Short Papers

Comparison of Iterative Methods for AC Analysis in PISCES-IIB

Deodatta R. Apte and Mark E. Law

Abstract—**This letter describes the implementation of an improved small-signal ac simulation capability in the general-purpose device simulator PISCES-IIB. The preconditioned generalized conjugate algorithm has been implemented, which allows ac simulations to be performed up to any frequency without convergence problems, although at great computational expense. The current implementation of PISCES-IIB uses the block successive overrelaxation algorithm for ac simulations. This algorithm fails at high frequencies, thus making it impossible to determine accurately the cutoff frequency and switching speed of high-frequency devices. The preconditioned GCR algorithm has been implemented using two different preconditioners. A comparison of the three methods shows that they are most efficient at different frequency ranges, which allows programming of an automatic switching algorithm which chooses the most efficient simulation method.**

I. Introduction

This letter describes the implementation of an improved small-signal ac simulation capability in PISCES-IIB. This improvement, although computationally expensive, allows simulation to any frequency without convergence problems.

Small-signal ac simulation is an important part of the device simulation process. It allows the determination of the frequency response of the device, which is used in both analog and digital design. Since the switching speed of a digital circuit is perhaps the most important parameter determining performance, its accurate determination using simulation is important. The frequency response obtained from small-signal ac simulation allows this determination. The ac simulation also allows the determination of device model parameters to be used in circuit simulations.

PISCES-IIB [1] is a general-purpose two-dimensional device simulator that solves the following system of equations on an irregular triangular grid of mesh points:

$$\epsilon \nabla^2 \psi + q(p - n + N_D^+ - N_A^-) - \rho_F = 0 \quad (1)$$

$$\frac{1}{q} \nabla \cdot J_n - U - \frac{\partial n}{\partial t} = 0 \quad (2)$$

$$-\frac{1}{q} \nabla \cdot J_p - U - \frac{\partial p}{\partial t} = 0 \quad (3)$$

where ψ is the electric potential, n is the electron concentration, p is the hole concentration, J_n and J_p are the electron and hole currents formulated as the sum of drift and diffusion components, ρ_F is the fixed charge, and U is the recombination rate. Equation (1) is Poisson's equation, (2) is the electron continuity equation, and (3) is the hole continuity equation. For dc simulations, these equations are discretized at the mesh points to yield a nonlinear system of equations:

$$F(x) = 0. \quad (4)$$

This system of equations is solved by using the Newton–Raphson iterative method. Equation system (4) is linearized to the following system of equations:

$$J(x^k) \cdot (x^{k+1} - x^k) = -F(x^k) \quad (5)$$

where J is the Jacobian matrix of the equation system (4). Superscript k denotes the Newton–Raphson iteration number. The linearized system (5) is solved directly by LU decomposition, resulting in the next guess of the solution vector x, which is used to reassemble the Jacobian for the next Newton–Raphson iteration. Convergence is obtained when the RHS in (5) is minimized.

II. Iterative Methods for AC Analysis

The small-signal ac simulation involves solving the linear equation system [2]:

$$\begin{bmatrix} J & -D \\ D & J \end{bmatrix} \begin{bmatrix} X_R \\ X_I \end{bmatrix} = \begin{bmatrix} B \\ 0 \end{bmatrix} \quad (6)$$

where J is the Jacobian matrix that already exists from the dc solution, D is a diagonal matrix that is the contribution of the frequency-dependent parts of the hole and electron continuity equations, X_R and X_I are the real and imaginary parts of the solution vector (the values of ψ, n, p at each mesh point), and the RHS is the stimulus to the device. Since only real sources are allowed by PISCES-IIB, the imaginary part of the stimulus is zero.

As currently implemented in PISCES-IIB, this system of equations is solved by using the block successive overrelaxation (SOR) iterative method, as described in [2]. The block SOR method alternates between the following two equations until convergence is obtained:

$$X_R^{k+1} = (1 - \omega_R)X_R^k + \omega_R J^{-1}(DX_I^k + B) \quad (7)$$

$$X_I^{k+1} = (1 - \omega_R)X_I^k + \omega_R J^{-1}(-DX_R^{k+1}) \quad (8)$$

where superscript k denotes the iteration number. The initial guess $X_I^0 = 0$ is used to start the iteration. The SOR relaxation parameter is ω_R. The Jacobian is already factorized into LU parts available in memory as part of the dc solution, and thus is very easy to invert. The SOR method is very efficient at frequencies less than $f_T/10$ but fails to converge at higher frequencies. Thus, to improve the range of possible simulations, the preconditioned generalized conjugate residual (GCR) algorithm was implemented in PISCES-IIB.

The GCR algorithm [3] is one of the class of algorithms that use the method of steepest descent to converge to a solution. The GCR algorithm is used for solving systems of equations with nonsymmetric coefficient matrices. In this particular implementation, the preconditioned GCR algorithm was implemented, since it was found that the nonpreconditioned algorithm failed to converge for the simulations considered.

Manuscript received September 7, 1990. This work was supported by SRC under Contract 89-SP-087. This paper was recommended by Associate Editor D. J. Rose.

The authors are with the Department of Electrical Engineering, University of Florida, Gainesville FL 32611.

IEEE Log Number 9104582.

Consider the linear system

$$A \cdot x = b \tag{9}$$

where A is a nonsymmetric matrix. Using preconditioning, the system of equations (9) is modified as

$$\tilde{A}x = [Q^{-1}A] \cdot [x] = [Q^{-1}b] = \tilde{b} \tag{10}$$

where Q is a nonsingular matrix. If systems of equations with Q as the coefficient matrix can be solved easily, then the use of Q as a preconditioner greatly improves convergence of the GCR algorithm, without significant computational overhead.

With the preconditioning as above, the GCR algorithm becomes:

```
Choose $x_0$ (initial guess).
Compute $r_0 = b - Ax_0$.
Set $p_0 = r_0$.
FOR $i = 0$ STEP 1 UNTIL Convergence DO
```

$$\alpha_i = \frac{(r_i \cdot Ap_i)}{(Ap_i \cdot Ap_i)} \tag{11}$$

$$x_{i+1} = x_i + \alpha_i p_i \tag{12}$$

$$r_{i+1} = r_i - \alpha_i Ap_i \tag{13}$$

$$\beta_j^{(i)} = -\frac{(AQ^{-1}r_{i+1} \cdot Ap_j)}{(Ap_j \cdot Ap_j)}, \quad j \leq i \tag{14}$$

$$p_{i+1} = Q^{-1}r_{i+1} + \sum_{j=0}^{i} \beta_j^{(i)} \cdot p_j \tag{15}$$

$$Ap_{i+1} = AQ^{-1}r_{i+1} + \sum_{j=0}^{i} \beta_j^{(i)} \cdot Ap_j \tag{16}$$

The above method of calculating α_i minimizes $\|b - A(x_i + \alpha p_i)\|_2 = \|r_{i+1}\|_2$ as a function of α, so that the Euclidean norm of the residual decreases at each step. Convergence is determined by comparing $\|r_{i+1}\|_2$ to a tolerance parameter.

The above method requires storing the values of the p_j vectors calculated during the previous iteration steps. To save memory, only five past vectors are stored, and after five iterations the p_j values are stored over the old values. Thus the maximum number of terms in the summation in (15) and (16) is 5.

Note that the matrix products Ap_j are calculated at the same time as the p_j vectors. Thus the only matrix multiply required is for calculating $AQ^{-1}r_{i+1}$. The $Q^{-1}r_{i+1}$ product is not computationally intensive since the preconditioner matrix Q is chosen so that its inverse is easy to calculate.

In this implementation of the preconditioned GCR algorithm, four different preconditioners were considered. The first one was the identity matrix, i.e., no preconditioning at all. This method failed to converge for any frequency and was thus useless.

The second preconditioner considered is the matrix

$$Q = \begin{bmatrix} J & 0 \\ 0 & J \end{bmatrix} \tag{17}$$

where J is the Jacobian matrix of the dc solution. The Jacobian is already decomposed into the L and U triangular matrices as part of the dc solution; thus it is easy to invert the Q matrix, involving two forward-backward substitutions. Also, the Jacobian matrix J is frequency independent, so that the LU factorization is valid at any frequency.

The next preconditioner considered is the matrix

$$Q = \begin{bmatrix} J & 0 \\ D & J \end{bmatrix} \tag{18}$$

which needs slightly more computation time to invert than (17). However, it was found that using (18) does not offer any advantages over (17), since both algorithms fail to converge at about the same frequency. Thus, the GCR algorithm using (18) was not implemented into PISCES-IIB.

It was observed that the preconditioned GCR algorithm using (17) or (18) as preconditioners failed to converge at frequencies about four times the frequencies at which the SOR method fails to converge. This may or may not be sufficient for the simulation requirements. Thus it is necessary to provide a method which will converge at higher frequencies. This is implemented by considering the next preconditioner, namely the matrix

$$Q = \begin{bmatrix} J & -D \\ D & J \end{bmatrix}$$

which is the same as the A matrix of the system of equations to be solved. This matrix, being the preconditioner, needs to be inverted in the GCR algorithm. This is done by factoring (19) into LU form before the start of the GCR algorithm and then using forward-backward substitution during the GCR iterations. The LU factorization is the most computationally intensive step. Thus to minimize computation time, the same LU factorization is used for different frequency steps in the PISCES-IIB simulations. Note that the D parts of the matrix are frequency dependent; hence the LU factorization will be incorrect at a different frequency. But since (19) is used only as a preconditioner, convergence is still possible. Thus the correct A matrix is used, together with an LU factorized preconditioner that was computed at a lower frequency, in the GCR algorithm to obtain convergence. This will fail to converge when the frequency difference between the A matrix and the preconditioner is too great, at which point the LU factorization is performed again. It is observed that when the frequency of the A matrix is about three to four times the frequency at which the preconditioner was factorized, the GCR algorithm fails to converge. At this point, the preconditioner is assembled and factored again. When the frequency of the A matrix matches that of the LU factorized preconditioner, effectively the problem has already been solved, and the GCR algorithm converges in one iteration.

Thus, using the third preconditioner, it is possible to obtain ac simulations at any frequency. This method is computationally intensive, and is applied only when the SOR method and the GCR method using (17) as preconditioner have both failed. This switching of methods is done automatically.

III. Results

The results obtained, convergence time of the various methods versus frequency, are shown in Fig. 1 and Fig. 2. The program runs on an Ardent Titan vector processor and is optimized for vector processing. All algorithms have a maximum iteration limit of 25.

In Fig. 1, the device simulated is a typical bipolar transistor with 666 mesh points and 1221 mesh elements. The bias applied is $V_{be} = 0.85$ V. In Fig. 2, the device is a MOSFET with 596 mesh points and 1094 mesh elements, with $V_t = 1.75$ V and applied bias of $V_{gs} = 2.5$ V.

As can be seen, the SOR algorithm is the most efficient at low frequencies. It fails to converge, for this particular example, beyond 800 MHz for the BJT and 400 MHz for the MOSFET. The GCR with (17) as preconditioner fails to converge at about 4 GHz for the BJT and 2 GHz for the MOSFET.

For the preconditioner of (19), as can be seen in the plots, the frequency at which the LU factorization is done takes of the order

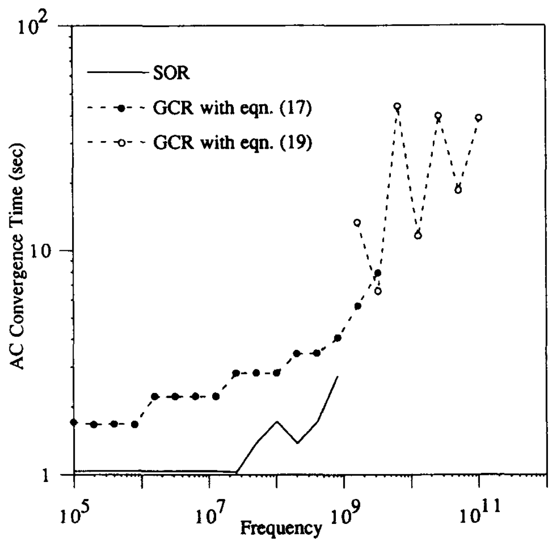

Fig. 1. BJT example.

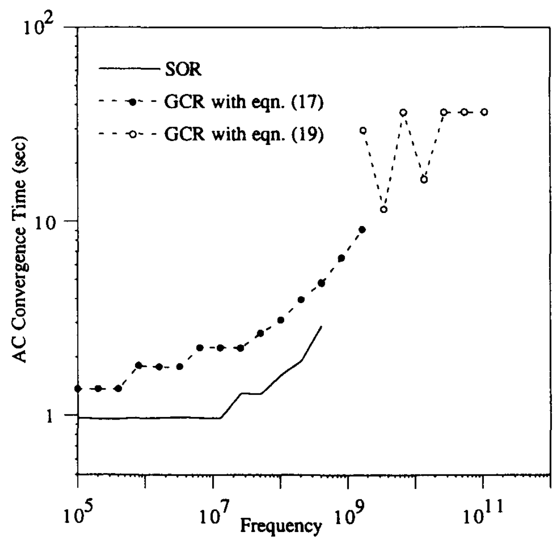

Fig. 2. MOSFET example.

of 30 s to solve. Most of this time is due to the factorization. At the next frequency step, which is twice the previous frequency, the convergence is faster. But as the frequency increases, it is not possible to use the old factorization and it is necessary to refactor (19) at the next frequency step, increasing the computation time. Ultimately, as the frequency increases to very high values, the preconditioner factorization from the previous frequency step does not yield convergence, and the factorization needs to be done at each frequency step. At this point, effectively the equation system is being solved directly at each frequency step, and the computation time is constant at each step. This can be clearly seen in the last three data points for the MOSFET. However, the frequency at which this happens is about 20 GHz, which is above the highest frequency of interest for this device.

The simulation has been run up to 100 GHz, which is well beyond the maximum frequency of interest for these devices. This method can be applied indefinitely. Therefore an option has been implemented which allows the user to set a maximum CPU time to be spent on ac analysis for each simulation (in addition to the number of frequency steps specified by the user), so that computation time can be limited.

Thus the final implementation of the small signal ac simulation is as follows:

1) Apply SOR till it fails to converge. SOR is the least computationally intensive method; hence it is applied first.
2) After SOR fails to converge, an automatic switch is made to the preconditioned GCR algorithm with the preconditioner given by (17). This algorithm fails to converge at about an order of magnitude higher than the SOR algorithm.
3) After the second algorithm fails to converge, an automatic switch is made to using the preconditioner in (19). This preconditioner, although accurate at only one frequency, can be used at other frequencies if the frequency difference is not too great. If convergence is not obtained, a refactorization of the preconditioner at the proper frequency is necessary.

IV. Conclusions

The preconditioned generalized conjugate residual algorithm has been implemented in PISCES-IIB for small-signal ac simulations. Two different preconditioners were implemented. This algorithm improves the small-signal ac simulation capability of PISCES-IIB, so that simulations can be performed at any frequency without convergence problems. A comparison of the three algorithms considered shows that they are efficient in different frequency ranges. This allows the programming of an automatic switching scheme by which the most efficient algorithm is chosen for ac simulation, minimizing computation time.

References

[1] M. R. Pinto *et al.*, "PISCES-IIB supplementary report," Stanford Electronics Laboratories Technical Report, Stanford University, 1985.

[2] S. E. Laux, "Techniques for small-signal analysis of semiconductor devices," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 472–481, Oct. 1985.

[3] H. C. Elman, "Preconditioned conjugate-gradient methods for nonsymmetric systems of linear equations," Department of Computer Science, Yale University, Research Report 203, Apr. 1981.