# Transient Simulation of Silicon Devices and Circuits

RANDOLPH E. BANK, WILLIAM M. COUGHRAN, JR., WOLFGANG FICHTNER, SENIOR MEMBER, IEEE, ERIC H. GROSSE, DONALD J. ROSE, AND R. KENT SMITH

*Abstract*—**In this paper, we present an overview of the physical principles and numerical methods used to solve the coupled system of nonlinear partial differential equations that model the transient behavior of silicon VLSI device structures. We also describe how the same techniques are applicable to circuit simulation. A composite linear multistep formula is introduced as the time-integration scheme. Newton-iterative methods are exploited to solve the nonlinear equations that arise at each time step. We also present a simple data structure for nonsymmetric matrices with symmetric nonzero structures that facilitates iterative or direct methods with substantial efficiency gains over other storage schemes. Several computational examples, including a CMOS latchup problem, are presented and discussed.**

## I. INTRODUCTION

THE continuing advances in the microelectronics area are closely linked to the ability to put an ever increasing number of devices on a silicon chip. This increase in packing density can only be achieved by a reduction of the feature sizes of the individual building blocks, such as the transistors, wires, contacts, etc. Recent very large scale integrated (VLSI) projects such as the 1-Mbit dRAM and 32-bit microprocessors use design rules with minimum feature sizes smaller than 1 $\mu$m. By the end of the decade, circuits with 0.5-$\mu$m design rules will be fabricated. In the laboratory, MOS devices with gate lengths as small as 0.25 $\mu$m have already been fabricated [1].

The development of future devices with submicron feature sizes relies heavily on the use of numerical simulation programs. These programs, if applied properly, can have a significant impact by drastically reducing the amount of time necessary to optimize the device design and the technology.

The state-of-the-art of numerical device simulation has advanced significantly in the last few years. In the past, most simulations have concentrated on the behavior of the active device regions of an integrated circuit (IC), such as a MOSFET or a bipolar transistor. The continuing reduction in size of the active device regions, however, makes it more and more difficult to ignore the surrounding parasitic components. Examples are the source/drain regions of a MOSFET with the parasitic resistance and capacitance components of the contact and junction areas. As a result, device simulators are now routinely applied to analyze complex device structures such as complete inverter structures and parasitic bipolar devices in CMOS technology.

The influence of parasitics becomes important if one is to analyze the switching behavior of densely packed integrated circuits. Any realistic calculation of the transient behavior of a MOSFET has to include the capacitive effects associated with junctions areas and conduction wires [2].

This paper summarizes our approach to the numerical simulation of complex two-dimensional silicon device structures under transient operating conditions. We also describe how the same techniques are applicable to circuit simulation.

In Section II, we briefly outline the physical principles and basic equations that govern the behavior of electrons and holes in the presence of time-varying fields. We restrict our discussion to silicon devices. We also include some observations on the validity of the basic equations for the transient modeling problem. Finally, we mention the form of the equations that occur in circuit simulation.

The numerical techniques used in our work are described in Section III. We introduce a composite linear multistep method for time integration, which has several useful properties. We discuss the use of a Newton–Richardson iteration to solve the nonlinear equations that arise at each time step efficiently. Finally, we describe a novel data structure that takes advantage of nonsymmetric matrices with symmetric nonzero structures, which occur naturally in device and circuit simulation.

Section IV contains several representative examples of actual simulations, including the switching of a submicron MOSFET under realistic loading conditions and latchup triggering in CMOS device structures.

Section V offers some conclusions.

## II. BASIC EQUATIONS

### *2.1. Physics of Transport in Time-Varying Fields for Semiconductors*

The applicability of numerical simulators to optimize the behavior of device structures with active dimensions of 0.5 $\mu$m or less (for example, MOS channel lengths) depends critically on the physical assumptions "built" into the programs.

In the following, we give a summary of the physical foundation that constitutes the basis for most simulation programs in use today. We restrict our discussion to devices built on silicon substrates. For relevant results con-

Manuscript received June 11, 1985.

R. E. Bank is with the Mathematics Department, University of California, San Diego, La Jolla, CA 92093.

W. M. Coughran, Jr., W. Fichtner, E. H. Grosse, and R. K. Smith are with AT&T Bell Laboratories, Murray Hill, NJ 07974.

0278-0070/85/1000-0436$01.00 © 1985 IEEE

cerning III–V devices, we refer the reader to the literature [3].

For devices of interest, the theory of semiclassical transport, based on the Boltzmann Transport equation (BTE), is adequate. The basic concept underlying the BTE formalism is the assumption of a distribution function $f(\boldsymbol{x}, \boldsymbol{P}; t)$ that represents the density of charge carriers (electrons and holes) in the six-dimensional phase space with space coordinates $\boldsymbol{x}$ and momenta $\boldsymbol{P}$.

The validity of the BTE depends on the various time scales in the device under study. For silicon devices of current and future interest, it can be assumed that $\tau_c << \tau << \tau_d$ where $\tau_c$ is the duration of a collision process, $\tau$ is the mean free time between collisions, and $\tau_d$ is the transit time through the active device region. Even for today's most advanced circuits, this transit time is considerably shorter than the clock frequencies used. A rough estimate shows that for a silicon MOSFET with an active channel length of 0.25 $\mu$m, the transit time from source to drain is around 5 ps. For circuits built with these devices, we can expect clock frequencies around 10 GHz.

The rigorous treatment of dynamic carrier flow using the BTE is well beyond present computing capabilities. Recently, several attempts have been made to obtain accurate solutions to the BTE under some simplifying assumptions. These calculations are typically based on Monte Carlo calculations. A good survey of the state of the art in this field can be found in this issue [4].

Results for the dynamic behavior of electrons and holes can be obtained if one takes moments of the BTE [5] and applies the relaxation term approximation to the Boltzmann equation. In the case of electrons, this leads to

$$\frac{\partial(n\boldsymbol{P})}{\partial t} = nq\boldsymbol{E} - \frac{n\boldsymbol{P}}{\tau_m} \tag{1a}$$

$$\frac{\partial}{\partial t}\left[\frac{nP^2}{2} + \frac{3}{2}nkT_e\right] = nq\boldsymbol{v}\cdot\boldsymbol{E} - \frac{nkT_e}{\tau_e} \tag{1b}$$

where $n$ is the electron density, $\boldsymbol{P}$ is the electron momentum, $q$ is the electronic charge, $\boldsymbol{E}$ is the electric field, $T_e$ is the electron temperature, $k$ is the Boltzmann constant, and $\tau_m$ and $\tau_e$ are the momentum and energy relaxation times, respectively.

The dynamic behavior of electrons is governed by the combined effects of momentum and energy relaxation, as expressed in (1a) and (1b). In a sufficiently intense electric field $\boldsymbol{E}$, the drift velocity of electrons asymptotically approaches the value

$$\boldsymbol{v} = -\frac{q\tau_m\boldsymbol{E}}{m} \tag{2}$$

where $m$ is the effective mass of the electron. This maximum velocity is never achieved, however, due to the rise in electron temperature in the high electric field. The increase in $T_e$ decreases the average values of the collision time and the drift velocity. The difference in the magnitude of the momentum and energy relaxation times results

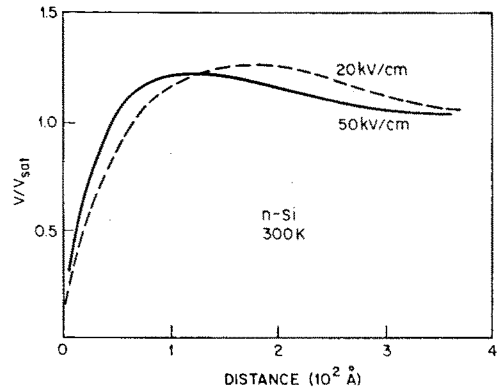



Fig. 1. Average velocity as a function of distance for electrons.

in an actual overshoot of the drift velocity over its steady-state value. For silicon devices, this phenomenon is of minor importance. Fig. 1 shows the results of a calculation [6] of the average velocity as a function of distance for electrons in two electric fields of 20 and 50 kV/cm. The critical dimension is on the order of 200–300 Å at room temperature, a value considerably shorter than the most advanced devices today.

Under the condition of transients that vary slowly compared to the transit time, (1) can be further simplified. This leads to continuity equations of the form

$$\frac{\partial n}{\partial t} + \nabla\cdot(n\boldsymbol{v}) = G - R. \tag{3}$$

As Blotekjaer has shown, the velocity $\boldsymbol{v}$ can be written as

$$\boldsymbol{v} = -\mu(E)\boldsymbol{E} - \frac{D(E)}{n}\nabla n \tag{4a}$$

or

$$\boldsymbol{v} = -\mu(E)\boldsymbol{E} - \frac{1}{n}\nabla\,[D(E)n] \tag{4b}$$

where the mobility $\mu$ and the diffusivity $D$ are viewed as local quantities and instantaneous functions of the electric field.

The question of whether (4a) or (4b) is correct has been discussed in [5]. In our work, we have used (4a). It constitutes the usual drift-diffusion transport equation that has been successfully used in device modeling for the last two decades.

### 2.2. *Semiconductor Equations*

On the basis of the arguments given above, we can write the basic equations of semiconductor transport in the following form.

The *Poisson* equation

$$\epsilon_s\nabla\cdot\boldsymbol{E} = -\epsilon_s\nabla^2\psi = \rho \tag{5}$$

relates the total space charge $\rho$ to the divergence of the electric field $\boldsymbol{E}$, which defines the electrostatic potential $\psi$ as

$$\boldsymbol{E} = -\nabla\psi. \tag{6}$$

Under the assumption of total ionization, the total space charge $\rho$ is given by

$$\rho = -q(n - p + N) \tag{7}$$

where $N = N_D^+ - N_A^-$ is the net concentration of electrically active impurities, $q$ is the electronic charge, and $n$ and $p$ are the electron and hole densities, respectively.

The connection between the behavior of the carrier densities and the electric field is given by the *current equations* for electrons and holes (see (3) and (4a)):

$$J_n = q\mu_n nE + qD_n \nabla n \tag{8a}$$

$$J_p = q\mu_p pE - qD_p \nabla p \tag{8b}$$

where $\mu_n$ and $\mu_p$ are the electron and hole mobilities, respectively, and $D_n$ and $D_p$ are the corresponding diffusion coefficients. Both mobilities and diffusion coefficients depend on the temperature, the doping level, and the electric field.

Under the assumption of Boltzmann statistics, the electron and hole concentrations can be written as

$$n = n_{ie} \exp\left[\frac{q(\psi - \phi_n)}{kT}\right] \tag{9a}$$

$$p = n_{ie} \exp\left[\frac{q(\phi_p - \psi)}{kT}\right] \tag{9b}$$

where we have defined the *quasi-fermipotentials* $\phi_n$ and $\phi_p$ [7]. The factor $n_{ie}$ is the effective intrinsic carrier concentration [8]. For low doping, $n_{ie}$ approaches the intrinsic carrier concentration $n_i$.

Assuming the *Einstein* relation [9] for both electrons and holes

$$D = \mu \frac{kT}{q}. \tag{10}$$

Equation (8) can be rewritten using (9) as

$$J_n = -q\mu_n n \nabla \phi_n \tag{11a}$$

$$J_p = -q\mu_p p \nabla \phi_p. \tag{11b}$$

The *continuity equations* for electrons and holes are then given by

$$-\frac{1}{q} \nabla \cdot J_n - G + R + \frac{\partial n}{\partial t} = 0 \tag{12a}$$

$$\frac{1}{q} \nabla \cdot J_p - G + R + \frac{\partial p}{\partial t} = 0 \tag{12b}$$

where $G$ and $R$ represent generation and recombination processes, respectively.

In the time-dependent case, the equation of total-current continuity couples the change in electric-field strength to the current densities. It is given by

$$\nabla \cdot J_T = \nabla \cdot \left(\epsilon \frac{\partial E}{\partial t} + J_n + J_p\right) = 0 \tag{13}$$

where $J_T$ is the total current, which consists of both the conduction components $J_n$ and $J_p$ and the displacement current $\epsilon \partial E/\partial t$. Equation (13) can be derived from the time derivative of (5) combined with (12).

The boundary conditions for semiconductor devices are given by neutrality and equilibrium conditions, namely,

$$pn = n_{ie}^2 \tag{14}$$

and

$$n - p + N = 0. \tag{15}$$

Thus at a Dirichlet contact, the three potentials are

$$\psi = V_c(t) + V_{bi} \tag{16a}$$

$$\phi_n = \phi_p = V_c(t) \tag{16b}$$

where $V_{bi}$ and $V_c(t)$ are the built-in and contact voltages, respectively. When external circuit elements are applied to the device, the contact voltage becomes an unknown and is given by

$$\frac{1}{R}[V_c(t) - V_a(t)] + C\frac{d}{dt}[V_c(t) - V_a(t)] + \int_\Gamma \nabla \cdot J \, dl = 0 \tag{17}$$

where $R$, $C$, and $V_a(t)$ are the resistance, capacitance, and applied voltage, respectively, and $\Gamma$ is an appropriate contour surrounding the contact.

The complete set of semiconductor equations is given by (5), (8), and (12) combined with appropriate initial and boundary conditions. Together they form a system of coupled, nonlinear, partial differential equations (PDE's), which are usually written in dimensionless form. We have followed the work of de Mari [10] by normalizing all spatial dimensions to the intrinsic *Debye* length $L_i = \sqrt{\epsilon_s kT/qn_i}$, all densities to the intrinsic concentration $n_i$, and all voltage terms to $kT/q$. Hence, we will make use of the normalized electrostatic potential $u$ and the normalized quasi-Fermi levels $v$ and $w$ of electrons and holes, respectively.

Once, the equations are spatially discretized [11], [12], the time-dependent problem is governed by an implicit system of ordinary differential equations

$$\frac{d}{dt} q(z(t)) + f(t, z(t)) = 0 \tag{18}$$

where

$$z = \begin{pmatrix} u \\ v \\ w \end{pmatrix}. \tag{19}$$

$\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{w}$ are now grid functions representing discrete spatial values at time $t$, and

$$q(z) = \begin{pmatrix} 0 \\ e^{u-v} \\ e^{w-u} \end{pmatrix}. \tag{20}$$

This system is a simple differential-algebraic system since $\partial q/\partial z$ is singular, which follows from the time-independence of the $u$ equation. (See [13]–[15] for detailed discussions of differential-algebraic equations.)

### 2.3. Circuit Equations

A circuit is governed by Kirchhoff's current and voltage laws as well as constitutive relations. The current law is

$$Ai = 0 \tag{21}$$

where $A \in R^{n \times m}$ is the circuit's reduced-incidence matrix [16] and $i \in R^m$ is the vector of branch currents. The voltage law implies that

$$v = A^T u \tag{22}$$

$$\mathbf{M}_1 = \begin{bmatrix} I & 0 & K_{v1} A_1^T \\ 0 & I & K_{vR} A_R^T \\ A_1 & A_R & 0 \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ A_1 & 0 & I \end{bmatrix} \begin{bmatrix} I & 0 & K_{v1} A_1^T \\ 0 & I & K_{vR} A_R^T \\ 0 & A_R & -A_1 K_{v1} A_1^T \end{bmatrix} \tag{29}$$

where $u \in R^n$ and $v \in R^m$ are the node and branch voltages, respectively. The constitutive relations are of the form

$$K(i, v) \equiv i - \left(\frac{d}{dt} q(v) + f(v)\right) = 0 \in R^m \tag{23}$$

for voltage-controlled elements that are "quasi-static."

If the constitutive relations are linearized about a point, the resulting equations are

$$\delta i + K_v \, \delta v = s \tag{24}$$

where $K_v \equiv \partial K/\partial v \in R^{m \times m}$ and $s$ is a source vector. The circuit equations can be summarized in matrix notation as

$$M_1 z \equiv \begin{bmatrix} I & K_v A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \delta i \\ \delta u \end{bmatrix} = t \tag{25}$$

which makes use of the substitution $v = A^T u$ [17], [18]. $M_1$ is the reduced-tableau matrix.

In particular, if the quasi-static approximation is assumed, the currents of a transistor $T_1$ are given by

$$i_1(t) = \frac{d}{dt} q(v_1(t)) + f(v_1(t)) \in R^3 \tag{26}$$

where the components of $\mathbf{i}_1$ are the currents associated with the source, gate, and drain terminals and $v_1 = (u_d - u_s, u_g - u_s, u_b - u_s)^T \in R^3$ represents the appropriate differences of terminal (node) voltages. Only three currents enter into the constitutive relation because Kirchhoff's current law insures that $T_1$'s four terminal currents sum to zero. Hence, the portion of the reduced-incidence matrix associated with $T_1$ is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & -1 \end{bmatrix} \tag{27}$$

where the three columns correspond to the dependency edges introduced by the voltage differences.

We can write $A = [A_1 \;\; A_R]$ where $A_1$ and $A_R$ correspond to $T_1$ and the remainder of the circuit, respectively. We can partition $u$, $i$, and $K$ similarly as $u^T = (u_1^T, u_R^T)$, $i^T = (i_1^T, i_R^T)$, and

$$K_v = \begin{bmatrix} K_{v1} & 0 \\ 0 & K_{vR} \end{bmatrix} \tag{28}$$

where $u_1 \in R^4$ represents $T_1$'s terminal voltages and $K_{v1}$ is $T_1$'s linearized constitutive relation. Then the reduced-tableau matrix can be formally block factored as (29) which forms a small part of the usual nodal matrix in the lower right block; this process is repeated to assemble the transistors into the nodal matrix [18]. Capacitors and resistors can be treated in the same fashion. We will restrict our attention to these elements and grounded voltage sources.

In summary, the assembly of the individual circuit elements into global circuit equations yields

$$\frac{d}{dt} q(u) + f(t, u) = 0 \tag{30}$$

where the explicit time-dependence of $f$ comes from the boundary conditions, that is, grounded voltage sources. In general, $q' \equiv \partial q/\partial u$ is singular so this represents a differential-algebraic system, possibly of nontrivial index [19].

## III. Numerical Procedures

We will concentrate our exposition on the semiconductor equations but we will note where the circuit-simulation problem differs.

For a discussion of numerical methods suitable to static device and circuit simulation see [11], [20], [18], [12] (and also [21]). Some of these techniques are important for the time-dependent case and are briefly reviewed below. We will not discuss the spatial discretization used but the reader should be aware that it is based on the box method over a triangulated domain.

### 3.1. Time-Stepping Scheme

We make use of a trapezoidal rule/backward-differentiation-formula (TR-BDF2) composite method. Consider integrating (18) (or (30)) from $t = t_n$ to $t_{n+1} \equiv t_n + h_n$. We apply TR to go from $t = t_n$ to $t_n + \gamma h_n$

$$2q_{n+\gamma} + \gamma h_n f_{n+\gamma} = 2q_n - \gamma h_n f_n. \tag{31}$$

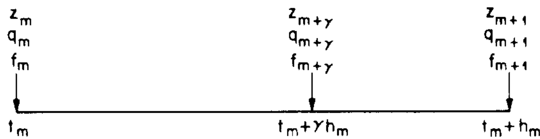


Fig. 2. Data layout for a single time step.

where $q_n \equiv q(z_n)$, etc. This implicit scheme has a Jacobian of the form $2q'_{n+\gamma} + \gamma h_n f'_{n+\gamma}$. We then apply the second-order backward-differentiation formula (BDF2) to go from $t = t_n + \gamma h_n$ to $t_{n+1}$

$$(2-\gamma)q_{n+1} + (1-\gamma)h_n f_{n+1} = \gamma^{-1} q_{n+\gamma} - \gamma^{-1}(1-\gamma)^2 q_n. \quad (32)$$

This implicit scheme has a Jacobian of the form $(2-\gamma)q'_{n+1} + (1-\gamma)h_n f'_{n+1}$. The data layout for the composite time step is sketched in Fig. 2.

The TR and BDF2 Jacobians have the same form if

$$\frac{2}{\gamma} = \frac{2-\gamma}{1-\gamma} \quad (33)$$

which implies $\gamma = 2 - \sqrt{2} \approx 0.59$. We assume this value of $\gamma$ for the remainder of this paper. As long as $q$ and $f$ vary slowly, it is possible to reuse Jacobian factorizations while still using quadratically convergent algorithms for the associated nonlinear equations (see the discussions of Newton–Richardson below).

A couple of remarks are relevant at this point. Consider applying a one-step method

$$y_{n+1} = A(h\lambda)y_n \quad (34)$$

where (hopefully) $y_n \approx y(nh)$, to the usual scalar test problem

$$\frac{dy}{dt} = \lambda y \quad (35)$$

with Re $\lambda < 0$. Recall that the one-step method is said to be A-stable if $|A(h\lambda)| < 1$ [22]; in other words, growing solutions are not allowed. Much is known about the properties of A-stable methods. For example, Dahlquist showed that the A-stable multistep method with the smallest local truncation error is the second-order TR.

A-stability may not be strong enough for extremely stiff problems. If TR is applied to (35) with Re $\lambda << 0$, then successively computed values will tend to be of the same magnitude with opposite sign ($y_{n+1} \approx -y_n$) unless the temporal discretization parameter is very small; such restrictions on $h$ are what one wishes to avoid with stiff problems. In other words, TR solutions may decay at the wrong rate and, without proper error control, "ringing" effects may occur, which can be exacerbated by a nonlinear problem.

A one-step method, $y_{n+1} = A(h\lambda)y_n$, is said to be L-stable if it is A-stable and $|A(h\lambda)| \to 0$ as $|h\lambda| \to \infty$ [22]. BDF2 is known to be L-stable so there are no restrictions on $h$ for (35), unlike TR. However, BDF2 has a higher truncation error than TR and is more difficult to implement (restart) since it is not a one-step method.

The composite TR-BDF2 procedure is an easily restarted, second-order, one-step algorithm, which is nearly as simple to implement as TR. Furthermore, the scheme inherits the L-stability of BDF2, as shown below, which make its suitable for very stiff problems requiring moderate accuracy.

*Proposition:* The TR-BDF2 method ((31) and (32)) with $\gamma = 2 - \sqrt{2}$ is L-stable.

*Proof:* Let $z = h\lambda$. After some manipulation, the composite formula applied to (35) is found to be

$$y_{n+1} = \frac{[1 + (1-\gamma)^2]z + 2(2-\gamma)}{\gamma(1-\gamma)z^2 + (\gamma^2 - 2)z + 2(2-\gamma)} y_n = A(z)y_n. \quad (36)$$

Obviously, $|A(z)| \to 0$ as $|z| \to \infty$. Moreover, $A(z)$ does not have a pole for Re $z \le 0$ since the denominator has a double positive root at $z = 2/\gamma \approx 3.4$. Consider the case when $z$ is purely imaginary, that is, $z = ib$. Then

$$|A(ib)|^2 = \frac{[1 + (1-\gamma)^2]^2 b^2 + 4(2-\gamma)^2}{[2(2-\gamma) - \gamma(1-\gamma)b^2]^2 + (\gamma^2 - 2)^2 b^2} \le 1. \quad (37)$$

For any $\epsilon > 0$, there is an $R_\epsilon > 0$ such that $|A(z)| \le \epsilon$ for $\{z \,|\, |z| \ge R_\epsilon$ and Re $z \le 0\}$. The result follows from the maximum modulus principle. □

In order to regulate the stepsize $h_n$, we need to estimate the local truncation error (LTE). There are several alternatives, which are all based on the principle truncation error term. (In general error estimation for differential-algebraic systems is much more difficult than for simple ODE's [13], [23].)

After some manipulation of Taylor series, the principal truncation term for a step of TR from $t_n$ to $t_n + \gamma h_n$ followed by a step of BDF2 from $t_n + \gamma h_n$ to $t_{n+1}$ is found to be

$$Ch_n^3 q^{(3)}(\xi)$$

where

$$C = \frac{-3\gamma^2 + 4\gamma - 2}{12(2-\gamma)} \approx -0.04. \quad (38)$$

Note that $|C(\gamma)|$ is minimized when $0 < \gamma = 2 - \sqrt{2} \le 1$.

One possibility is to directly compute an estimate for the third derivative $q^{(3)}$. A divided difference of the three available $f$ values gives

$$\frac{h_n^2 q^{(3)}}{2} \approx (1-\gamma)^{-1} f_{n+1} - \gamma^{-1} \cdot (1-\gamma)^{-1} f_{n+\gamma} + \gamma^{-1} f_n. \quad (39)$$

Another possibility is to use polynomial extrapolation as a predictor and to estimate the error as the difference between the predicted and corrected values [24], [13], [23].

A final possibility is to use a second step of TR from $t = t_n + \gamma h_n$ to $t_{n+1}$

$$\frac{q_{n+1} - q_{n+\gamma}}{(1-\gamma)h_n} + \frac{f_{n+1} + f_{n+\gamma}}{2} = 0 \qquad (40)$$

and then employ the classical Milne's device [22] to approximate the LTE. The method composed of TR followed by another step of TR has a principal truncation term of the form $\tilde{C}h_n^3 q^{(3)}$, which is precisely what is required for Milne's device. The use of Newton–Richardson (see the next subsection) makes it possible to avoid factoring the Jacobian associated with the final TR step.

In our limited experience, we have found that the divided–difference estimator

$$\tau_{n+1} = 2Ch_n\,[\gamma^{-1}f_n - \gamma^{-1}(1-\gamma)^{-1}f_{n+\gamma} + (1-\gamma)^{-1}f_{n+1}] \approx Ch_n^3 q^{(3)} \qquad (41)$$

approximates the LTE reasonably well and is inexpensive to compute. Note that the error estimator gives an indication of the truncation error in terms of $q$; in the semiconductor case, the portion of $\tau$ associated with the potential $u$ is zero.

Given a per component LTE estimate, we can predict a new candidate stepsize $h^*$, expected to satisfy a specified error tolerance, by

$$h^* = h_n r^{-1/3} \qquad (42)$$

where

$$e_{n+1,i} = \epsilon_R |q_{n+1,i}| + \epsilon_A \qquad (43)$$

and

$$r^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\tau_{n+1,i}}{e_{n+1,i}}\right)^2 \qquad (44)$$

in the device-simulation case and

$$r = \frac{||\tau_{n+1}||_2}{||e_{n+1}||_2} \qquad (45)$$

is often used in the circuit-simulation case. Here the second subscript on $q$ represents the component and $\epsilon_R$ and $\epsilon_A$ are absolute- and relative-error parameters, respectively, while the 1/3 reflects the second-order nature of the scheme. In our device simulator, we do not sum over the trivial portion of $q$ associated with the potential $u$. Equation (44) represents a so-called root-mean-squared (RMS) norm that is better able to deal with the scaling problems associated with the semiconductor equations whereas (45) is a rather "greedy" norm that assumes a well-behaved solution. In other words, (44) reacts to large relative changes in small values, which occurs when a large part of a device is quiescent, but dominates $||q_{n+1}||_2$, while small regions of the device are active.

If $r \lesssim 1$ (say not more than 2), the LTE is acceptable; otherwise the step is repeated with $h_n \leftarrow 0.9h^*$, where the 0.9 is a "paranoia" factor. If for some reason the nonlinear equations cannot be solved in a small fixed number of iterations (see the next subsection), the step is repeated with $h_n \leftarrow h_n/2$. If the step is accepted, the next stepsize is taken as $h_{n+1} = \min(\theta_1 h^*, 2h_n)$ subject to minor adjustments as mentioned below; $\theta_1$ is taken to be 1 and 0.9 for the semiconductor and circuit problems, respectively. This last rule restricts the rate of increase in order to avoid stepsize oscillations.

Since input wave forms often have natural breakpoints that should be sampled exactly at the corners for graphical reasons, we have employed the device described by Gear [25], which further limits the stepsize. In particular, if the integration is to stop at $t_{\text{stop}}$, we take

$$h_{n+1} \leftarrow \frac{t_{\text{stop}} - t}{\lceil (t_{\text{stop}} - t)/h_{n+1} - \epsilon \rceil} \qquad (46)$$

where $\epsilon$ is a small multiple of the machine epsilon and $\lceil\ \rceil$ denotes the integer ceiling function.

Some circuit-analysis packages try to get away with much simpler stepsize control schemes. One choice is to cut back the stepsize when the previous time step took more than a certain number of Newton iterations and to increase the stepsize when the previous step took less than another certain number of Newtons. This approach tries to maintain a "reasonable" number of Newtons per time step. If this simple procedure is applied to a linear *RC* network, the Newton procedure will always converge in one iteration resulting in continual stepsize increases and, thereby, arbitrarily bad truncation errors. The problem of local stepsize control is thrust upon the user. Moreover, some device simulators employ a constant stepsize specified by the user, which the experiments in Section IV suggest must be costly.

### *3.2. Nonlinear Equation Solution*

The nonlinear equations that arise at each time step are solved with the damped-Newton scheme of Bank and Rose [26]. When solving $g(z) = 0$, the algorithm computes a change by

$$g_k' x_k = -g_k \qquad (47)$$

where $g_k' \equiv (\partial g(z)/\partial z)(z_k)$ and $g_k \equiv g(z_k)$, and then forms

$$z_{k+1} = z_k + s_k x_k \qquad (48)$$

where $0 < s_k \le 1$ is chosen to satisfy the sufficient decrease condition

$$1 - \frac{||g_{k+1}||}{||g_k||} > \epsilon_M s_k \qquad (49)$$

and $\epsilon_M$ is the machine epsilon. (One strategy for selecting $s_k$ is described in [18].) Note that extrapolation is employed to get an initial guess $z_0$ at each time step. We stop the Newton iteration if $||g_n|| \le \epsilon_R ||g_0||$ or $||x_n||/||z_n|| \le \epsilon_R/2$.

The Newton–Richardson (NR) procedure discussed by Bank and Rose [26] offers inexpensive quadratic convergence, especially for successive transient solution steps as the values are changing slowly. The NR scheme starts with a given splitting

$$g_k' = M_k - N_k \qquad (50)$$

where $||M_k^{-1}N_k|| = ||I - M_k^{-1}g_k'|| \le \rho < 1$ for all $k$. In our applications, $M_k$ represents the LU factors of an old

Jacobian. $x_k$ is found by an inner iteration given by

$$M_k(x_{km} - x_{k,m-1}) = -(g_k' x_{k,m-1} + g_k) \quad (51)$$

where $x_{k0} = 0$; $x_k = x_{km_k}$ for some $m_k$. The inner iteration is terminated when

$$\frac{||g_k' x_{km_k} + g_k||}{||g_k||} \leq \alpha \frac{||g_k||}{||g_0||} \quad (52)$$

where $0 < \alpha < 1$ is experimentally determined.

Newton–Richardson has the potential for substantial savings in CPU time, but requires roughly the same amount of storage as the full damped-Newton scheme where $g_k'$ is factored each time. NR uses additional residual computations, Jacobian (times a vector) multiplications, and backsolves with the LU factors represented by $M_k$ to save matrix factorizations. The NR iteration does not compute the solution to (47) exactly so more (outer) Newton iterations may be required. It is possible to decide to use NR or full Newton based on timings made at run time.

In a transient simulation, one $M_k$ will not work for all time steps, but the matrices $g_k'$ change less when the truncation error request is small. A fixed strategy of how often to factor can be used or the convergence rate of the inner iteration (equation (51)) can be estimated so as to do a factorization only when necessary. There is a small overhead cost associated with estimating the convergence rate of (51).

When the cost of forming $g_k$ and $g_k' x$ and backsolving with known LU factors is significantly less expensive than factoring $g_k'$, NR is superior to the full damped-Newton procedure. This is the case for the semiconductor problem because matrix factorizations dominate the computation (see the discussion in the next section). On the other hand, our experimental circuit simulator makes use of tensor-product variation-diminishing splines [27] for its transistor models and model evaluation dominates the computation time except for large (more than 1000 nodes) circuits.

### 3.3. Linear Equation Solution

Let $A = (a_{ij}) \in R^{n \times n}$. We assume A is sparse with a symmetric nonzero structure; that is, both $a_{ij}$ and $a_{ji}$ are to be treated as nonzero elements if $|a_{ij}| + |a_{ji}| > 0$. We further assume that the diagonal elements $a_{ii}$ are nonzero. For the sparse matrices arising from the semiconductor and circuit equations, $a_{ij} \neq 0$ typically implies that $a_{ji} \neq 0$, but not necessarily that $a_{ij} = a_{ji}$.

Our data structure, which is also employed in PLTMG [28], is a variant of the usual Yale Sparse Matrix Package (YSMP) data structure (IA, JA, A) for storing symmetric, positive-definite, sparse matrices [29]–[31]. Let $\eta$ be the number of nonzeroes in the strict upper triangle of $A$. Our data structure consists of a single integer array $JA$ of length $\eta + n + 1$ and a real array $A$ of length $\eta + n + 1$ if the matrix is symmetric, and $2\eta + n + 1$ if the matrix is nonsymmetric. Note that the symbol $A$ is used for both the original matrix and the array that contains the nonzero entries.

Let $r_i$ be the number of nonzeroes in the strict upper triangular part of row $i$ so

$$\eta = \sum_{i=1}^{n} r_i.$$

Then the entries of $JA$ and $A$ are defined as follows:

$$JA[1] = n + 2$$
$$JA[i+1] = JA[i] + r_i \quad \text{for } 1 \leq i < n$$
$$A[i] = a_{ii} \quad \text{for } 1 \leq i \leq n$$
$$A[n+1] = \begin{cases} -1, & \text{for diagonal matrices} \\ 0, & \text{for symmetric matrices} \\ \eta, & \text{for nonsymmetric matrices} \end{cases}$$
$$JA[k] = j \text{ (column index of } a_{ij}) \text{ for } JA[i] \leq k < JA[i+1], \quad 1 \leq i \leq n$$
$$A[k] = a_{ij} \text{ for } JA[i] \leq k < JA[i+1], \quad 1 \leq i \leq n$$

and if the matrix is not symmetric

$$A[k+\eta] = a_{ji} \text{ for } JA[i] \leq k < JA[i+1], \quad 1 \leq i \leq n.$$

$JA[1]$ through $JA[n+1]$ represent the same information as YSMP's IA array. The remainder of $JA$ corresponds to YSMP's JA array for symmetric matrices. In the $A$ array, the diagonal is stored first, followed by the strict upper triangle stored row-wise. If the matrix is not symmetric, then this is followed by the strict lower triangle stored column-wise. The matrix type is specified by $A[n+1]$. Since $A$ has a symmetric nonzero structure, the column indices for the upper triangle are identical to the row indices for the lower triangle and, hence, need not be duplicated.

As an example, let $n = 5$ and

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ a_{21} & a_{22} & 0 & a_{24} & 0 \\ a_{31} & 0 & a_{33} & a_{34} & a_{35} \\ 0 & a_{42} & a_{43} & a_{44} & 0 \\ 0 & 0 & a_{53} & 0 & a_{55} \end{bmatrix}. \quad (53)$$

Then

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***JA*** | 7 | 9 | 10 | 12 | 12 | 12 | 2 | 3 | 4 | 4 | 5 | | | | | |
| ***A*** | $a_{11}$ | $a_{22}$ | $a_{33}$ | $a_{44}$ | $a_{55}$ | 5 | $a_{12}$ | $a_{13}$ | $a_{24}$ | $a_{34}$ | $a_{35}$ | $a_{21}$ | $a_{31}$ | $a_{42}$ | $a_{43}$ | $a_{53}$ |
| | Diagonal | | | | | $\eta$ | Upper | | | | | Lower | | | | |

Compared to the YSMP scheme, this data structure saves $n$ integer storage locations if the matrix is symmetric and $n + \eta$ storage locations if it is nonsymmetric because we store the diagonal elements separately, avoiding indices, and take advantage of the symmetric nonzero structure. In addition to saving storage, this data structure allows certain economies in matrix algorithms. Some examples are discussed below.

Sparse matrix multiplication is fundamental to most iterative methods. To illustrate our data structure, consider the following matrix multiplication algorithm (where indention is used to denote nesting)

```
procedure spmult (n, JA, A, x, y)
    ishift ← A[n + 1]
    for i ← 1 to n
            y[i] ← A[i] x[i]
    if ishift ≥ 0
            lshift ← ishift
            ushift ← 0
            for i ← 1 to n
                    for k ← JA[i] to JA[i + 1] - 1
                            j ← JA[k]
                            y[i] ← y[i] + A[k + lshift]x[j]
                            y[i] ← y[j] + A[k + ushift]x[i]
```

This algorithm computes $y = Ax$ for a diagonal, symmetric, or nonsymmetric matrix $A$. In the nonsymmetric case, $y = A^T x$ may be computed by setting *lshift* $= 0$ and *ushift* $=$ *ishift*. Note that the overhead cost in indirect addressing is reduced since each reference to $JA$ for a column (row) index corresponds to two floating-point operations rather than one.

Many iterative methods access the lower triangle ($L$), upper triangle ($U$), and diagonal ($D$) entries of $A$ separately. With SSOR [32], for example, linear systems of the form $By = c$ where $B = (D + \omega L)D^{-1}(D + \omega U)$ occur. Since $D$, $L$, and $U$ are stored separately, it is easy to implement efficient algorithms for solving the required linear systems. The standard sparse data structure requires a (partial) ordering of the column indices for each row in the $JA$ array and an extra pointer array of length $n$ to allow easy access to $D$, $L$, and $U$ separately; otherwise column indices must be continually searched.

Consider a sparse factorization of the form

$$PAP^T = LDU \tag{54}$$

where $P$ is a permutation matrix, $D$ is diagonal, and $L$ and $U$ are unit lower and upper triangular, respectively. Such factorizations are known to be well defined when $A + A^T$ is positive definite. The permutation matrix $P$ is chosen by the minimum degree algorithm [30], [33]. The main disadvantage of this scheme is that, without pivoting, the numerical stability cannot be guaranteed for general nonsymmetric matrices.

Our data structure for storing the $LDU$ factorization consists of two arrays $JU$ and $U$, which are analogous to $JA$ and $A$. In the array $U$, the diagonal entries of $D$ are stored in the first $n$ locations, followed by the strict upper triangular part of $U$ stored row-wise. This is followed by the strict lower triangular part of $L$ stored column-wise in the nonsymmetric case. As before, $U[n + 1]$ is used to specify the matrix type. If there are $\nu$ nonzeros in the strict upper triangular portion of the matrix $U$, then the array $U$ is of length $n + 1 + \nu$ or $n + 1 + 2\nu$, depending on whether the storage of $L$ is required.

If $JU$ were exactly analogous to $JA$, then its length would be $n + 1 + \nu$ since row indices for $L$ need not be stored. However, considerable integer storage can be saved using a compressed storage scheme [30], which does not store duplicate sequences of column indices for the filled-in matrix. This requires $n$ extra pointers. In typical nonsymmetric PDE problems of moderate size, $JU$ in roughly 20–30 percent of the length of $U$, using the compressed scheme.

To compute the factorization, a sparse variant of the Crout reduction technique [34] is used. It is most efficient to compute the entries of the $i$th row of $U$ and the $i$th column of $L$ in temporary (expanded) vectors of length $n$ and, then, to transfer the final results to the array $U$, as

in the YSMP approach. The resulting procedure produces twice as many floating-point operations per indirect address (through $JU$) compared to the analogous YSMP routines. Hence, a significant fraction of the addressing overhead is saved. Once $A$ has been factored, the forward- and back-substitution (backsolve) algorithms are easily described in terms of our data structure; the same code can be used to solve either $Ax = b$ or $A^T x = b$.

The data structure described above may be easily extended to block matrices. Such matrices arise in solving systems of PDE's on the same grid and in finite element methods with several degrees of freedom associated with each grid point. The $JA$ array is constructed to reflect the symmetric nonzero structure of a single equation. We will consider two different partitions of the matrix $A$. In any case, the matrix multiplication and Crout algorithms are applicable, with scalar operations replaced by matrix operations.

For block iterative methods [32], [35], it is often convenient to partition the full matrix into $k^2$ block matrices. For example, the matrix associated with the semiconductor equations can be partitioned into a $3 \times 3$ block matrix with the upper left block corresponding to the Laplacian operator in the $u$ equation. Each block matrix will then have the sparse structure described by a single $JA$ with different values of $A[n + 1]$; moreover, one $JU$ is sufficient.

Another possibility is to treat $A$ as a sparse $n \times n$ matrix with dense $k \times k$ blocks. Each column index in $JA$ then refers to a $k \times k$ block in $A$. It is quite apparent that the economics of space and work will be larger with increasing $k$. For nonsymmetric systems with $n >> \eta$, $JA$ requires about $1/2k^2$ of the storage of $A$. For matrix multiplication, each indirect address now produces $2k^2$ floating-point operations.

We will now present some numerical experiments to illustrate the efficacy of our data structure. For comparison, all of our experiments were repeated with (vintage) nonsymmetric YSMP codes using compressed storage. In subsequent tables, total storage requirements are separated into integers and reals. Computation times are separated into: (1) "symbolic factorization"—the time required for a minimum-degree ordering, reordering of the sparse matrix, and computing the symbolic factorization; (2) "numeric factorization"—the time to compute the numerical factorization; and (3) "backsolve"—the time to perform a forward and back substitution. For each grid, the upper row are our results, labeled BLSMP, while the lower row pertains to YSMP. All experiments were performed on a Cray-1A using the CFT 1.11 compiler.

A simple Poisson equation on an $m \times m$ grid with centered finite differences was chosen as a test problem. This leads to a symmetric positive-definite matrix of order $n = m^2$ with an average of 5 nonzeros per row, although the symmetry was ignored. Table I shows the effect of indirect addressing on overall computation time and the effect of the data structure on storage. Furthermore, our data structure requires half the integer storage of YSMP, even with compressed storage. The real storage requirements of the two methods is approximately the same, with minor variations due to the minimum-degree algorithms.

TABLE I
SINGLE NONSYMMERTRIC 5-POINT LAPLACE EQUATION (SCALAR)

| | | | Storage | | CPU time (seconds) | | |
|---|---|---|---|---|---|---|---|
| Grid | Unknowns | Method | Integer | Real | Symbolic Factorization | Numeric Factorization | Backsolve |
| 10 × 10 | 100 | BLSMP | 572 | 1215 | .0122 | .00592 | .00136 |
| | | YSMP | 1168 | 1214 | .0176 | .00615 | .00119 |
| 20 × 20 | 400 | BLSMP | 2643 | 7427 | .0601 | .0431 | .00685 |
| | | YSMP | 5352 | 7248 | .0841 | .0497 | .00626 |
| 30 × 30 | 900 | BLSMP | 6266 | 20277 | .146 | .139 | .01761 |
| | | YSMP | 12594 | 19948 | .203 | .170 | .0164 |
| 40 × 40 | 1600 | BLSMP | 11461 | 40695 | .270 | .317 | .0343 |
| | | YSMP | 23004 | 40804 | .377 | .414 | .0326 |
| 50 × 50 | 2500 | BLSMP | 18281 | 71805 | .441 | .650 | .0591 |
| | | YSMP | 36636 | 69944 | .608 | .806 | .0547 |
| 60 × 60 | 3600 | BLSMP | 26788 | 114333 | .653 | 1.18 | .0923 |
| | | YSMP | 53608 | 113234 | .906 | 1.56 | .0869 |
| 70 × 70 | 4900 | BLSMP | 36846 | 167369 | .912 | 1.95 | .133 |
| | | YSMP | 73924 | 168088 | 1.27 | 2.68 | .127 |
| 80 × 80 | 6400 | BLSMP | 48581 | 237541 | 1.22 | 3.19 | .187 |
| | | YSMP | 97474 | 243552 | 1.71 | 4.68 | .181 |
| 90 × 90 | 8100 | BLSMP | 61863 | 312609 | 1.57 | 4.40 | .245 |
| | | YSMP | 123892 | 312264 | 2.18 | 6.06 | .232 |
| 100 × 100 | 10000 | BLSMP | 76670 | 401257 | 1.95 | 6.07 | .312 |
| | | YSMP | 153752 | 406546 | 2.73 | 8.64 | .300 |

For both codes, the time for computing the ordering is about the same. The timing differences for the numerical factorization reflect the increased number of floating-point operations per indirect address in our implementation. Assuming that this time difference is due only to elimination of indirect addresses, their cost may be estimated. For the largest computed grid ($n = 10^4$), roughly 42 percent of our code was spent computing indirect addressing while indirect addressing accounted for almost 55 percent of the total computing time for YSMP. Clearly, a substantial fraction of the total time goes into non-floating-point work. The backsolve times are about the same for both codes. Since the backsolve must be done sequentially, our data structure does not reduce the amount of indirect addressing required. For this experiment the Cray-1 was run in scalar mode (vectorization disabled) and, thus, timing results should be (qualitatively) applicable to other machines.

We repeated our Poisson experiments with a vectorized version of the code that was written for our semiconductor device simulator [12] and makes use of several sparse BLAS [36], which are generally two to three times faster than scalar BLAS. The results are summarized in Table II.

The storage requirements for both codes are obviously the same as before. In comparing the YSMP times for scalar mode and vector mode, it is clear that the CFT compiler was unable to do much vectorization, which is to be expected. Neither symbolic factorization routine benefits much from vectorization. Our numerical factorization code benefits from reduction in indirect addressing and vectorization. It is important to note that some of the potential speedup is offset by the nonnumerical overhead required to initiate the vector operations. This overhead was estimated by running our code in scalar mode and was roughly

TABLE II
SINGLE NONSYMMETRIC 5-POINT LAPLACE EQUATION (VECTORIZED)

| | | | Storage | | CPU time (seconds) | | |
|---|---|---|---|---|---|---|---|
| Grid | Unknowns | Method | Integer | Real | Symbolic Factorization | Numeric Factorization | Backsolve |
| 10 × 10 | 100 | BLSMP | 572 | 1215 | .0134 | .00625 | .000959 |
| | | YSMP | 1168 | 1214 | .0175 | .00602 | .00117 |
| 20 × 20 | 400 | BLSMP | 2643 | 7427 | .0638 | .0413 | .00390 |
| | | YSMP | 5352 | 7248 | .0838 | .0484 | .00617 |
| 30 × 30 | 900 | BLSMP | 6266 | 20277 | .152 | .122 | .00918 |
| | | YSMP | 12594 | 19948 | .202 | .165 | .0161 |
| 40 × 40 | 1600 | BLSMP | 11461 | 40695 | .277 | .262 | .0169 |
| | | YSMP | 23004 | 40804 | .375 | .402 | .0320 |
| 50 × 50 | 2500 | BLSMP | 18281 | 71805 | .446 | .500 | .0276 |
| | | YSMP | 36636 | 69944 | .606 | .780 | .0538 |
| 60 × 60 | 3600 | BLSMP | 26788 | 114333 | .651 | .859 | .0413 |
| | | YSMP | 53608 | 113234 | .903 | 1.51 | .0856 |
| 70 × 70 | 4900 | BLSMP | 36846 | 167369 | .899 | 1.35 | .0579 |
| | | YSMP | 73924 | 168088 | 1.26 | 2.58 | .125 |
| 80 × 80 | 6400 | BLSMP | 48581 | 237541 | 1.19 | 2.08 | .0785 |
| | | YSMP | 97474 | 243552 | 1.70 | 4.51 | .178 |
| 90 × 90 | 8100 | BLSMP | 61863 | 312609 | 1.52 | 2.83 | .101 |
| | | YSMP | 123892 | 312264 | 2.17 | 5.84 | .228 |
| 100 × 100 | 10000 | BLSMP | 76670 | 401257 | 1.88 | 3.80 | .127 |
| | | YSMP | 153752 | 406546 | 2.72 | 8.32 | .296 |

TABLE III
THREE COUPLED NONSYMMETRIC 5-POINT EQUATIONS (VECTORIZED)

| | | | Storage | | CPU time (seconds) | | |
|---|---|---|---|---|---|---|---|
| Grid | Unknowns | Method | Integer | Real | Symbolic Factorization | Numeric Factorization | Backsolve |
| 5 × 5 | 75 | BLSMP | 107 | 1620 | .00256 | .00722 | .00134 |
| | | YSMP | 758 | 1611 | .0200 | .0103 | .00133 |
| 10 × 10 | 300 | BLSMP | 572 | 10935 | .013 | .0528 | .00654 |
| | | YSMP | 3796 | 10926 | .103 | .100 | .00812 |
| 15 × 15 | 675 | BLSMP | 1433 | 31248 | .0350 | .167 | .0165 |
| | | YSMP | 9366 | 30807 | .273 | .363 | .0222 |
| 20 × 20 | 1200 | BLSMP | 2643 | 66843 | .0636 | .398 | .0326 |
| | | YSMP | 17256 | 65232 | .507 | .956 | .0461 |
| 25 × 25 | 1875 | BLSMP | 4228 | 11348 | .103 | .699 | .0529 |
| | | YSMP | 27350 | 11969 | .835 | 1.87 | .0783 |
| 30 × 30 | 2700 | BLSMP | 6266 | 182493 | .152 | 1.27 | .0825 |
| | | YSMP | 40554 | 179532 | 1.24 | 3.51 | .124 |
| 35 × 35 | 3675 | BLSMP | 8691 | 272178 | .211 | 2.12 | .119 |
| | | YSMP | 56092 | 272151 | 1.76 | 6.42 | .187 |
| 40 × 40 | 4800 | BLSMP | 11461 | 366255 | .277 | 2.88 | .158 |
| | | YSMP | 73842 | 367236 | 2.32 | 8.95 | .252 |

equal to the overhead reduction of the previous example. Thus for the smaller grids, the factorization times for both codes are equal. The benefits of vectorization are realized for the large problems where our code ran two times faster than YSMP. The situation is similar for the backsolve procedure.

In a final experiment, the semiconductor equations were discretized on an $n \times n$ grid. This results in a block $n \times n$ matrix in which each block is $3 \times 3$. In particular, the diagonal and off-diagonal blocks have the form

$$\boldsymbol{a}_{ii} = \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \qquad \boldsymbol{a}_{ij} = \begin{bmatrix} x & 0 & 0 \\ x & x & 0 \\ x & 0 & x \end{bmatrix}. \tag{55}$$

The block zero/nonzero structure is the same as in the preceding examples. The results from our code and the nonsymmetric YSMP are summarized in Table III; both codes were run in vector mode.

The integer storage for our code is much smaller than for YSMP due to blocking. For sufficiently large problems, we would expect the ratio to approach 1/18. Since the compressed storage scheme employed by YSMP is more effective for the block system, this storage reduction is not fully realized in the present example. The differences in real storage reflect differences in ordering and again are not significant.

The time for ordering and symbolic factorization now is significantly smaller for the block method. This primarily reflects the fact that our method orders blocks and takes advantage of the symmetric nonzero pattern, and hence has only about 1/18 of the computations to perform. Since the construction of $\boldsymbol{JU}$ is performed only once in many applications, this savings in computation time will have little consequence. On the other hand, the total computational time is often dominated by the numerical factorization. For the intermediate to large problems, the vectorized block method is two to three times faster than YSMP. This reduction in computation time is primarily due to two effects. First, the factorization routines in our code require 1/18 of the indirect addressing required by YSMP making the block method more numerically intensive. For the smaller problems, the effect of this overhead reduction is probably the dominant factor. Secondly, vectorization will play a larger role as the size of the problem increases. As can be seen by comparing Tables II and III, the benefits of vectorization are enhanced by utilizing the block structure of the matrix during factorization. Similar comments apply to the backsolve phase.

Let us close this subsection with a discussion of some specific iterative methods that are applicable to the semiconductor equations. We have found the Orthomin conjugate-gradient iteration [35], [37] with a previously factored Jacobian as the preconditioner to be effective, particularly in the Newton–Richardson context. Alternatively, an incomplete factorization can be performed, by restricting the matrix factors to have the same structure as the original matrix (allowing no fill-in), to provide a preconditioner [35], [37]; generalized to block matrices as in the coupled system, the algorithm allows some fill-in to occur between the variables associated with each grid point, but restricts the knot coupling to nearest neighbors.

## IV. EXAMPLES

In this section, we shall illustrate the performance of the above algorithms and techniques. The device examples were run with a relative accuracy request of $\epsilon_R = 10^{-3}$ and the circuit problems with $\epsilon_R = 10^{-2}$.

### 4.1. MOSFET Switching

The first example to be considered is a transient simulation of the switching behavior of a MOS inverter structure with resistive and capacitive load elements. Although the transient behavior of MOSFET's has attracted considerable attention in the past, essentially all published papers rely on simplified analytical models [38]–[42]. Only recently, numerical solutions of two-dimensional MOSFET structures have been reported [43]. In our work, we

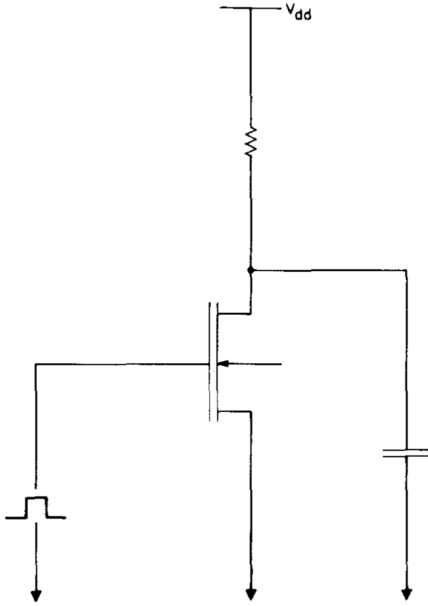



Fig. 3. Schematic of the loaded n-channel MOSFET.

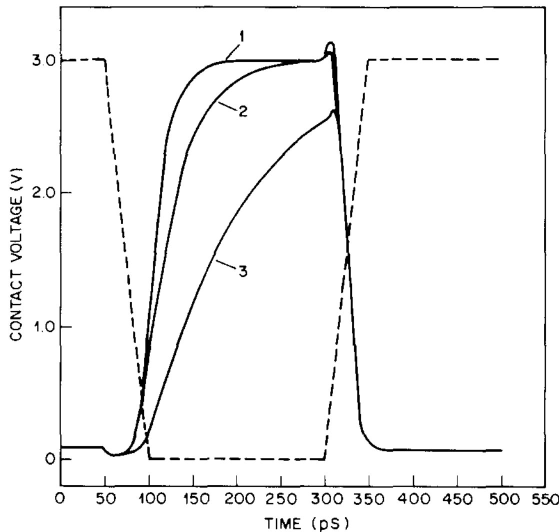



Fig. 4. Input and output node voltages for the loaded MOSFET.

have extended the two-dimensional numerical solution to include external parasitic elements that arise from interconnection lines, pad and/or I/O structures, etc. The idea of coupling external circuit elements to device structures is a powerful technique to study device behavior in the circuit environment.

Fig. 3 gives a schematic representation of the circuit being simulated. The driver device is a scaled n-channel MOSFET optimized for a gate of length 0.5 $\mu$m [2]. We have simulated this rather simple circuit under three different loading conditions, each with a load resistance of 200 Ω. The ideal case with a vanishing output capacitance corresponds to an "unloaded" inverter stage. The two other cases ($C$ = 10 and 50 pF) are typical for an inverter stage with fanouts of one and five.

Fig. 4 shows the simulated node-voltage waveforms at the input and output nodes for the three cases. At $t$ = 50

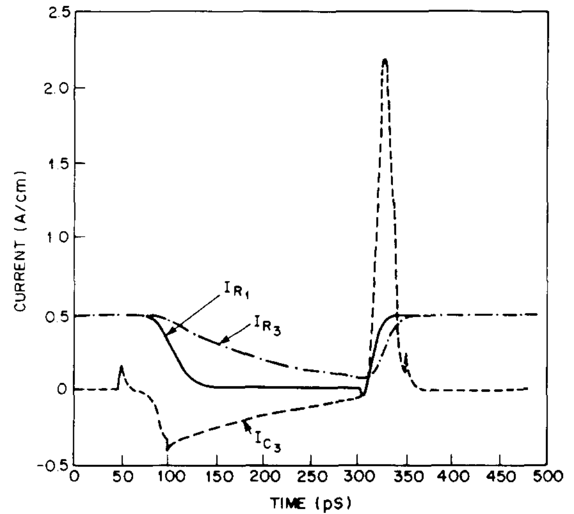



Fig. 5. Currents for the loaded MOSFET.

ps, the gate of the transistor is switched off and on with a 3-V pulse (see Fig. 4, dashed line). For case three, the high loading capacitance prevents the stage from completely turning off. The corresponding currents for cases one and three are compared in Fig. 5. For the purely resistive case, the current $I_{R_1}$ through the load resistor follows the voltage excitation in a nearly ideal fashion. For the capacitively loaded case, the curves labeled $I_{R_3}$ and $I_{C_3}$ show the behavior of the currents through the external elements. Note that a large part of the total current is needed to charge and discharge the load capacitance. Furthermore, it is evident that in case three the driver does turn off completely during the low part of the gate pulse. In a circuit environment, this situation could lead to a loss of logic levels.

### 4.2 CMOS Latchup

The second example concerns the time-dependent behavior of a parasitic CMOS structure under triggering and latchup conditions.

Fig. 6 shows the cross section of part of a CMOS inverter. The $p^+$- region at the left is the p-tub contact, and the $n^+$-contact could be the source of an n-channel transistor. The $p^+$-region at the right of the structure and the substrate are at 3 V. This substrate consists of a highly doped n-substrate and a n-epi layer. The left and middle contact are initially grounded. The n-cathode, p-tub, and n-substrate form a parasitic, vertical, bipolar n-p-n transistor. The p-anode, n-substrate, and the p-tub form a parasitic, lateral, bipolar n-p-n device. The two coupled bipolar devices form a p-n-p-n-thyristor structure with the $n^+$-cathode and the $p^+$-anode at the right. Under ordinary bias conditions, both bipolar devices are "off," and the thyristor is turned off. In our example, we are triggering latchup via an external voltage pulse.

Fig. 7(a)–(f) illustrates the behavior of the electrostatic potential and the electron and hole densities as a function of time. (The colors are carefully chosen from the spectrum with dark blue representing small values and red

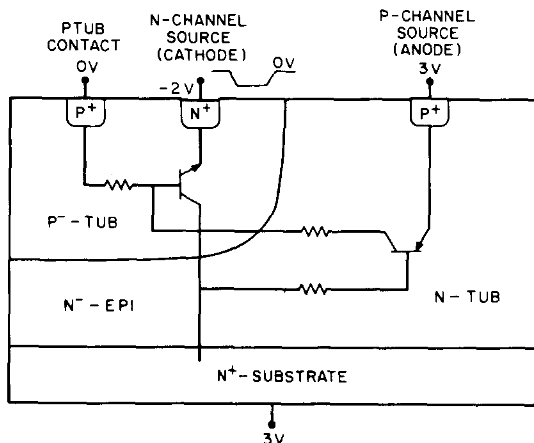



Fig. 6. Cross section of a CMOS inverter, used for latchup simulation.

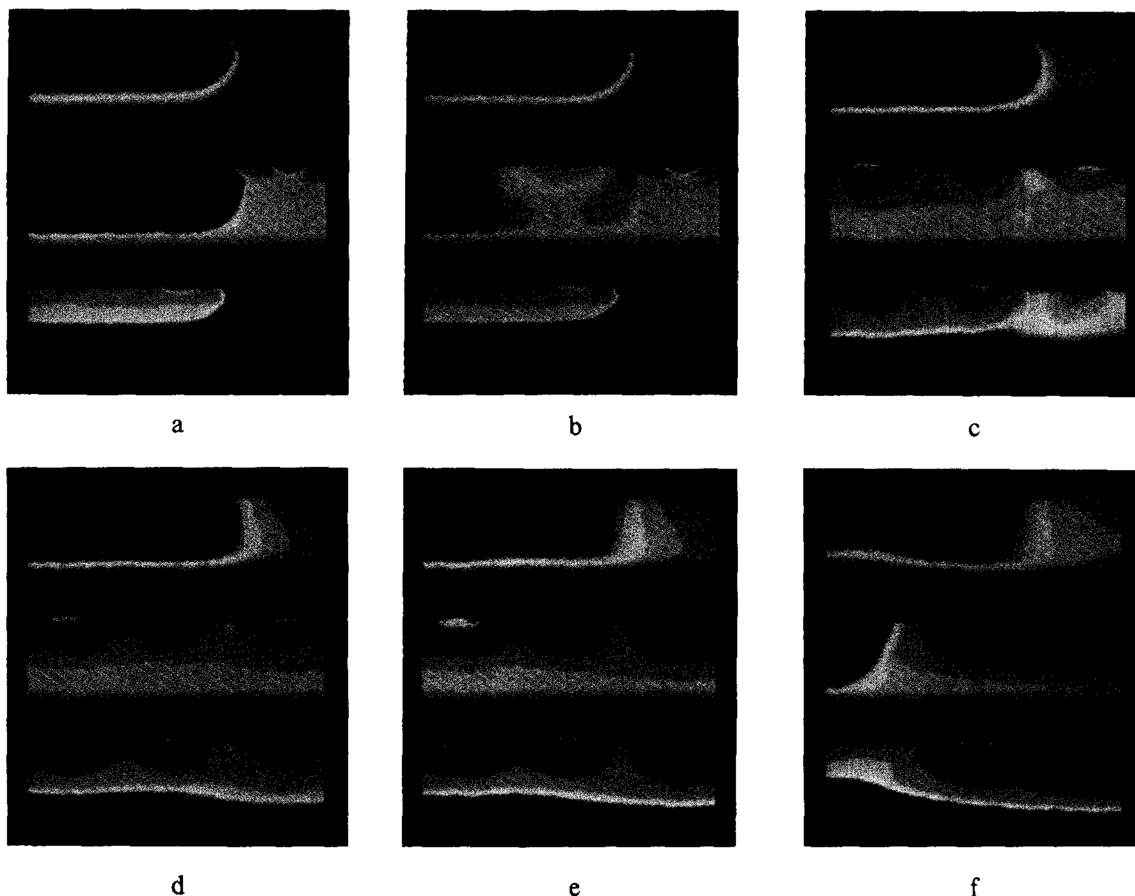



Fig. 7. Electrostatic potential and electron and hole densities as a function of time in the inverter cross section.

representing large values [44]. Unfortunately, the reproduction processes used to create Fig. 7 lost some of the effect of the selected colors.) At $t$ = 0, a 0.5-ns ramp to $-2$ V is applied to the emitter of the vertical n-p-n device. This ramp turns on the vertical n-p-n transistor and electrons flow in vertical direction to the collector substrate. At $t$ = 410 ps, the vertical n-p-n device is fully turned on with the majority of the electrons flowing to the substrate. Some of these electrons "spill over" into the n-epi beneath the $p^+$-anode, making the internal voltage less positive until the anode begins to inject holes. This can be clearly seen in Fig. 7(b). These holes travel through the n-epi, which acts as base of the lateral p-n-p device. Most of the holes enter the space-charge region between the p-tub and the n-substrate (that is, the p-n-p's base-collector junction), turning on the p-n-p. At $t$ = 12.5 ns, the lateral pnp device is completely turned on (Fig. (7c)). The injection of holes into the p-tub forward biases the p-tub cathode junction even more, sending more electrons into the epi, thus getting more holes, and so forth (see Fig. (7c)).

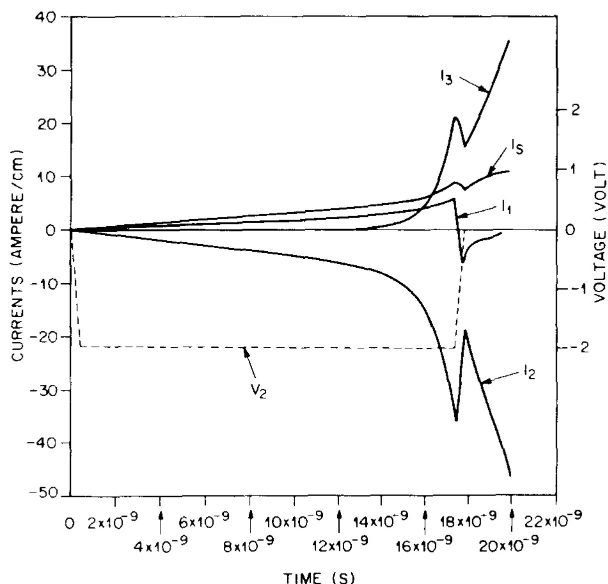



Fig. 8. Contact currents as a function of time in the inverter cross section.

TABLE IV
NUMERICAL PERFORMANCE FOR THE CMOS LATCHUP EXAMPLE

| | Full Newton | Newton-Richardson |
|---|---|---|
| Number of time steps | 226 | 243 |
| Minimum $h_n$ | $2.778 \times 10^{-12}$ | $1.642 \times 10^{-12}$ |
| Average $h_n$ | $8.850 \times 10^{-11}$ | $8.230 \times 10^{-11}$ |
| Maximum $h_n$ | $4.567 \times 10^{-10}$ | $4.518 \times 10^{-10}$ |
| Number of restarts | 2 | 6 |
| Number of Newton Iterations | 485 | 626 |
| Number of Newton Iterations/time step | 2.146 | 2.576 |
| Total CPU time (relative units) | 2.11 | 1.0 |
| CPU time/time step | 2.27 | 1.0 |
| CPU time/Newton | 2.72 | 1.0 |
| Solve time/ assembly time | 6.79 | 2.05 |

Beginning at $t = 17.5$ ns, the initial trigger pulse is turned off in 0.5 ns. Fig. 7(e) shows the internal situation at $t = 18$ ns after the trigger pulse has been turned off. While this process reduces the number of electrons injected into the p-tub, there are too many mobile carriers in the p-tub and the epi to stop the injection process totally. Although we have returned to the initial bias conditions, the structure latches due to internal forward-biasing of the junctions. The final potential and carrier distributions are shown in Fig. 7(f).

Fig. 8 shows the time dependence of the contact currents. Once the structure begins to latch, the anode and cathode current increase sharply. This increase results in a strong coupling between the n-p-n and p-n-p devices, corresponding to high injection conditions in the structure.

From the numerical point of view, transient latchup is a stringent test for both the time-stepping algorithm and the nonlinear and linear iterations. Table IV gives a summary for the simulation and Fig. 9 shows the distribution of time steps for the calculation. The size of the steps closely tracks the behavior of the current; in other words, the steps become small when the current changes rapidly.

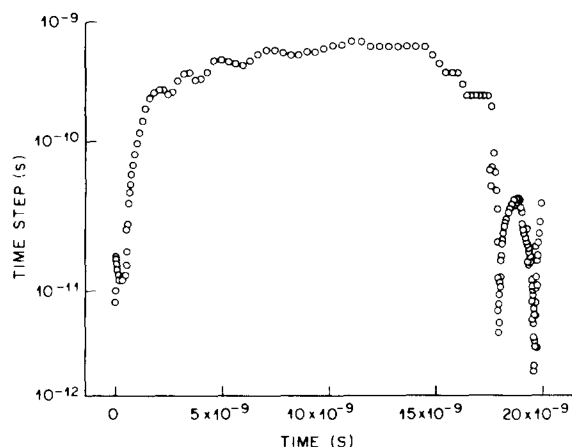



Fig. 9. Distribution of stepsizes for the latchup example.

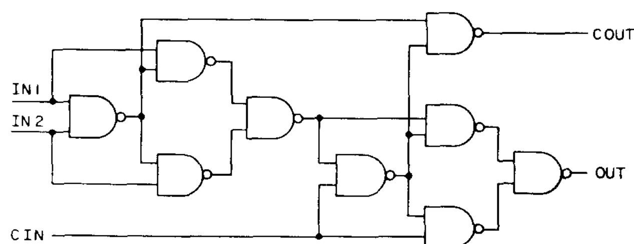



Fig. 10. One-bit full adder circuit.

The power of the Newton–Richardson (NR) iteration, accelerated by Orthomin, becomes evident from the results in Table IV. The overall computation time is determined by two factors: the efforts expended in the outer Newton loop and in the inner iteration (see (51)) to "solve" the linear system for the change. For the full Newton simulation where the Jacobian is factored each time, over 85 percent of the computation time is spent in solving linear systems. The effect of the NR iteration is to trade the inner linear iterations for outer Newton iterations. That is, the linear systems are solved 3.5 times faster but to a lesser degree of accuracy. Thus a 30-percent increase in the number of Newton iterations is required to obtain the same accuracy. However, when combined with a slight increase in the number of time steps, the NR iteration reduces the overall computation time by more than a factor of two.

### *4.3. Ripple-Carry Adder*

The final example we consider is a 64-bit ripple-carry adder circuit made up of full-bit adders composed of NAND's (see Fig. 10). The outputs are lightly loaded. A number of short pulses are input. For this problem, there are 1152 unknowns and the underlying nodal equations are a permutation of a small-bandwidth matrix. We experimented with two Newton strategies for solving the nonlinear equations at each time substep: (1) doing at most 5 full (factoring the Jacobian every time) Newtons and (2) doing a full Newton, followed by a Newton–Richardson step with

TABLE V
SUMMARY OF TRANSIENT SIMULATIONS OF A 64-BIT RIPPLE-CARRY ADDER

| Newton Strategy | Time Steps | Jacobian Factorizations | Jacobian Backsolves | Error Failures | Newton Failures |
|---|---|---|---|---|---|
| Newton(5) | 123 | 542 | 542 | 4 | 0 |
| N-NR-N-NR(2) | 128 | 347 | 685 | 0 | 8 |

at most 5 inner iterations, followed by a full Newton, and finally 2 Newton–Richardson steps. Table V summarizes our results.

The Newton/Newton–Richardson (N/NR) procedure saves 36 percent of the factorizations used by the Newton approach, at the cost of doing 26 percent more backsolves and a few additional function evaluations. The cost of the function evaluations is such that both the N/NR and Newton strategies use roughly the same amount of time. In other words, the cost of forming $g$ and $g'x$ and doing a backsolve is not much cheaper than doing an additional factorization. However, in more complex circuits with more interconnections, the N/NR should prove more efficient as in the semiconductor case. This points out the importance of an implementation to dynamically determine the relative costs of function evaluations, multiplying the Jacobian times a vector, factoring the Jacobian, and backsolving in order to optimize the nonlinear algorithm.

## V. CONCLUSIONS

In this paper, we have discussed some of the relevant physics that lead to the usual semiconductor equations. Appropriate boundary conditions and the addition of external circuit elements have been considered. The circuit equations also fit into the same general class of differential-algebraic systems.

We introduced an easily implemented one-step method, composed of the trapezoidal rule and the second-order backward-differentiation formula, for integrating time-dependent equations. (Others have advocated the use of backward-differentiation formulae for differential-algebraic systems [24], [45], [23], [15] while some others have suggested cyclic multistep methods for differential equations [46], [47].) Second-order methods appear to be nearly optimal for the device and circuit simulations we have performed, which agrees with the previous experience for circuit analysis [48], [49]. The use of Newton-iterative methods for improving the performance of the overall time-integration has also been explored.

We demonstrated the effectiveness of a data structure for sparse nonsymmetric matrices with symmetric nonzero structures, which arise from PDE's and circuit analyses. The matrices are stored in diagonal, upper triangular, and lower triangular components with integer pointers to only the upper triangular segments, which makes iterative and direct methods simple to implement. In addition to saving storage, indirect addressing is reduced since each reference to a column or row index corresponds to two floating-point operations rather than one. For sparse factorizations and backsolves on a CRAY-1, non-floating-point computations represent roughly half of the total computation time so vectorization can produce a savings of a factor of two, at most. Further improvements in performance must be accompanied by a reduction in overhead, which can be obtained with block methods for systems of PDE's. Numerical experiments indicate that a factor of three reduction in computation time can be obtained for the semiconductor equations.

The simulations we presented in the previous section, including a CMOS latchup example, show the efficacy of our techniques.

## REFERENCES


[1] W. Fichtner, E. N. Fuls, R. L. Johnston, R. K. Watts, and W. W. Weick, "Optimized MOSFETs for subquartermicron channel lengths," in *IEDM Tech. Dig.*, pp. 384–387, 1983.

[2] W. Fichtner, "Physics and simulation of small MOS devices," in *IEDM Tech. Dig.*, pp. 638–641, 1982.

[3] H. L. Grubin, D. K. Ferry, G. J. Iafrate, and J. R. Barker, "The numerical physics of micron-length and submicron-length semiconductor devices," *VLSI Electronics — Microstructure Science 3*, pp. 197–300, 1982.

[4] K. Yokoyama, M. Tomizawa, A. Yoshii, and T. Sudoh, "Semiconductor device simulation at NTT," this issue, pp. 452–461.

[5] K. Blotekjaer, "Transport equations for electrons in two-valley semiconductors," *IEEE Trans. Electron Devices*, vol. ED-17, pp. 38–47, 1970.

[6] J. R. Barker and D. K. Ferry, "On the physics and modeling of small semiconductor devices I," *Solid-State Electron.*, vol. 23, pp. 519–530, 1980.

[7] W. Shockley, *Electrons and Holes in Semiconductors.* Princeton, NJ: van Nostrand, 1950.

[8] R. P. Mertens, R. J. Van Overstraeten, and H. J. De Man, "Heavy doping effects in silicon," *Adv. Electron. Elec. Phys.*, vol. 55, pp. 77–118, 1981.

[9] R. A. Smith, *Semiconductors.* Cambridge, UK: Cambridge Univ. Press, 1978.

[10] A. De Mari, "An accurate numerical one-dimensional solution of the pn-junction under arbitrary transient conditions," *Solid-State Electron.*, vol. 11, pp. 1021–1053, 1968.

[11] R. E. Bank, D. J. Rose, and W. Fichtner, "Numerical methods for semiconductor device simulation," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1031–1041, 1983.

[12] R. E. Bank, W. M. Coughran, Jr., W. Fichtner, D. J. Rose, and R. K. Smith, "Computational aspects of transient device simulation," in *Process and Device Simulation*, W. L. Engl, Ed. New York: North-Holland, to be published.

[13] L. R. Petzold, "Differential/algebraic equations are not ODE's," *SIAM J. Scientific Statistical Computing*, vol. 3, pp. 367–384, 1982.

[14] W. C. Rheinboldt, "Differential-algebraic systems as differential-equations on manifolds," Univ. of Pittsburgh Inst. for Computational Mathematics and Applications Rep. ICMA-83-55, 1983.

[15] C. W. Gear and L. R. Petzold, "ODE methods for the solution of differential/algebraic Systems," *SIAM J. Numer. Anal.*, vol. 21, pp. 716–728, 1984.

[16] L. O. Chua and P.-M. Lin, *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques.* Englewood Cliffs, NJ: Prentice-Hall, 1975.

[17] G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The sparse tableau approach to network analysis and design," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 101–113, 1971.

[18] W. M. Coughran, Jr., E. H. Grosse, and D. J. Rose, "CAzM: A circuit analyzer with macromodeling," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1207–1213, 1983.

[19] S. Sastry, C. Desoer, and P. Varaiya, "Jump behavior of circuits and systems," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 451–454, 1981.

[20] G. D. Hachtel and A. L. Sangiovanni-Vincentelli, "A survey of third-generation simulation techniques," *Proc. IEEE*, vol. 69, pp. 1264–1280, 1981.

[21] W. Fichtner, D. J. Rose, and R. E. Bank, "Semiconductor device sim-

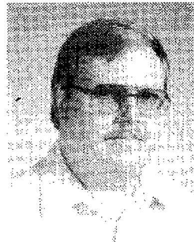
ulation," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1018-1030, 1983.
[22] J. Lambert, *Computational Methods in Ordinary Differential Equations*. London: Wiley, 1973.
[23] L. R. Petzold, "A description of DASSL: A Differential/Algebraic System Solver," in *Scientific Computing. New York:* North-Holland, 1983, pp. 65-68; also presented at the 10th IMACS World Congress on Systems Simulation and Scientific Computation, Montreal, Canada, Aug. 8-13, 1982.
[24] C. W. Gear, "The simultaneous numerical solution of differential-algebraic systems," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 89-95, 1971.
[25] —, "Efficient stepsize control for output and discontinuities," Univ. of Illinois Computer Science Dep. Rep. UIUCDCS-R-82-1111, 1982.
[26] R. E. Bank and D. J. Rose, "Global approximate Newton methods," *Numer. Math.*, vol. 37, pp. 279-295, 1981.
[27] W. M. Coughran, Jr., E. H. Grosse, and D. J. Rose, "Variation diminishing splines in simulation," *SIAM J. Scientific and Statistical Computing,* to be published.
[28] R. E. Bank, "PLTMG user's guide-Edition IV," Univ. of California, San Diego, Mathematics Dep. Rep., 1985.
[29] S. C. Eisenstat, M. H. Schultz, and A. H. Sherman, "Considerations in the design of software for sparse gaussian elmination," in *Sparse Matrix Computations*, J. R. Bunch and D. J. Rose, Eds. New York: Academic, 1976, pp. 263-274.
[30] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman, "Yale sparse matrix package I: The symmetric codes," *Int. J. Numer. Methods in Eng.*, vol. 18, pp. 1145-1151, 1982.
[31] —, "Yale sparse matrix package II: The nonsymmetric codes," Yale Univ. Computer Science Dep. Research Rep. 114, 1977.
[32] R. S. Varga, *Matrix Iterative Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1962.
[33] A. George and J. W.-H. Liu, *Computer Solutions of Large Sparse Positive Definite Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
[34] G. W. Stewart, *Introduction to Matrix Computations*. New York: Academic, 1973.
[35] L. A. Hageman and D. M. Young, *Applied Iterative Methods*. New York: Academic, 1981.
[36] W. P. Petersen, "Basic linear algebra subprograms for CFT usage," Cray Res. Tech. Note 2240208, 1979.
[37] H. C. Ellman, "Iterative methods for large, sparse, nonsymmetric systems of linear equations," Yale Univ. Computer Science Dep. Res. Rep. 229, 1982.
[38] T. J. O'Reilly, "The transient response of insulated-gate field-effect transistors," *Solid-State Electron.*, vol. 8, pp. 947-956, 1965.
[39] J. R. Burns, "High-frequency characteristics of the insulated-gate field-effect transistor," *RCA Rev.*, 28, pp. 385-418, 1967.
[40] —, "Large-signal transit-time effects in the MOS transistor," *RCA Rev.*, vol. 30, pp. 15-35, 1969.
[41] K. Goser, "Channel formation in an insulated-gate field-effect transistor (IGFET) and its equivalent circuit," *Siemens Res. Dev.*, vol. 1, pp. 3-9, 1971.
[42] M. E. Zahn, "Calculation of the turn-on behavior of MOST," *Solid State Electron.*, vol. 17, pp. 843-854, 1974.
[43] S. Laux, "Techniques for small-signal analysis of semiconductor diodes," this issue, pp. 472-481.
[44] E. H. Grosse, "Automatic choice of colors for level plots," AT&T Bell Labs. Numerical Analysis Manuscript 85-1, 1985.
[45] R. K. Brayton, F. G. Gustavson, and G. D. Hachtel, "A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas," *Proc. IEEE*, vol. 60, pp. 98-108, 1972.
[46] P. Tischer and R. Sacks-Davis, "A new class of cyclic multistep formulae for stiff systems," *SIAM J. Scientific and Statistical Computing*, vol. 4, pp. 733-747, 1983.
[47] J. M. Tendler, T. A. Bickart, and Z. Picel, "A stiffly stable integration process using cyclic composite methods," *ACM Trans. Math. Software*, vol. 4, pp. 339-368, 1978.
[48] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott, "Algorithms for ASTAP—A network-analysis program," *IEEE Trans. Circuit theory*, vol. CT-20, pp. 628-634, 1973.
[49] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Univ. California, Berkeley, Electronics Res. Lab. Memo ERL-M520, 1975.

**Randolph E. Bank** was born in Pittsburgh, PA, on September 13, 1949. He received the B.S. degree in chemistry from the University of Denver, Denver, CO, in 1971 and the M.A. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA, in 1972 and 1975, respectively.

From 1975 to 1981 he held academic positions at the University of Chicago, Chicago, and the University of Texas, Austin, and a visiting position at Yale University, New Haven, CT. In 1981, he became a member of the mathematics faculty at the University of California, San Diego. His main research interest is the study of numerical algorithms for solving partial differential equations.

Dr. Bank is a member of SIAM.

*



**William M. Coughran, Jr.** was born in Fresno, CA, on February 13, 1953. He received the B.S. and M.S. degrees in mathematics from the California Institute of Technology, Pasadena, in 1975, and the M.S. and Ph.D. degrees in computer science from Stanford University, Palo Alto, CA, in 1977 and 1980, respectively.

In 1980, he became a member of technical staff at AT&T Bell Laboratories where he researches scientific computing problems and numerical analysis.

Dr. Coughran is a member of SIAM.

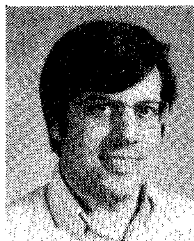
*

**Wolfgang Fichtner** (M'79-SM'84) received the Dipl.Ing. degree in physics and the Ph.D. degree in electrical engineering from the Technical University of Vienna, Vienna, Austria, in 1974 and 1978, respectively.

From 1975 to 1978, he was an Assistant Professor in the Department of Electrical Engineering, Technical University of Vienna, working on noise in semiconductor devices and device modeling and characterization. In 1979, he joined Bell Laboratories, Murray Hill, NJ, where in 1982, he became Supervisor of a group concerned with all aspects of computer modeling and VLSI device testing.

Dr. Fichtner is a member of the Electrochemical Society, the American Institute of Physics, and the Society of Industrial and Applied Mathematics.

*



**Eric H. Grosse** was born in Detroit, MI, on February 24, 1953. He received the B.S. degree in mathematics from the Michigan State University, East Lansing, in 1975 and the Ph.D. degree in computer science from Stanford University, Palo Alto, CA, in 1980.

In 1980, he became a member of technical staff at AT&T Bell Laboratories where he works on various topics in scientific computing, particularly multivariate approximation.

Dr. Grosse is a member of SIAM.

**Donald J. Rose** was born in Santa Ana, CA, on May 25, 1944. He received the B.A. degree in mathematics from the University of California at Berkeley in 1966, and the M.A. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA, in 1967 and 1970, respectively.

From 1970 to 1984, Dr. Rose held research positions at the University of Denver, Harvard University, Vanderbilt University, and AT&T Bell Labortories. In 1984, he became the Chairman of the Computer Science Department at Duke University, Durham, NC, where he investigates problems involving numerical analysis and scientific computation.

Dr. Rose is a member of SIAM, AMS, MAA, and ACM.

**R. Kent Smith** received the B.S. degree in electrical engineering and the M.S. degree in physics from Bucknell University, Lewisburg, PA, in 1965 and 1967, respectively, and the Ph.D. degree in physics from the University of Maryland, College Park, in 1972.

From 1972 to 1975 he was a Alexander von Humboldt fellow at the Institut fuer Physik in Frankfurt, West Germany, and from 1975 to 1979 he was an Assistant Professor of physics at Duke University, NC. In 1979 he joined AT&T Bell Labs as a member of the technical staff where he is engaged in the numerical simulation of semiconductor devices.