05 f=1.e10
tlinpy:T4  4  5  z0mag=75.00 length=844.57 k=7 a=5.990686e-05 f=1.e10
tlinpy:T5  5  6  z0mag=75.00 length=804.11 k=7 a=5.990686e-05 f=1.e10
tlinpy:T6  6  7  z0mag=75.00 length=765.59 k=7 a=5.990686e-05 f=1.e10
tlinpy:T7  7  8  z0mag=75.00 length=728.92 k=7 a=5.990686e-05 f=1.e10
tlinpy:T8  8  9  z0mag=75.00 length=694.00 k=7 a=5.990686e-05 f=1.e10
tlinpy:T9  9  10  z0mag=75.00 length=660.75 k=7 a=5.990686e-05 f=1.e10
tlinpy:T10  10  11  z0mag=75.00 length=629.10 k=7 a=5.990686e-05 f=1.e10
*
*res:rl   11  0  R=75. nonlinear=1 poly 1 2 0 0. 1.
res:rl   11  0  R=75.
res:rs  202 201 R=75.
.end
```

4. soliton20.net : dense case

```
spice.net
.options type = "svhb"
.hb iterationdump = "off" dumptables = "off" nonlinear = "on"
+ dump = "on" writenet = "off" dumpnet = "off" tree = "off"
+ mattype = "MNA"  spts = 32 fundamental = 9.e9
+ max_iter = 5000 chord=1
+ oversample=2
+ jupdm= 4 verb_level=2 steps = 0 oversample=2
+method=3
*+ block=4
*
mvac:ain   202 0 mf=<0.0,9.e9,18.e9,27.e9,36.e9,45.e9,54.e9,63.e9,72.e9,
+ 81.e9,90.e9,99.e9,108.e9,117.e9,126.e9,135.e9,144.e9,153.e9,162.e9,
+ 171.e9,180.e9,189.e9,198.e9,207.e9,216.e9,225.e9,234.e9,243.e9,252.e9,
+ 261.e9,270.e9,279.e9,288.e9,297.e9,306.e9,315.e9,324.e9,333.e9,342.e9,351.e9,
+ 360.e9,369.e9,378.e9,387.e9,396.e9,405.e9,414.e9,423.e9,432.e9,441.e9,450.e9,
+ 459.e9,468.e9,477.e9,486.e9,495.e9,504.e9,513.e9,522.e9,531.e9,540.e9,549.e9,
+ 558.e9,567.e9>
+ mv=<-6.,9.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.>
+ mphase=<0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.>
*+ mv=<-6.,4.94975,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
diode:D1 1 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=0. vb= -16. area=271.64
diode:D2 2 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=0. vb=-16. area=258.63
diode:D3 3 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=246.24
diode:D4 4 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=234.45
diode:D5 5 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=223.21
diode:D6 6 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=212.52
diode:D7 7 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=202.34
diode:D8 8 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=192.65
```

```
diode:D9 9 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=183.42
diode:D10 10 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=174.63
diode:D11 11 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=166.27
diode:D12 12 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=158.3
diode:D13 13 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=150.72
diode:D14 14 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=143.5
diode:D15 15 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=136.63
diode:D16 16 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=130.08
diode:D17 17 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=123.85
diode:D18 18 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=117.92
diode:D19 19 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=112.27
diode:D20 20 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=106.89
*
tlinpy:T0 201 1 z0mag=75.00 length=501.29 k=7 a=5.990686e-5 f=1.e10
tlinpy:T1  1  2  z0mag=75.00 length=978.57 k=7 a=5.990686e-05 f=1.e10
tlinpy:T2  2  3 z0mag=75.00 length=931.69 k=7 a=5.990686e-05 f=1.e10
tlinpy:T3  3  4 z0mag=75.00 length=887.06 k=7 a=5.990686e-05 f=1.e10
tlinpy:T4  4  5  z0mag=75.00 length=844.57 k=7 a=5.990686e-05 f=1.e10
tlinpy:T5  5  6  z0mag=75.00 length=804.11 k=7 a=5.990686e-05 f=1.e10
tlinpy:T6  6  7  z0mag=75.00 length=765.59 k=7 a=5.990686e-05 f=1.e10
tlinpy:T7  7  8  z0mag=75.00 length=728.92 k=7 a=5.990686e-05 f=1.e10
tlinpy:T8  8  9  z0mag=75.00 length=694.00 k=7 a=5.990686e-05 f=1.e10
tlinpy:T9  9  10  z0mag=75.00 length=660.75 k=7 a=5.990686e-05 f=1.e10
tlinpy:T10  10  11  z0mag=75.00 length=629.10 k=7 a=5.990686e-05 f=1.e10
tlinpy:T11  11  12  z0mag=75.00 length=598.97 k=7 a=5.990686e-05 f=1.e10
tlinpy:T12  12  13  z0mag=75.00 length=570.27 k=7 a=5.990686e-05 f=1.e10
tlinpy:T13  13  14  z0mag=75.00 length=542.96 k=7 a=5.990686e-05 f=1.e10
tlinpy:T14  14  15  z0mag=75.00 length=516.95 k=7 a=5.990686e-05 f=1.e10
tlinpy:T15  15  16  z0mag=75.00 length=492.18 k=7 a=5.990686e-05 f=1.e10
tlinpy:T16  16  17  z0mag=75.00 length=468.61 k=7 a=5.990686e-05 f=1.e10
tlinpy:T17  17  18  z0mag=75.00 length=446.16 k=7 a=5.990686e-05 f=1.e10
tlinpy:T18  18  19  z0mag=75.00 length=424.79 k=7 a=5.990686e-05 f=1.e10
tlinpy:T19  19  20  z0mag=75.00 length=404.44 k=7 a=5.990686e-05 f=1.e10
```

```
tlinpy:T20  20  21  z0mag=75.00 length=385.06 k=7 a=5.990686e-05 f=1.e10
*
res:rl   21  0  R=75. nonlinear=1 poly 1 2 0 0. 1.
res:rs  202 201 R=75.
.end
```

5. soliton30.net : dense case

```
spice.net
.options  type = "svhb"
.hb iterationdump = "off" dumptables = "off" nonlinear = "on"
+ dump = "on" writenet = "off" dumpnet = "off" tree = "off" mattype = "MNA"
+ type = "hb" spts = 32 fundamental=9.e9
+ max_iter = 500 chord=1
+ jupdm= 4 verb_level=2 steps = 0 oversample=2
+method=3
*+ oversample=2
*+ block=2

mvac:ain   202 0 mf=<0.0,9.e9,18.e9,27.e9,36.e9,45.e9,54.e9,63.e9,72.e9,
+ 81.e9,90.e9,99.e9,108.e9,117.e9,126.e9,135.e9,144.e9,153.e9,162.e9,
+ 171.e9,180.e9,189.e9,198.e9,207.e9,216.e9,225.e9,234.e9,243.e9,252.e9,
+ 261.e9,270.e9,279.e9,288.e9,297.e9,306.e9,315.e9,324.e9,333.e9,342.e9,
+ 351.e9,360.e9,369.e9,378.e9,387.e9,396.e9,405.e9,414.e9,423.e9,432.e9,
+ 441.e9,450.e9,459.e9,468.e9,477.e9,486.e9,495.e9,504.e9,513.e9,522.e9,
+ 531.e9,540.e9,549.e9,558.e9,567.e9>
+ mv=<-6.,9.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.>
+ mphase=<0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
+ 0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.>
*
diode:D1 1 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=0. vb= -16. area=271.64
diode:D2 2 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=0. vb=-16. area=258.63
diode:D3 3 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=246.24
diode:D4 4 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=234.45
diode:D5 5 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=223.21
diode:D6 6 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=212.52
diode:D7 7 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
```

```
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=202.34
diode:D8 8 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=192.65
diode:D9 9 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=183.42
diode:D10 10 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=174.63
diode:D11 11 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=166.27
diode:D12 12 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=158.3
diode:D13 13 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=150.72
diode:D14 14 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=143.5
diode:D15 15 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=136.63
diode:D16 16 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=130.08
diode:D17 17 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=123.85
diode:D18 18 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=117.92
diode:D19 19 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=112.27
diode:D20 20 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=106.89
diode:D21 21 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=101.77
diode:D22 22 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=96.89
diode:D23 23 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=92.25
diode:D24 24 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=87.83
diode:D25 25 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=83.63
diode:D26 26 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=79.62
diode:D27 27 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=75.81
diode:D28 28 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=72.18
diode:D29 29 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=68.72
```

```
diode:D30 30 0 js=1.e-12 alfa=38.696 e=10 ct0=1.08e-16
+ fi=0.643 gama=0.451 jb=1.e-5 vb=-16. area=65.43
*
tlinpy:T0 201 1 z0mag=75.00 length=501.29 k=7 a=5.990686e-5 f=1.e10
tlinpy:T1  1  2  z0mag=75.00 length=978.57 k=7 a=5.990686e-05 f=1.e10
tlinpy:T2  2  3 z0mag=75.00 length=931.69 k=7 a=5.990686e-05 f=1.e10
tlinpy:T3  3  4 z0mag=75.00 length=887.06 k=7 a=5.990686e-05 f=1.e10
tlinpy:T4  4  5  z0mag=75.00 length=844.57 k=7 a=5.990686e-05 f=1.e10
tlinpy:T5  5  6  z0mag=75.00 length=804.11 k=7 a=5.990686e-05 f=1.e10
tlinpy:T6  6  7  z0mag=75.00 length=765.59 k=7 a=5.990686e-05 f=1.e10
tlinpy:T7  7  8  z0mag=75.00 length=728.92 k=7 a=5.990686e-05 f=1.e10
tlinpy:T8  8  9  z0mag=75.00 length=694.00 k=7 a=5.990686e-05 f=1.e10
tlinpy:T9  9  10  z0mag=75.00 length=660.75 k=7 a=5.990686e-05 f=1.e10
tlinpy:T10  10  11  z0mag=75.00 length=629.10 k=7 a=5.990686e-05 f=1.e10
tlinpy:T11  11  12  z0mag=75.00 length=598.97 k=7 a=5.990686e-05 f=1.e10
tlinpy:T12  12  13  z0mag=75.00 length=570.27 k=7 a=5.990686e-05 f=1.e10
tlinpy:T13  13  14  z0mag=75.00 length=542.96 k=7 a=5.990686e-05 f=1.e10
tlinpy:T14  14  15  z0mag=75.00 length=516.95 k=7 a=5.990686e-05 f=1.e10
tlinpy:T15  15  16  z0mag=75.00 length=492.18 k=7 a=5.990686e-05 f=1.e10
tlinpy:T16  16  17  z0mag=75.00 length=468.61 k=7 a=5.990686e-05 f=1.e10
tlinpy:T17  17  18  z0mag=75.00 length=446.16 k=7 a=5.990686e-05 f=1.e10
tlinpy:T18  18  19  z0mag=75.00 length=424.79 k=7 a=5.990686e-05 f=1.e10
tlinpy:T19  19  20  z0mag=75.00 length=404.44 k=7 a=5.990686e-05 f=1.e10
tlinpy:T20  20  21  z0mag=75.00 length=385.06 k=7 a=5.990686e-05 f=1.e10
tlinpy:T21  21  22  z0mag=75.00 length=366.62 k=7 a=5.990686e-05 f=1.e10
tlinpy:T22  22  23  z0mag=75.00 length=349.05 k=7 a=5.990686e-05 f=1.e10
tlinpy:T23  23  24  z0mag=75.00 length=332.33 k=7 a=5.990686e-05 f=1.e10
tlinpy:T24  24  25  z0mag=75.00 length=316.41 k=7 a=5.990686e-05 f=1.e10
tlinpy:T25  25  26  z0mag=75.00 length=301.26 k=7 a=6.197262e-05 f=1.e10
tlinpy:T26  26  27  z0mag=75.00 length=286.83 k=7 a=6.418592e-05 f=1.e10
tlinpy:T27  27  28  z0mag=75.00 length=273.09 k=7 a=6.656318e-05 f=1.e10
tlinpy:T28  28  29  z0mag=75.00 length=260.00 k=7 a=7.188824e-05 f=1.e10
tlinpy:T29  29  30  z0mag=75.00 length=247.55 k=7 a=7.488358e-05 f=1.e10
tlinpy:T30  30  31  z0mag=75.00 length=235.69 k=7 a=7.813938e-05 f=1.e10
*
res:rl   31  0  R=75.
res:rs  202 201 R=75.
.end
```

射 频 和 天 线 设 计 培 训 课 程 推 荐

易迪拓培训(www.edatop.com)由数名来自于研发第一线的资深工程师发起成立，致力并专注于微波、射频、天线设计研发人才的培养；我们于 2006 年整合合并微波 EDA 网(www.mweda.com)，现已发展成为国内最大的微波射频和天线设计人才培养基地，成功推出多套微波射频以及天线设计经典培训课程和 ADS、HFSS 等专业软件使用培训课程，广受客户好评；并先后与人民邮电出版社、电子工业出版社合作出版了多本专业图书，帮助数万名工程师提升了专业技术能力。客户遍布中兴通讯、研通高频、埃威航电、国人通信等多家国内知名公司，以及台湾工业技术研究院、永业科技、全一电子等多家台湾地区企业。

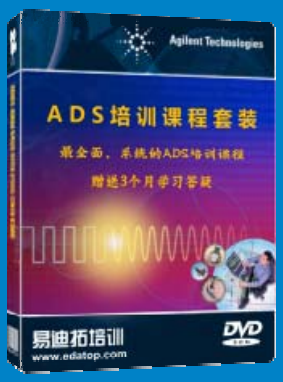
易迪拓培训课程列表：http://www.edatop.com/peixun/rfe/129.html

射频工程师养成培训课程套装

该套装精选了射频专业基础培训课程、射频仿真设计培训课程和射频电路测量培训课程三个类别共 30 门视频培训课程和 3 本图书教材；旨在引领学员全面学习一个射频工程师需要熟悉、理解和掌握的专业知识和研发设计能力。通过套装的学习，能够让学员完全达到和胜任一个合格的射频工程师的要求…

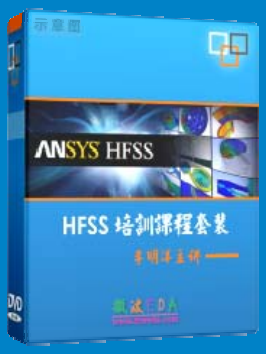
课程网址：http://www.edatop.com/peixun/rfe/110.html

ADS 学习培训课程套装



该套装是迄今国内最全面、最权威的 ADS 培训教程，共包含 10 门 ADS 学习培训课程。课程是由具有多年 ADS 使用经验的微波射频与通信系统设计领域资深专家讲解，并多结合设计实例，由浅入深、详细而又全面地讲解了 ADS 在微波射频电路设计、通信系统设计和电磁仿真设计方面的内容。能让您在最短的时间内学会使用 ADS，迅速提升个人技术能力，把 ADS 真正应用到实际研发工作中去，成为 ADS 设计专家...

课程网址： http://www.edatop.com/peixun/ads/13.html



HFSS 学习培训课程套装

该套课程套装包含了本站全部 HFSS 培训课程，是迄今国内最全面、最专业的 HFSS 培训教程套装，可以帮助您从零开始，全面深入学习 HFSS 的各项功能和在多个方面的工程应用。购买套装，更可超值赠送 3 个月免费学习答疑，随时解答您学习过程中遇到的棘手问题，让您的 HFSS 学习更加轻松顺畅…

课程网址：http://www.edatop.com/peixun/hfss/11.html

CST 学习培训课程套装

该培训套装由易迪拓培训联合微波 EDA 网共同推出，是最全面、系统、专业的 CST 微波工作室培训课程套装，所有课程都由经验丰富的专家授课，视频教学，可以帮助您从零开始，全面系统地学习 CST 微波工作的各项功能及其在微波射频、天线设计等领域的设计应用。且购买该套装，还可超值赠送 3 个月免费学习答疑…

课程网址：http://www.edatop.com/peixun/cst/24.html

HFSS 天线设计培训课程套装

套装包含 6 门视频课程和 1 本图书，课程从基础讲起，内容由浅入深，理论介绍和实际操作讲解相结合，全面系统的讲解了 HFSS 天线设计的全过程。是国内最全面、最专业的 HFSS 天线设计课程，可以帮助您快速学习掌握如何使用 HFSS 设计天线，让天线设计不再难…

课程网址：http://www.edatop.com/peixun/hfss/122.html

13.56MHz NFC/RFID 线圈天线设计培训课程套装

套装包含 4 门视频培训课程，培训将 13.56MHz 线圈天线设计原理和仿真设计实践相结合，全面系统地讲解了 13.56MHz 线圈天线的工作原理、设计方法、设计考量以及使用 HFSS 和 CST 仿真分析线圈天线的具体操作，同时还介绍了 13.56MHz 线圈天线匹配电路的设计和调试。通过该套课程的学习，可以帮助您快速学习掌握 13.56MHz 线圈天线及其匹配电路的原理、设计和调试…

详情浏览：http://www.edatop.com/peixun/antenna/116.html

我们的课程优势：

※ 成立于 2004 年，10 多年丰富的行业经验，

※ 一直致力并专注于微波射频和天线设计工程师的培养，更了解该行业对人才的要求

※ 经验丰富的一线资深工程师讲授，结合实际工程案例，直观、实用、易学

联系我们：

※ 易迪拓培训官网：http://www.edatop.com

※ 微波 EDA 网：http://www.mweda.com

※ 官方淘宝店：http://shop36920890.taobao.com