SIAM J. SCI. STAT. COMPUT.
Vol. 13, No. 1, pp. 289–306, January 1992

015

FAST ITERATIVE SOLUTION OF CARRIER CONTINUITY EQUATIONS FOR THREE-DIMENSIONAL DEVICE SIMULATION*

O. HEINREICHSBERGER†, S. SELBERHERR†, M. STIFTINGER†, AND K.P. TRAAR‡

Abstract. In this paper the use of iterative methods for the solution of the carrier continuity equations in three-dimensional semiconductor device simulators is summarized. An overview of the derivation of the linear systems from the basic stationary semiconductor device equations is given and the algebraic properties of the nonsymmetric coefficient matrices are discussed. Results from the following classes of iterative methods are presented: The classical conjugate gradient (CG), the symmetrized conjugate gradient (SCG), the generalized minimum residual (GMRES), and the conjugate gradient squared (CGS) method. Preconditioners of incomplete factorization type with partial fill-in are considered. High performance implementations for these algorithms on vector, concurrent, and vector–concurrent computers are presented.

Key words. semiconductor equations, nonsymmetric systems, preconditioned iterative methods, vector computers

AMS(MOS) subject classifications. 65L10, 65F10, 65F20, 35J65

1. Introduction.

The three-dimensional numerical analysis of semiconductor devices by device simulators is increasingly becoming an indispensable tool in design and optimization of micro-miniaturized devices. Such simulators compute the discrete self-consistent solution of the semiconductor device partial differential equations. We restrict ourselves to the decoupled solution method of the three nonlinear device equations on a three-dimensional nonuniform tensor product grid [15]. In this case each single nonlinear (outer) iteration consists of the solution of the Poisson equation for the electrostatic potential ψ and of two carrier continuity equations for the electron and hole concentrations, respectively. The coefficient matrices of the discrete continuity equations in the most practical variable set n (electrons) and p (holes) are nonsymmetric. In this contribution we consider preconditioned iterative methods for the solution of these nonsymmetric linear systems. Related work is found in [12], [19], [21], [28], and [32].

Iterative methods applied to the discrete continuity equations have to cope with high condition numbers of the coefficient matrices [1]. Another problem is the enormous numerical range of the solution vector that has to be computed accurately both in the depletion zones of the device under consideration as well as in high injection regimes. Contrary to the Poisson equation, the discrete continuity equations have to be evaluated much more accurately to guarantee the stability of the nonlinear iteration. This results in substantially higher iteration counts compared to the Poisson equation, and therefore the linear nonsymmetric solvers dominate in the solution process.

We have performed a comparative study of various preconditioned conjugate gradient type solvers of which the conjugate gradient squared (CGS) method [14], [26] was identified as the fastest and most economical. The success of this method (and related ones) depends quite critically on robust preconditioning. We have concentrated on incomplete factorizations of the coefficient matrix. Partial fill-in substantially reduces the iteration count at the expense of more arithmetic work per iteration.

* Received by the editors April 5, 1990; accepted for publication (in revised form) February 26, 1991. This work was supported by Siemens AG, Munich, and Digital Equipment Corporation, Hudson.

† Institute for Microelectronics, Technical University of Vienna, Vienna, Austria.

‡ SIEMENS AG Austria, ETG 215, Vienna, Austria.

Multiple solutions of linear systems of rank $\mathcal{O}(10^5)$ or even larger on vector or vector–concurrent computers with large main memory requires vectorized and/or parallelized iterative procedures. Our implementation has therefore been concentrated on an efficient vectorizable ILU preconditioner. We show how high performance is achieved on supercomputers such as the CRAY 2 (one vector unit), Fujitsu VP200, and on superminicomputers such as the Alliant FX40 (two vector CEs) and a Digital VAX 6260 (one to six scalar processors).

The outline of this paper is as follows. In §2 the semiconductor partial differential equations and the nonlinear expressions for the physical quantities within these equations are summarized, the discretization of which is discussed in §3. The brief consideration of the algebraic matrix properties in §4 provides the preliminaries for the iterative procedures outlined in §5. Robust preconditioning is vital for the iterative solution of the carrier continuity equations. We consider incomplete factorization preconditioners in §6 and their implementation on vector and parallel computers in §7. Numerical experiments and conclusive remarks are found in §8.

2. The semiconductor partial differential equations. We consider the time-invariant case on a three-dimensional rectangular spatial domain using finite difference discretization. The semiconductor equations for the variables (ψ, n, p) consist of the Poisson equation and the carrier continuity equations. Poisson's equation for the electrostatic potential ψ reads

$$\operatorname{div}(\epsilon \cdot \operatorname{grad}\psi) = -\rho \tag{1}$$

with the space charge $\rho = q \cdot (p - n + C)$, where C denotes the net doping density, n the hole, p the electron concentrations, and q the elementary charge. The carrier continuity equations for the electron and hole current densities $\vec{J}_{n,p}$ read

$$\operatorname{div}\vec{J}_n = q \cdot R, \tag{2}$$

$$\operatorname{div}\vec{J}_p = -q \cdot R, \tag{3}$$

where R denotes the carrier generation and recombination rate.

The current densities $\vec{J}_{n,p}$,

$$\vec{J}_n = \mu_n^{LISF} \cdot n \cdot \vec{F}_n, \tag{4}$$

$$\vec{J}_p = \mu_p^{LISF} \cdot p \cdot \vec{F}_p, \tag{5}$$

are assumed to be proportional to the driving forces $\vec{F}_{n,p}$. An extended drift–diffusion approach allows the treatment of hot electron effects in one-band semiconductors such as silicon. This approach for the driving forces reads [13]

$$\vec{F}_n = -q\left(\operatorname{grad}\psi - \frac{1}{n} \cdot \operatorname{grad}\left(\frac{k \cdot T_n}{q} \cdot n\right)\right), \tag{6}$$

$$\vec{F}_p = -q\left(\operatorname{grad}\psi + \frac{1}{p} \cdot \operatorname{grad}\left(\frac{k \cdot T_p}{q} \cdot p\right)\right), \tag{7}$$

where carrier heating is modeled by carrier temperatures $T_{n,p}$. For the mobilities $\mu_{n,p}^{LISF}$ the effect of carrier heating is modeled by a nonlinear dependence on the magnitude of the driving forces $F_{n,p}$:

$$\mu_{n,p}^{LISF} = \frac{2\mu_{n,p}^{LIS}}{1 + \left(1 + \left(2\mu_{n,p}^{LIS} \cdot |\vec{F}_{n,p}|/(q \cdot v_{n,p}^{sat})\right)^{\alpha_{n,p}}\right)^{1/\alpha_{n,p}}} \tag{8}$$

with $\alpha_n = 2$, $\alpha_p = 1$. $\mu_{n,p}^{LIS}$ denotes the zero-field mobility due to lattice (L), impurity (I), and surface (S) scattering mechanisms and $v_{n,p}^{sat}$ the saturation velocity.

Approximations for the carrier temperatures $T_{n,p}$ can be derived by a series expansion of the energy conservation equations

$$T_{n,p} = T_0 + \frac{2}{3} \cdot \frac{q}{k} \cdot \tau_{n,p}^{\epsilon} \cdot \left(v_{n,p}^{sat}\right)^2 \cdot \left(\frac{1}{\mu_{n,p}^{LISF}} - \frac{1}{\mu_{n,p}^{LIS}}\right) \tag{9}$$

with the Boltzmann constant k, the ambient temperature T_0 and the energy relaxation times $\tau_{n,p}^{\epsilon}$.

The carrier generation and recombination rate R on the right-hand side of the carrier continuity equations represents the sum of the impact ionization rate R^{II}, the Shockley–Read–Hall recombination rate R^{SRH}, and the Auger recombination rate R^{AU}:

$$R = R^{II} + R^{SRH} + R^{AU}. \tag{10}$$

The impact ionization rate is modeled by the Chynoweth formulae

$$R^{II} = -\alpha_n \cdot \frac{\left|\vec{J}_n\right|}{q} - \alpha_p \cdot \frac{\left|\vec{J}_p\right|}{q}, \tag{11}$$

in which the $\alpha_{n,p}$ depend exponentially on the local electric field:

$$\alpha_{n,p} = \hat{\alpha}_{n,p} \cdot \exp\left(\frac{-\beta_{n,p}}{\left(\vec{J}_{n,p}/|\vec{J}_{n,p}|\right) \cdot \mathrm{grad}\psi}\right). \tag{12}$$

The SRH recombination rate is expressed by

$$R^{SRH} = \frac{\left(n \cdot p - n_i^2\right)}{\tau_p \left(n + n_1\right) + \tau_n \left(p + p_1\right)} \tag{13}$$

with positive constants n_1,p_1,$\tau_{n,p}$. Last, the Auger recombination rate is given by

$$R^{AU} = \left(C_n \cdot n + C_p \cdot p\right) \cdot \left(n \cdot p - n_i^2\right) \tag{14}$$

with $C_{n,p} \geq 0$.

3. Discretization of the nonlinear system of equations. The nonlinear system of equations can be solved either by a full Newton iteration or by decoupling the three partial differential equations (Gummel's algorithm [8]). We restrict ourselves to the latter option. In that case, a nonlinear Gauss–Seidel block iterative scheme is obtained neglecting the nonlinearities in the carrier mobilities and temperatures:

$$\mathrm{div}\,\mathrm{grad}\psi^{k+1} = -\frac{q}{\epsilon}\left(\frac{\partial \left(p - n + C\right)^k}{\partial \psi}\left(\psi^{k+1} - \psi^k\right) + p^k - n^k + C\right), \tag{15}$$

$$\mathrm{div}\vec{J}_p\left(\psi^{k+1}, p^{k+1}\right) = -q\left(R\left(\psi^{k+1}, n^k, p^k\right) + \frac{\partial R}{\partial p}\left(p^{k+1} - p^k\right)\right), \tag{16}$$

$$\mathrm{div}\vec{J}_n\left(\psi^{k+1}, n^{k+1}\right) = q\left(R\left(\psi^{k+1}, n^k, p^{k+1}\right) + \frac{\partial R}{\partial n}\left(n^{k+1} - n^k\right)\right). \tag{17}$$

The set of equations is now discretized on a three-dimensional domain. The boundary value problem is of mixed Dirichlet–Neumann type. For the idealized ohmic contacts, Dirichlet boundary conditions hold. For the artificial interfaces in the deep semiconductor bulk, homogenous Neumann boundary conditions have to be applied. Nonhomogenous Neumann boundary conditions for the electrostatic potential are valid in case of interface charges, e.g., at semiconductor–oxide interfaces. Nonvanishing interface recombination velocities at Schottky contacts yield nonhomogenous Neumann boundary conditions for the carrier concentrations.

The nonlinearities in R have to be treated carefully. The derivatives of R^{II} with respect to the carrier concentrations can be neglected if the carrier generation rate is not updated at every nonlinear iteration but in a superimposed generation *supercycle*. For the recombination rates R^{SRH}, however, the derivatives with respect to n or p are computed, because these contributions increase the diagonal dominance in the resulting linear system. Such a stabilizing effect is not necessarily true for R^{AU}, therefore, negative contributions of the derivatives of R^{AU} to the main diagonal of the coefficient matrix are discarded.

For the finite difference discretization of the carrier continuity equations an exponential interpolation scheme for the carrier concentrations n and p must be used [3], [24]. This is due to an exponential dependence of the carrier concentrations on the electrostatic potential (Scharfetter–Gummel interpolation). Herein the quantities ψ, $\mu_{n,p}$, and $T_{n,p}$ are interpolated linearly. For a one-dimensional nonuniform discretization with mesh spacings h_i, neglecting the derivatives of R and assuming constant carrier temperatures $T_{n,p}$, one obtains for the three-point stencil

$$\begin{aligned} &n_{i-1}D_{n,i-1/2}\frac{\mathcal{B}(-\Delta_{i-1})}{2h_{i-1}} + n_{i+1}D_{n,i+1/2}\frac{\mathcal{B}(\Delta_i)}{2h_i} \\ &-n_i\left(D_{n,i-1/2}\frac{\mathcal{B}(\Delta_{i-1})}{2h_{i-1}} + D_{n,i+1/2}\frac{\mathcal{B}(-\Delta_i)}{2h_i}\right) = R_i\frac{h_{i-1}+h_i}{2}, \end{aligned} \tag{18}$$

$$\begin{aligned} &p_{i-1}D_{p,i-1/2}\frac{\mathcal{B}(\Delta_{i-1})}{2h_{i-1}} + p_{i+1}D_{p,i+1/2}\frac{\mathcal{B}(-\Delta_i)}{2h_i} - \\ &p_i\left(D_{p,i-1/2}\frac{\mathcal{B}(-\Delta_{i-1})}{2h_{i-1}} + D_{p,i+1/2}\frac{\mathcal{B}(\Delta_i)}{2h_i}\right) = R_i\frac{h_{i-1}+h_i}{2}. \end{aligned} \tag{19}$$

In these formulae the diffusivities obey the Einstein relation $D_{n,p} = \mu_{n,p} \cdot Ut$, where $Ut = \frac{k \cdot T_0}{q}$ denotes the thermal voltage. $\mathcal{B}$ is the Bernoulli function

$$\mathcal{B}(x) = \frac{x}{\exp(x) - 1}. \tag{20}$$

The arguments of the Bernoulli function are $\Delta_i = \frac{\psi_{i+1} - \psi_i}{Ut}$.

For nonconstant carrier temperatures $T_{n,p}$, the above expressions are generalized as follows [27]. Let

$$Ut_{(n,p),i} = \frac{k \cdot T_{(n,p),i}}{q} \tag{21}$$

denote the so-called local "electronic voltages" at the meshpoint i. Assume further that the electronic voltages $Ut_{(n,p),i}$ vary linearly between the meshpoints as is the assumption for the electrostatic potential. Then a local one-dimensional continuity equation is solved for the current densities $J_{n,p}$ between neighboring meshpoints.

Assuming constant current densities, the following midpoint interpolation formula replaces the diffusivities

$$D_{(n,p),i+1/2} = \mu_{(n,p),i+1/2} \cdot \left(Ut_{(n,p),i+1} - Ut_{(n,p),i}\right) \cdot \left[\ln\left(\frac{Ut_{(n,p),i+1}}{Ut_{(n,p),i}}\right)\right]^{-1}. \tag{22}$$

The midpoint values for the mobilities $\mu_{(n,p),i+1/2}$ are approximated by linear interpolation. The Bernoulli function argument evaluates to

$$\Delta_{(n,p),i} = \frac{(\psi_{i+1} - \psi_i) - \left(Ut_{(n,p),i+1} - Ut_{(n,p),i}\right)}{\left(Ut_{(n,p),i+1} - Ut_{(n,p),i}\right)} \cdot \ln\left(\frac{Ut_{(n,p),i+1}}{Ut_{(n,p),i}}\right). \tag{23}$$

4. Algebraic properties of the coefficient matrices. The exponential interpolation scheme outlined in the last section produces a nonsymmetric, diagonally dominant, two-cyclic, seven-band coefficient matrix A.

Nonsymmetry is caused by the inequality $\mathcal{B}(-x) \neq \mathcal{B}(x)$. Diagonal dominance is due to the fact that each negative column sum of the offdiagonal elements is less than (for nonvanishing R) or equal (for vanishing R) to the main diagonal pivot. This implies at least semidefiniteness of A. Consider the case of constant carrier temperatures $T = T_n = T_p$. The equality

$$\mathcal{B}(-x) = \exp(x) \cdot \mathcal{B}(x) \tag{24}$$

and the fact that the exponentially scaled potential increments can be factored yields that A can be transformed to a symmetric, positive definite matrix $\bar{A}$,

$$\bar{A} = W^{-1} \cdot A \cdot W, \tag{25}$$

by a diagonal similarity transformation. The diagonal matrix W is positive definite and the elements $w_{(n,p),i}$ are given by $\exp(+\psi_i/2Ut)$ for electrons and $\exp(-\psi_i/2Ut)$ for holes. Since a similarity transformation leaves the spectrum of the matrix A unchanged, we have a nonsymmetric system of linear equations, in which A has a positive real spectrum. For local carrier temperature we are not aware of such a transformation, hence we cannot make statements of the spectrum of A in this case.

We note the enormous numerical range of the W matrices. For a maximum electrostatic potential of 100 Volts and liquid nitrogen temperature (77 K) we may expect exponents of the order $\log_{10}(w_{i,\max}) = 3275$. These numbers make the explicit transformation of the linear system undesirable on standard, double precision computer arithmetics. The very large rank of A gives preference to iterative methods over sparse Gaussian elimination.

5. Selected iterative methods for the linear systems. In this section we discuss the iterative procedures that were used in our computations. In the case of symmetrizability these are the classical conjugate gradient (CG) algorithm, then a variant of CG that circumvents the explicit symmetrization (SCG), and the conjugate gradient squared (CGS) procedure. For the nonsymmetrizable case we consider the generalized minimum residual algorithm (GMRES) and again CGS.

The numerical condition of the discrete carrier continuity equations can be rather poor [1], therefore efficient preconditioning is important. Incomplete (left or split) LU factorizations are used for preconditioning together with (left or symmetric) scaling by the main diagonal pivots $\tilde{D}$ of the preconditioner, and the Eisenstat procedure [7] is used to compute the preconditioned matrix-vector multiply. In §8, where various

TABLE 1
ILU-SCG.

Choose x_0
$\hat{r}_0 = Q_R^{-1} Q_L^{-1} (b - A x_0)$
$x_{-1} = \hat{r}_{-1} = 0$
$\hat{\rho}_0 = 1$
FOR $n = 0$ STEP 1 UNTIL convergence DO
$\hat{u} = W^{-2} Q_L Q_R \hat{r}_n$
$\hat{\sigma}_n = \langle \hat{u}, \hat{r}_n \rangle$
$\hat{v} = Q_R^{-1} Q_L^{-1} A \hat{r}_n$
$\hat{\gamma}_n = \dfrac{\hat{\sigma}_n}{\langle \hat{u}, \hat{v} \rangle}$
$\hat{\rho}_n = \left(1 - \dfrac{\hat{\gamma}_n}{\hat{\gamma}_{n-1}} \dfrac{\hat{\sigma}_n}{\hat{\sigma}_{n-1}} \dfrac{1}{\hat{\rho}_{n-1}}\right)^{-1}$ (if $n > 0$, $\hat{\rho}_0 = 1$)
$x_{n+1} = \hat{\rho}_n (x_n + \hat{\gamma}_n \hat{r}_n) + (1 - \hat{\rho}_n) x_{n-1}$
$\hat{r}_{n+1} = \hat{\rho}_n (\hat{r}_n - \hat{\gamma}_n \hat{v}) + (1 - \hat{\rho}_n) \hat{r}_{n-1}$
END FOR

numerical experiments will be presented, estimates for the extremal eigenvalues of the preconditioned matrix, obtained by the conjugate gradient method, illustrate that the preconditioned problem is well conditioned. We use the following notational conventions: The index s denotes the scaled matrix A_s and its strictly triangular parts L_s and U_s. Angle brackets denote the dot-product. Quantities with a hat ($\hat{}$) refer to the preconditioned system.

In this section we shall make explicit use of the similarity property, which was derived in the last section. In case of knowledge of the diagonal transformation matrices W, an explicit transformation (e.g., during the sparse matrix assembly) of the linear system into symmetric form is the most straightforward approach provided that the computer arithmetic is sufficiently accurate for the numbers generated by the similarity transformation. This transformation saves matrix storage and the classical preconditioned conjugate gradient algorithm (CG) is the optimal iterative method.

The very large number range in the iterates can be circumvented by a variant of the conjugate gradient algorithm, e.g., proposed in [11]. In this case the diagonal transformation matrices are confined to the inner products in the variant of the CG algorithm given in Table 1. An appropriate scaling can be employed for the inner products, thus avoiding restrictions on computer arithmetics. A three-term recursion is used with left preconditioning. The $W^T W Q^{-1} A$ norm of the solution error is minimized at each iteration step.

If the linear systems are not symmetrizable one must choose from a more general class of iterative methods for nonsymmetric linear systems. We concentrate on two methods: The generalized minimum residual method GMRES, an orthogonalization method, and the conjugate gradient squared method CGS, which has no (known) minimization property. Algorithms that were not considered are methods that require (e.g., dynamically computed) eigenvalue estimates, such as the Manteuffel algorithm, and the more recent hybrid methods, which adaptively switch between different acceleration schemes [6], [16].

The GMRES algorithm minimizes the two-norm of the residual at each iteration

TABLE 2
ILU-GMRES(m).

Choose $\hat{x}_0$
$\hat{b}_s = (I + L_s)^{-1} b_s$
Choose m
FOR $n = 0$ STEP 1 UNTIL convergence DO
  $\hat{t} = (I + U_s)^{-1} \hat{x}_n$
  $\hat{u} = \hat{t} + (I + L_s)^{-1} \left((diag\,(A_s) - 2I)\,\hat{t} + \hat{x}_n \right)$
  $\hat{r}_n = \hat{b}_s - \hat{u}$
  $\hat{\beta}_n = \|\hat{r}_n\|$
  $\hat{v}_1 = \dfrac{\hat{r}_n}{\hat{\beta}_n}$
  FOR $j = 1$ STEP 1 UNTIL $j = m$ DO
    $\hat{t} = (I + U_s)^{-1} \hat{v}_j$
    $\hat{u} = \hat{t} + (I + L_s)^{-1} \left((diag\,(A_s) - 2I)\,\hat{t} + \hat{v}_j \right)$
    FOR $i = 1$ STEP 1 UNTIL $i = j$ DO
      $\hat{h}_{i,j} = \langle \hat{u}, \hat{v}_i \rangle$
    END FOR
    $\hat{v}_{j+1} = \hat{u} - \sum_{i=1}^{j} \hat{h}_{i,j} \hat{v}_i$
    $\hat{h}_{j+1,j} = \|\hat{v}_{j+1}\|$
    $\hat{v}_{j+1} = \hat{v}_{j+1} / \hat{h}_{j+1,j}$
  END FOR
  Solve least squares problem $\|\hat{\beta}_n e_1 - \hat{\bar{H}}_m \hat{y}\|$ for $\hat{y}$
    with $e_1 = [1, 0, \cdots, 0]^T \in \boldsymbol{R}^{m+1}$, $\hat{\bar{H}}_m \in \boldsymbol{R}^{m+1 \times m}$
    (upper Hessenberg matrix consisting of the $\hat{h}_{i,j}$), $\hat{y} \in \boldsymbol{R}^m$
  $\hat{x}_{n+1} = \hat{x}_n + \hat{V}_m \hat{y}$
    with $\hat{V}_m = [\hat{v}_1, \hat{v}_2, \cdots, \hat{v}_m] \in \boldsymbol{R}^{N \times m}$
END FOR
$x_{n+1} = \tilde{D}^{-1/2} (I + U_s)^{-1} \hat{x}_{n+1}$

step. An orthonormal basis is built by an Arnoldi process, and the Hessenberg least squares problem can be solved, e.g., by Householder transformations. See Table 2. The monotonic convergence of GMRES has to be paid for. Full orthogonalization at iteration step n, which yields optimum convergence speed, requires storage of n vectors. This is prohibitive for large, three-dimensional problems and makes restarting of the iteration necessary, thus abandoning optimality. Nevertheless, the convergence of the restarted GMRES(m) is certainly monotonic. Values for the restarting frequency m less than, say, 6 have been found acceptable. Although the solution vector is updated every m iterations, the residual norm is available at each iteration step at no extra cost. This is a by-product of the QR decomposition for the solution of the least squares problem, if the Q and R matrices are updated at each iteration step.

A way of decreasing storage and arithmetic requirements is the use of Lanczos methods, such as the biconjugate gradient (BiCG) algorithm or the biconjugate gradient squared (CGS) algorithm [26] given in Table 3. Both algorithms construct approximations to the solution in the same Krylov subspace as GMRES, but a biorthogonality condition to the transposed system is used rather than an orthogonality condition to

TABLE 3
ILU-CGS

Choose x_0
$\hat{r}_0 = (I + L_s)^{-1}(b_s - A_s x_0)$
$\hat{x}_0 = (I + U_s)\tilde{D}^{1/2} x_0$
Choose $\hat{y}_0$ such that $\langle \hat{y}_0, \hat{r}_0 \rangle \neq 0$
$\hat{q}_0 = \hat{p}_{-1} = 0$
$\hat{\rho}_{-1} = 1$
FOR $n = 0$ STEP 1 UNTIL convergence DO
$\hat{\rho}_n = \langle \hat{y}_0, \hat{r}_n \rangle$
$\hat{\beta}_n = \dfrac{\hat{\rho}_n}{\hat{\rho}_{n-1}}$
$\hat{u} = \hat{r}_n + \hat{\beta}_n \hat{q}_n$
$\hat{p}_n = \hat{u} + \hat{\beta}_n \left(\hat{q}_n + \hat{\beta}_n \hat{p}_{n-1} \right)$
$\hat{t} = (I + U_s)^{-1} \hat{p}_n$
$\hat{v} = \hat{t} + (I + L_s)^{-1} \left((diag\,(A_s) - 2I)\,\hat{t} + \hat{p}_n \right)$
$\hat{\sigma}_n = \langle \hat{y}_0, \hat{v} \rangle$
$\hat{\alpha}_n = \dfrac{\hat{\rho}_n}{\hat{\sigma}_n}$
$\hat{q}_{n+1} = \hat{u} - \hat{\alpha}_n \hat{v}$
$\hat{u} = \hat{u} + \hat{q}_{n+1}$
$\hat{t} = (I + U_s)^{-1} \hat{u}$
$\hat{v} = \hat{t} + (I + L_s)^{-1} \left((diag\,(A_s) - 2I)\,\hat{t} + \hat{u} \right)$
$\hat{x}_{n+1} = \hat{x}_n + \hat{\alpha}_n \hat{u}$
$\hat{r}_{n+1} = \hat{r}_n - \hat{\alpha}_n \hat{v}$
END FOR
$x_{n+1} = \tilde{D}^{-1/2} (I + U_s)^{-1} \hat{x}_{n+1}$

construct the direction vectors. The residual does not decrease monotonically, hence the stopping procedure is more difficult compared to GMRES. The initial vector $\hat{y}_0$ may be chosen arbitrarily such that $\langle \hat{y}_0, \hat{r}_0 \rangle \neq 0$. We conform to the common practice to set this vector equal to the initial residual vector $\hat{r}_0$. A well-known property of this algorithm is the possibility of breakdown by vanishing of certain inner products. This phenomenon, which cannot be excluded a priori, is certainly a reason for worry. It is an experimental observation that effective preconditioning makes the event of (near) breakdown unlikely. We have observed that a sufficiently high machine precision can avoid breakdown occurring when the iteration is near the true solution.

Related squared Lanczos algorithms [10] (BIORES^2 and BIODIR^2) exist due to the analogy to the Lanczos-ORTHORES and Lanczos-ORTHODIR algorithms. These algorithms can be implemented to generate the same iterates for the solution vector (in exact arithmetic) if identical vectors x_0 and $\hat{y}_0$ are chosen. Though mathematically equivalent, unavoidable roundoff errors cause the iterates of the three biorthogonalization algorithms to drive apart in the course of the iteration. BIOMIN^2, i.e., CGS, not only proved to perform best under presence of roundoff, but could also be implemented most economically concerning storage.

In Table 4 the arithmetic work of the iterative procedures is listed. These figures refer to one linear iteration. For the restarted GMRES(m) method the iteration counter is incremented after the computation of one new orthogonal basis vector.

TABLE 4
Comparison of arithmetic work/iteration.

Solver	Ax	$\langle x, y \rangle$	$x + y$	αx
CG	1	2	4	10
SCG	1	3	4	10
CGS	2	2	7	6
GMR	1	$\frac{m}{2} + 1$	$\frac{m}{2}$	$\frac{m}{2} + 1$

6. Efficient preconditioning. We start with a comparison of two preconditioners that have been examined intensively: Block Jacobi preconditioning $P_J = D$, where D is the tridiagonal part of A, and incomplete factorization preconditioning [17], where

$$P_{ILU} = \left(L + \tilde{D}\right) \tilde{D}^{-1} \left(U + \tilde{D}\right) \tag{26}$$

with L the strictly lower and U the strictly upper triangular part of A and $\tilde{D}$ computed such that

$$diag\,(P_{ILU}) = diag\,(A)\,. \tag{27}$$

It is trivial to see that for the matrices under consideration the nonzero pattern of the LU factors of P_J is a proper subset of the nonzero pattern of the factors of P_{ILU}. It can be proven that in this case the ILU preconditioner is superior to the Jacobi preconditioner in the sense that if $A = P_J - R_J$ and $A = P_{ILU} - R_{ILU}$, then

$$\rho\left(P_{ILU}^{-1} R_{ILU}\right) \leq \rho\left(P_J^{-1} R_J\right) \tag{28}$$

where ρ denotes the spectral radius. Thus, a stationary iterative method based on the ILU splitting will converge at least as fast as a method based on a Jacobi splitting. Accelerated methods (see §5) will be influenced in a similar way. On the other hand, the arithmetic work for the Jacobi preconditioner compared with the ILU preconditioner is smaller and involves only first order recurrences, which can be vectorized more easily. Summarizing our experimental work, we state that the use of the Jacobi preconditioner seems to be limited to low bias voltage applications, i.e., examples where the diffusion term dominates in the current relations (4)–(6). For higher bias voltages unpleasant numerical effects have been observed (especially with the Lanczos methods), such as convergence stagnation or near breakdown in the Lanczos process at the beginning of the iteration. Another disadvantage is the orientation sensitivity due to the line elimination that forces the swapping of the matrix and vector elements to the most favorable direction, thus causing inconvenient computational overhead. A really clear relationship for both the minority and majority carrier continuity equations and the direction of the main current flow that would facilitate a detection of a favorable preconditioning orientation, however, could not be established. Therefore we cannot recommend this type of preconditioner for general use.

An incomplete factorization preconditioner of alternating direction type, P_T, with tridiagonal matrix factors, has been proposed in [5]. The basic idea is to use tridiagonal factors that lend themselves more to parallelization, rather than triangular factors as with ILU, at the same time maintaining the ILU sparsity pattern. For a seven-point stencil such a factorization would read

$$P_T = T_1 D^{-1} T_2 D^{-1} T_3, \tag{29}$$

where D is the main diagonal of A, and the T_i, $i = 1, 2, 3$, are tridiagonal matrices such that

$$T_1 + T_2 + T_3 = A + 2D. \tag{30}$$

The elements in the factors of P_T are altogether equal to the corresponding elements in A, hence there is no need to compute diagonal pivots $\tilde{D}$ as with ILU. However, there are more error terms outside the nonzero pattern in P_T; therefore it is no surprise that the iteration count in the iterative solver is higher compared with the ILU preconditioner. Computing $y = P_T^{-1}x$ involves the backsolves of the LU factors of three tridiagonal matrices. Since each tridiagonal matrix consists itself of many independent tridiagonal systems this work can be parallelized easily (e.g., by the partitioning method, cyclic reduction, etc.).

Let NX, NY, and NZ denote the number of gridlines in the respective directions. The inversion of T_1, which is assumed to contain the innermost diagonals of A, consists of $NY \cdot NZ$ independent tasks and has stride NX. T_3, which is assumed to contain the outermost diagonals of A, has $NX \cdot NY$ independent tasks and stride 1. For T_2 the situation is different because the stride is constant for NZ data sets only. The authors in [5] have reported that one preconditioning step with P_T can be performed up to three times as fast as one step in the hyperplane-ILU preconditioner on vector-supercomputers. Our numerical experiments, however, indicate that the convergence decrease with respect to ILU(0) is often larger than three even for simple examples. We refrained from an implementation on a supercomputer.

The most commonly used and probably most efficient preconditioner, at least on scalar computers, is incomplete LU factorization with allowable fill-in denoted by ILU(k), see, e.g., [2], [4], [12]. The index k denotes a controllable sparsity pattern along the matrix diagonals, $k = 0$ denoting no fill-in, $k = 1$ denoting fill-in caused by the original nonzero pattern but no further, and so on. As expected, a higher degree of fill-in reduces the iteration count. However, the number of operations for the factorization and for each iteration as well as the memory requirements increase considerably. For example, the ILU(1) preconditioner needs four extra diagonals within the original seven-diagonal nonzero pattern. For the fill-in ILU preconditioners we are not aware of a comparably efficient implementation to compute the preconditioned matrix vector multiply as proposed by Eisenstat for the ILU(0) [7], [29].

The two ILU(0) factorization variants under consideration are split ILU

$$\hat{A}_1\hat{x} \equiv \tilde{D}^{1/2}\left(\tilde{D}+L\right)^{-1} A \left(\tilde{D}+U\right)^{-1} \tilde{D}^{1/2}\hat{x} = \tilde{D}^{1/2}\left(\tilde{D}+L\right)^{-1} b \equiv \hat{b} \tag{31}$$

$$\hat{x} \equiv \tilde{D}^{-1/2}\left(\tilde{D}+U\right)^{-1} x \tag{32}$$

and left ILU

$$\hat{A}_2 x \equiv \left(\tilde{D}+U\right)^{-1} \tilde{D}\left(\tilde{D}+L\right)^{-1} Ax = \left(\tilde{D}+U\right)^{-1} \tilde{D}\left(\tilde{D}+L\right)^{-1} b \equiv \hat{b}. \tag{33}$$

The preconditioned matrix-vector multiply $\hat{A}_{1,2}\hat{p}$ can be simplified in the following manner. For the split ILU(0) one obtains, after having scaled the matrix symmetrically by $\tilde{D}$,

$$A_s = \tilde{D}^{-1/2} A \tilde{D}^{-1/2} \equiv diag\,(A_s) + L_s + U_s, \tag{34}$$

TABLE 5
Hyperplane-ILU(k) *on vectorcomputers.*

	VP-200			Cray-2			Alliant FX40		
k	B	Speedup	MFlop	B	Speedup	MFlop	B	Speedup	MFlop
0	4	13.75	96	14	4.30	27	212	1.34	1.8
1	8	12.12	96	26	4.76	30	327	1.51	2.3
2	8	12.62	128	30	5.93	34	410	1.64	2.5

$$\hat{A}_1\hat{p} = \left[\hat{t} + (I + L_s)^{-1}\left(\hat{p} - (2I - diag\,(A_s))\,\hat{t}\right)\right] \tag{35}$$

with

$$\hat{t} = (I + U_s)^{-1}\hat{p}, \tag{36}$$

and for the left ILU(0) together with left scaling

$$\hat{A}_2\hat{p} = (I + U_s)^{-1}\left[\hat{p} + (I + L_s)^{-1}\,(diag\,(A_s) - I + U_s)\,\hat{p}\right]. \tag{37}$$

In our implementation the split ILU(0) requires two additional scratch vectors and N square-roots, whereas the left ILU(0) requires no extra storage but a triangular matrix vector multiply.

Computing the diagonal pivots of the incomplete factorization such that $P_{ILU} - A$ has zero *column* sums (or row sums in the symmetric case) leads to modified incomplete factorization-type preconditioners (MILU) originated by [9]. A modification factor α in the interval $[0, 1]$ is usually introduced to smoothly sweep between pure ILU and full MILU factorization. Our results concerning the choice of such a modification factor do not admit a clear statement. It seems that $\alpha = 1$ always decreases the performance with respect to $\alpha = 0$ slightly. We found a number of device examples where a choice of $\alpha = 0.5$ yields a performance enhancement of about 10 percent to 30 percent concerning the iteration count. However, this gain is partly compensated by the higher arithmetic work for the factorization.

Parallelizable variants of ILU such as the Neumann polynomial preconditioner [31] were investigated as well. Numerical experiments carried out with the NSPCG (Non-Symmetric Preconditioned Conjugate Gradient) code [18] identified none of them competitive with ILU.

7. Implementation notes for parallel and vector computers. Vectorizing and parallelizing the backsolves of triangular or tridiagonal systems is a good exercise to explore computer architecture. We concentrate on vectorization techniques that are unlikely to degrade the performance of the incomplete factorization preconditioners; hence we do not consider multicolor orderings [20]. We further exclude computers that permit only unity-stride vector operations such as the CYBER 205 and disregard optimization measurements possibly required by memory bank and related conflicts. We aim at a production code that is as generally applicable as possible. This can be achieved by a reordering technique that does not change the preconditioner and produces long vector lengths. The *hyperplane* method, which is a plane-diagonalwise reordering, excellently reported in [2], [29], achieves this goal. The price for the rather general implementation we are aiming at is indirect addressing by list vectors.

If the unknowns as well as the matrix elements are indexed by $(i1, i2, i3)$ in the three spatial directions, then hyperplanes H_m (or "computational wavefronts") are

defined by the set of all mesh points in the simulation domain that satisfy the equation

$$i1 + (k+1)(i2 + i3) = m, \tag{38}$$

where m is constant. k denotes the level of fill-in. Obviously the computation of the unknowns in H_m can be carried out in parallel (and hence is a vector operation) since they depend only on the unknowns in H_{m-1} for the lower triangular or H_{m+1} for the upper triangular system, respectively.

For an implementation of the hyperplane method it is desirable to form a unique vector of the unknowns in a particular H_m. This is achieved by initially computing the addresses of the unknowns to be processed in an integer list vector $LIST$ and marking the beginning of each hyperplane by an additional list vector $LPTR$. The vectorlengths increase from 1 to $\mathcal{O}(NX \cdot NY \cdot NZ)^{2/3}$.

Special attention has to be paid to the meshpoints at the simulation boundary. Addressing (nonexisting) elements at the boundary points can be prevented by proper IF statements in the code or by computing the unknowns at the boundary outside the loop. We decided to extend the array of the unknowns such that unallowed addressing at the boundaries cannot happen. This is done by allocating an array of size $NX \cdot NY \cdot (NZ+2)$ for the vector of the unknowns $X(I)$ and filling the front and back plane of this vector with zeros. Then the code for the solution of the lower triangular system for $k = 0$ is surprisingly simple:

```
      DO 1 L=2,NX+NY+NZ-2
      DO 1 M=LPTR(L-1)+1,LPTR(L)
      I=LIST(M)
1     X(I)=R(I)-B(I)*X(I-1)-D(I)*X(I-NX)-F(I)*X(I-NX*NY).
```

R denotes the right-hand side and B, D, F the strictly lower triangular part of A_s. The inner loop is vectorizable. The implementation for the upper triangular system and the higher order recurrences ($k = 1, 2$) is straightforward. The approach sketched above is used in a similar manner to vectorize the ILU factorization at the beginning of the iteration.

Using the computational front approach, additional parallelism can be achieved by twisting the incomplete factorization in one specific direction [29], [30]. Then the factorization and the backsubstitutions can be performed concurrently from both ends to the center and from the center to both ends. Since such a twisting splits the domains into two equal halves, the mean hyperplane vectorlength in each half is decreased unless the number of meshpoints in the direction of splitting is significantly larger than the number of meshpoints in the remaining directions. It has been reported in [29] that the twisted hyperplane approach tends to decrease the iteration count in the linear solver, however, such an effect could not be verified with our type of equations. We think that the twisted hyperplane method is not advantageous, at least on a two-processor machine. To qualify the performance of the hyperplane-ILU(k) implementation, tests have been carried out on a Fujitsu VP200, a Cray-2, an Alliant FX40, and a Digital VAX 6260 computer. In Table 5 the CPU time for one triangular backsubstitution (B) in milliseconds (ms), the overall achieved speedup over the autovectorized code megaflop (MFlop) rate for this operation is given. This test example uses a $40 \times 40 \times 40$ grid and the measured numbers are mean values for 100 solves. For a larger number of meshpoints the values are even more favorable.

The values in Table 6 show the convergence speed improvement factors the level 1, 2 preconditioners must reach to beat the ILU(0) preconditioner. As can be seen,

TABLE 6
ILU-level 1, 2 preconditioner break-even points.

k	VP-200	Cray-2	Alliant FX-40
1	1.56	1.4	1.44
2	2.15	2.0	1.58

TABLE 7
Parallel hyperplane ILU(0) *on the VAX* 6260.

# Processors	1	2	3	4	5	6
MFlop	0.58	1.15	1.64	2.06	2.41	2.65
Speedup	1.00	1.98	2.82	3.54	4.14	4.56

these values tend to favour ILU(1), since practical values for the convergence speedups against ILU(0) are about 2. Thus, if memory usage is not too restrictive, ILU(1) is to be preferred to ILU(0).

In Table 7 the performance of ILU(0) on a Digital VAX 6260 with six scalar processors is presented. The megaflop rates and the speedup over one processor for the above test example are shown.

8. Numerical results. In this section the effectiveness of the preconditioned iterative methods outlined in the previous chapters will be demonstrated. We chose two simulations of silicon MOS-transistors. The finite-difference grids of both simulations are comparatively small, thus the "true" solution $\bar{x}$ of the linear systems were obtained by a sparse direct solver and the 2-norm of the relative solution error

$$e_n = \frac{\|\bar{x} - x_n\|}{\|\bar{x}\|} \tag{39}$$

was evaluated. A (heuristic) error threshold of $\zeta = 10^{-8}$ for e_n was found satisfactory in practical simulations. The computations with the CG and the SCG were performed with quad precision, the CGS and GMRES(5) iterations in double precision. These tests were carried out on a Digital VAX 8800, the precision of 1.0 is $1.387 \cdot 10^{-17}$ for double precision and $9.629 \cdot 10^{-35}$ for quad precision arithmetic. All data presented in the following have been extracted from version 5 of the device simulator MINIMOS [25].

Example 1 is a moderately nonplanar N-channel MOSFET with a channel length of 1.5 microns. The gate voltage is $U_g = 0.5$ Volts, the drain voltage is $U_d = 1.0$ Volt, all other terminal voltages are zero. The finite-difference grid is built self-adaptively and is small due to the low biasing: $23 \times 27 \times 16$ in x- (channel length), y- (pointing into the substrate), and z- (channel width) direction. Gummel's decoupling algorithm converges in five iterations. The terminal currents are small: $5.30 \cdot 10^{-9}$ Amperes for the drain current (I_D), and $-1.20 \cdot 10^{-13}$ Amperes for the bulk current (I_B). Carrier temperatures are judiciously neglected in this simulation, hence the nonsymmetric linear equations can conveniently be symmetrized and solved by the classical CG algorithm. The convergence of CG is compared with the symmetrized variant of the CG algorithm (SCG) and the (CGS) method. ILU(0) and ILU(1) are used as preconditioners. The CG algorithm provides cheap estimates of the extremal eigenvalues of the preconditioned matrix and hence for the spectral condition number $\kappa = \lambda_{\max}/\lambda_{\min}$.

TABLE 8
Example 1 : *Majority carrier continuity equation.*

	ILU(0) $\kappa = 35$			ILU(1) $\kappa = 10$		
N	I-CG	I-SCG	I-CGS	I-CG	I-SCG	I-CGS
1	17	32	25	9	22	12
2	34	29	24	20	22	11
3	29	29	22	18	21	11
4	33	29	24	20	21	11
5	21	41	21	15	26	11

TABLE 9
Example 1 : *Minority carrier continuity equation.*

	ILU(0) $\kappa = 102$			ILU(1) $\kappa = 70$		
N	I-CG	I-SCG	I-CGS	I-CG	I-SCG	I-CGS
1	50	84	48	41	45	35
2	54	78	43	43	40	32
3	53	78	39	42	39	30
4	54	68	42	42	37	27
5	51	86	47	40	46	35

In Table 8 the iteration history of the majority carriers (holes) are listed, in Table 9 the minorities (electrons) are listed. The spectral condition number estimate κ varies only marginally during the nonlinear iterations.

As can be seen from the tables, the iteration counts of the CG algorithm correspond nicely to the bounds suggested by the spectral condition number. The CGS algorithm converges twice as fast as the CG method for the best conditioned problem (majorities with ILU(1)).

Convergence curves for the three iterative methods with the matrices from the first nonlinear iteration are given in Figs. 1 and 2.

The second example is a P-channel MOSFET with bias voltages of $U_s = 0.0$ Volt, $U_g = -4.0$ Volts, $U_d = -1.0$ Volt, and $U_b = 2.0$ Volts. The drain current is $I_d = -6.3 \cdot 10^{-5}$ Amperes and the bulk current is $I_b = -1.7 \cdot 10^{-9}$ Amperes. This time electron and hole carrier temperatures are simulated self-consistently. First an initial solution is computed, which satisfies the classical semiconductor equations for constant carrier (i.e., ambient) temperature. This requires ten Gummel iterations. Subsequently the solution of the semiconductor equations with the extended drift-diffusion transport equations (see §2) is found by relaxation with respect to the local carrier temperatures. This requires seven further nonlinear iterations.

Within this second relaxation scheme, the linear systems are no longer diagonally similar to a symmetric matrix, hence CG and SCG are no longer applicable, but CGS and GMRES are. The restarting frequency m for GMRES is chosen to be 5 and the results are in columns $GMR(5)$. In Table 10 the iteration counts for the carrier relaxation cycles are given. Convergence curves for the three iterative methods with the matrices from the first nonlinear iteration (for locally varying carrier temperatures) are given in Figs. 3 and 4, respectively.

We report some timing results using ILU(0)-CGS. Both small examples execute quickly on supercomputers: Less than 30 seconds on the Fujitsu VP and slightly more than a minute on the Cray-2. This timing ratio is in good accordance with the megaflop counts (see §7) of the forward and backsolves of the triangular systems

TABLE 10
Example 2 : Carrier temperature relaxation.

	Majorities (electrons)				Minorities (holes)			
	ILU(0)		ILU(1)		ILU(0)		ILU(1)	
N	CGS	GMR(5)	CGS	GMR(5)	CGS	GMR(5)	CGS	GMR(5).
1	45	375	34	90	87	250	40	175
2	37	315	15	105	87	270	42	170
3	34	465	16	160	81	235	41	195
4	35	355	15	120	79	290	35	195
5	35	370	15	135	81	270	38	150
6	33	405	15	65	77	285	37	160
7	41	330	25	125	81	270	36	175

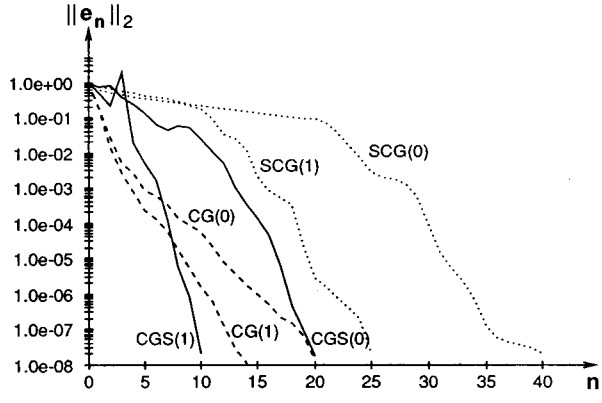

FIG. 1. *Convergence curves for the discrete majority continuity equation of Example* 1.

on these machines. A computationally complex example requires substantially more time, e.g., on the Fujitsu VP computer: A nonplanar N-channel MOSFET with $U_d =$ 5.0 Volts, $U_g = 1.0$ Volt, requires a grid of $55 \times 48 \times 30$ points. The grid loop and the computation of the initial solution requires 150 seconds. The subsequent nonlinear solution pass takes 80 Gummel iterations and a total of 1500 CPU seconds. The linear systems derived from the majority carrier continuity equation converge in roughly 70, from the minority carrier continuity equation 140 iterations (mean values). The same example computed on a fast scalar computer (NAS XL/80) takes approximately ten times more CPU time. The total time, however, is then increased substantially by references to secondary storage.

9. Conclusions. In this paper preconditioned iterative methods for the carrier continuity equations in three-dimensional device simulators have been studied and the performance of these algorithms on various high performance computing systems has been evaluated.

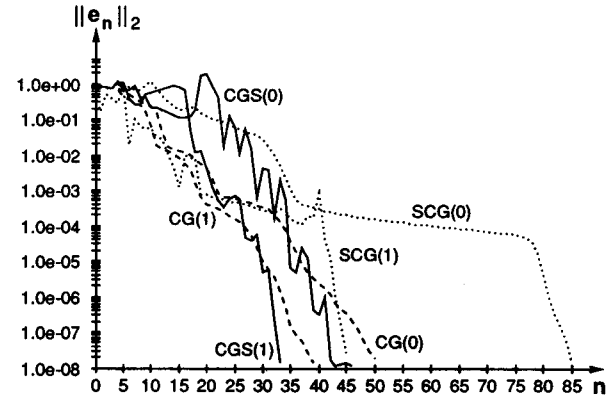

FIG. 2. *Convergence curves for the discrete minority continuity equation of Example* 1.

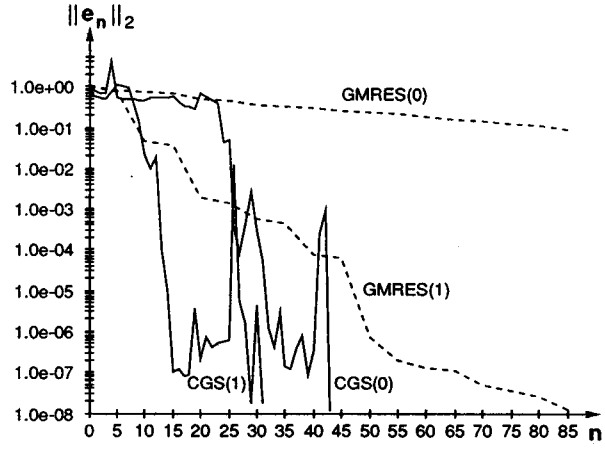

FIG. 3. *Convergence curves for the discrete majority continuity equation of Example* 2.

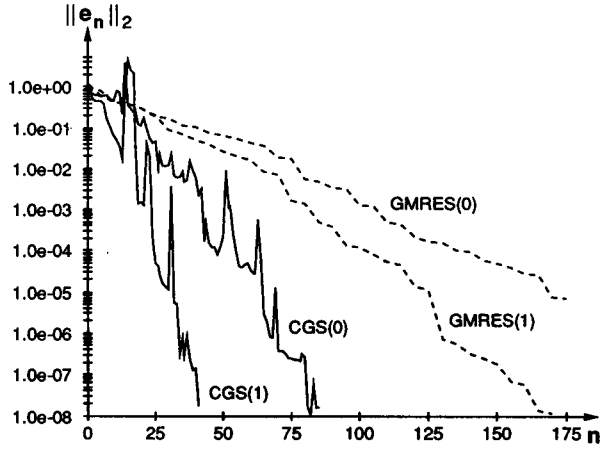

FIG. 4. *Convergence curves for the discrete minority continuity equation of Example* 2.

Among the iterative procedures we feel that the CGS method is the most versatile. It avoids possible numerical problems with the symmetrization matrices and is applicable to "real" nonsymmetric problems. Regarding convergence speed the GMRES(5) algorithm is clearly exceeded.

More decisive than the iterative (acceleration) procedures is the choice of a robust parallelizable preconditioner. We have investigated incomplete LU factorization of levels 0–2 and we have shown that quite a high performance (exceeding 100 megaflops on the Fujitsu VP200 computer) is reachable for the preconditioned matrix-vector multiply.

Acknowledgments. The authors are indebted to Martin Schubert and Hans-Peter Falkenburger from the Institute of Microelectronics, Stuttgart; Dr. Martin Thurner from the Campus-based Engineering Center Vienna; and H. Dietrich, G. Koessl, and H. Wiktorin from the Computer Services of Cooperate Research and Development, Siemens, Munich. The authors wish to thank the anonymous referees for valuable suggestions.

REFERENCES

[1] U. ASCHER, P. MARKOVICH, C. SCHMEISER, H. STEINRÜCK, AND R. WEISS, *Conditioning of the steady state semiconductor device problem*, SIAM J. Appl. Math., 49 (1989), pp. 165–185.

[2] C. C. ASHCRAFT AND R. G. GRIMES, *On vectorizing incomplete factorization and* SSOR *preconditioners*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 122–151.

[3] R. E. BANK, D. J. ROSE, AND W. FICHTNER, *Numerical methods for semiconductor device simulation*, IEEE ED-30 (1983), pp. 1031–1041.

[4] P. CONCUS, G. H. GOLUB, AND G. MEURANT, *Block preconditioning for the conjugate gradient method*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 220–252.

[5] S. DOI AND N. HARADA, *Tridiagonal factorization algorithm: A preconditioner for nonsymmetric system solving on vectorcomputers*, J. Inform. Process., 11 (1987), pp. 38–46.

[6] H. C. ELMAN, Y. SAAD, AND P. E. SAYLOR, *A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 840–855.

[7] S. C. EISENSTAT, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 1–4.

[8] H. K. GUMMEL, *A selfconsistent iterative scheme for one-dimensional steady state transistor calculations*, IEEE ED-11 (1964), pp. 455-465.

[9] I. GUSTAFSSON, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[10] M. H. GUTKNECHT, *The unsymmetric Lanczos algorithms and their relations to Padé approximations, continued fractions and the QD algorithm*, in Proceedings of the Second Copper Mountain Conference on Iterative Methods, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[11] L. H. HAGEMAN, F. T. LUK, AND D. M. YOUNG, *On the equivalence of certain iterative methods*, SIAM J. Numer. Anal., 17 (1980), pp. 852–873.

[12] K. HANE, *Supercomputing for process/device simulations*, in Proc. Sixth Internat. NASECODE Conference, J. J. H. Miller ed., Trinity College, Boole Press, Ltd., Dublin, Ireland, 1989, pp. 11–21.

[13] W. HÄNSCH AND S. SELBERHERR, MINIMOS 3: *A* MOSFET *simulator that includes energy balance*, IEEE ED-34 (1978), pp. 1074–1078.

[14] C. DEN HEIJER, *Preconditioned iterative methods for nonsymmetric linear systems*, in Proc. Int. Conf. on Simulation of Semiconductor Devices and Processes, Pineridge Press, Swansea, U.K., 1984, pp. 267–285.

[15] T. KERKHOVEN, *On the effectiveness of Gummel's method*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 48–60.

[16] T. A. MANTEUFFEL, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327.

[17] H. MEIJERINK AND H. VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[18] T. C. OPPE, W. D. JOUBERT, AND D. R. KINCAID, NSPCG *User's Guide*, Center of Numerical Analysis, University of Texas, Austin, TX, 1984.

[19] S. J. POLAK, C. DEN HEIJER, W. H. SCHILDERS, AND P. MARKOVICH, *Semiconductor device modelling from the numerical point of view*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 763–838.

[20] E. L. POOLE AND J. M. ORTEGA, *Multicolor* ICCG *methods for vector computers*, SIAM J. Numer. Anal., 24 (1987), pp. 1394–1418.

[21] C. S. RAFFERTY, M. R. PINTO, AND R. W. DUTTON, *Iterative Methods in Semiconductor Device Simulation*, IEEE ED-32 (1985), pp. 2018–2027.

[22] Y. SAAD AND M. H. SCHULTZ, GMRES: *A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[23] Y. SAAD, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1200–1232.

[24] D. L. SCHARFETTER AND H. K. GUMMEL, *Large-signal analysis of a silicon read diode oscillator.*, IEEE ED-16 (1969), pp. 64-77.

[25] S. SELBERHERR, *MINIMOS* 5 *Users's Guide*, Institute for Microelectronics, Technical University of Vienna, Vienna, Austria, 1990.

[26] P. SONNEVELD, CGS, *A fast Lanczos-type solver for nonsymmetric systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[27] M. THURNER P. LINDORFER AND S. SELBERHERR, *Numerical treatment of nonrectangular field-oxide for* 3D *MOSFET simulation*, IEEE CAD-9 (1990), pp. 1189–1197.

[28] T. TOYABE, H. MASUDA, Y. AOKI, H. SHUKURI, T. HAGIWARA, *Three-dimensional device simulator CADDETH with highly convergent matrix solution algorithms*, IEEE ED-32 (1985), pp. 2038–2044.

[29] H. VORST, *High performance preconditioning*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1174–1185.

[30] ———, *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers*, Parallel Comput., 5 (1987), pp. 45-54.

[31] ———, *A vectorizable variant of some* ICCG *methods*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 350–356.

[32] A. YOSHII, M. TOMIZAWA, AND K. YOKOYAMA, *Investigation of numerical algorithms in semiconductor device simulation*, Solid-State Electronics, 30 (1987), pp. 913-820.