Computer Physics Communications 53 (1989) 201–212
North-Holland, Amsterdam

ITERATIVE METHODS IN SEMICONDUCTOR DEVICE SIMULATION

R.E. BANK [1]

University of California at San Diego, La Jolla, CA 92093, USA

W.M. COUGHRAN, Jr. [2], M.A. DRISCOLL [3], R.K. SMITH [4]

AT&T Bell Laboratories, Murray Hill, NJ 07974, USA

and

W. FICHTNER [5]

Swiss Federal Institute of Technology, Integrated Systems Laboratory, CH-8092 Zürich, Switzerland

Received 28 September 1988; in revised form 8 November 1988

A number of Krylov-subspace methods and their applicability to the drift-diffusion equations for semiconductor devices are surveyed. Triangular (non-tensor-product) grids are appropriate for these highly nonsymmetric problems. The role of the underlying discretization is considered, including both the traditional box-method variant of the Scharfetter–Gummel scheme and a new upwinding (streamline-diffusion-like) procedure. Several preconditioners are discussed, including a novel alternate-block-factorization method. Graph coloring is used to enhance performance on vector computers.

1. Introduction

The continuing miniaturization of integrated circuit elements makes the conventional trial-and-error approach through experiments increasingly expensive and uneconomical. For this reason, computer simulations offer an interesting and challenging alternative. Such simulations allow new devices to be designed on a computer, predict a (possible) number of pathologies or shortcomings, characterize current–voltage relations for higher levels of simulation, and so forth. Device simulation requires the solution of difficult systems of nonlinear partial differential equations (PDEs). The commonly used set of modeling equations were introduced in 1950 by Van Roosbroeck [1]. Since then, these *drift–diffusion* equations have been popular for device analysis, but only in the last decade has it become feasible to solve the equations in two space dimensions. While the numerical effort needed is rather challenging, a number of effective methods have been developed that make these important simulations possible. There have been some books written on the subject of device simulation [2–4] and the proceedings of several thermatic meetings have appeared.

We have considered a number of Krylov-subspace (conjugate-gradient-like) methods [5–7] applied to the so-called drift–diffusion equations. Our approach involves specialized PDE discretization schemes and non-tensor-product grids, resulting in nonlinear and linear systems of equations that are difficult to solve. Based on our limited experience, we currently believe that bi-conjugate gradient methods coupled with specialized block

[1] irl45%igrad2@ucsd.edu.

[2] uunet!research!wmc or wmc@research.att.com.

[3] Present address: Merrill Lynch World Headquarters, North Tower, World Financial Center, New York, NY 10281, USA.

[4] uunet!research!barkeep!rksmith or rksmith@barkeep.att.com.

[5] uunet!mcvax!iis!fw or fw%ethz.ch@relay.cs.net.

0010-4655/89/$03.50 © Elsevier Science Publishers B.V.
(North-Holland Physics Publishing Division)

iterations and preconditioning allow an efficient solution of many device simulation problems.

Given the complexity of the subject and the brevity of this paper, we will not describe the device modeling problem in any detail [8]. Nevertheless, we believe that the matrices that arise from these problems are not trivially solved by Krylov-subspace methods. We would welcome further research into this specialized area. Hence, we are planning to put some representative matrices into the NETLIB collection [9], so that researchers interested in iterative methods can experiment with some new problems.

Section 2 presents the drift–diffusion equations for device modeling and mentions the spatial-discretization techniques that are used. Section 3 describes some of the technique employed to solve the nonlinear equations that arise. Section 4 discussions methods for the associated linear systems of equations. Section 5 presents some computational results, for a MOS field-effect transistor (MOSFET) and a Hall sensor, and draws some conclusions.

2. The drift-diffusion equations for device modeling

2.1. The continuous problem

A number of device simulators have been developed that approximately solve the drift–diffusion equations in a appropriate space–time domain, $\bar{\Omega} \times [0, T_0]$ where $\Omega \subset \mathbb{R}^2$ (or, more recently, $\Omega \subset \mathbb{R}^3$) and T_0 is the final time of interest, and with appropriate initial and boundary conditions. The equations provide values for the electrostatic potential, $\psi(x, t)$, as well as the electron and hole carrier concentrations, $n(x, t)$ and $p(x, t)$, where $x \in \Omega$. The drift–diffusion equations [1] may be written as

$$g_1(\psi, n, p) = -\nabla\cdot(\epsilon\nabla\psi) + q(n - p - N(x)) = 0, \tag{1}$$

$$g_2(\psi, n, p) = q\frac{\partial n}{\partial t} - \nabla\cdot J_n - qR_n(n, p) = 0, \tag{2}$$

$$g_3(\psi, n, p) = q\frac{\partial n}{\partial t} + \nabla\cdot J_p - qR_p(n, p) = 0, \tag{3}$$

where the electron and hole current densities are given by

$$J_n = -q\mu_n n\nabla\psi + qD_n\nabla n, \tag{4}$$

$$J_p = -q\mu_p p\nabla\psi - qD_p\nabla_p, \tag{5}$$

respectively, and ϵ, q, N and R_* are the dielectric constant, electron charge, net impurity (doping) concentration and recombination–generation terms, respectively. In eqs. (4) and (5), $\mu_*(x, \nabla\psi)$ and $D_*(x, \nabla\psi)$ are the mobilities and diffusion coefficients, respectively. In oxide regions, eqs. (1)–(5) are replaced by

$$-\nabla\cdot(\epsilon_0\nabla\psi) = 0. \tag{6}$$

(In addition to ψ, n and p, currents flowing out of boundary segments can be derived via line integrals of current densities.)

Eqs. (1)–(6) can be augmented by additional Kirchhoff equations for lumped extrinsic circuit elements [8,10–12]. The equations have been extended to include magnetic effects [13].

Eqs. (1)–(6) are typically normalized by scaling procedures, such as proposed by deMari [14,15] or Markovich [4] (also see ref. [3]). The form of mobility and diffusion coefficients that appear in eqs. (4) and (5) are complicated functions and are subject to some debate, see refs. [16,3,17]. Some model forms for the recombination–generation terms in eqs. (2) and (3) are discussed, for example, in refs. [16,3,17].

In this paper, we will primarily discuss the static problem. That is, we will assume

$$\frac{\partial n}{\partial t} = \frac{\partial p}{\partial t} = 0, \tag{7}$$

unless otherwise noted. Eqs. (1)–(6) then represent a coupled system of boundary-value problems.

We will not discuss the appropriate boundary conditions for eqs. (1)–(6) in any detail (see refs. [2–4]). The boundary conditions are Dirichlet on the terminal contacts and Neumann elsewhere. (For example, the terminal contacts of a MOSFET are at the source, gate, drain and substrate.)

Eqs. (1)–(6) are written in terms of the primitive variables, ψ, n and p. The primitive variables have the advantage that (1) is nonlinear in ψ but eqs. (2) and (3) are linear in n and p, respectively.

except for the recombination terms, R_*, and possibly the mobility and diffusion coefficients. Disadvantages of the primitive variables include: (1) the huge range of n and p; and (2) the necessity of ensuring $n > 0$ and $p > 0$ computationally. Another choice of variables is used when the so-called Einstein relation holds. We will use tildes above quantities to denote the deMari scaled versions of the quantities appearing in eqs. (1)–(6). The Einstein relation then amounts to the assumption that $\tilde{D}_* = \tilde{\mu}_*$ and the quasi-Fermi variables, $\tilde{\psi}$, v and w, are defined as

$$\tilde{n} = e^{\tilde{\psi} - v}, \tag{8}$$

$$\tilde{p} = e^{w - \tilde{\psi}}. \tag{9}$$

The static drift–diffusion equations become

$$-\Delta\tilde{\psi} + e^{u-v} - e^{w-\tilde{\psi}} - \tilde{N} = 0, \tag{10}$$

$$\nabla \cdot \left(\tilde{\mu}_n \, e^{\tilde{\psi}-v} \nabla v \right) - \tilde{R}_n = 0, \tag{11}$$

$$\nabla \cdot \left(-\tilde{\mu}_p \, e^{w-\tilde{\psi}} \nabla w \right) - \tilde{R}_p = 0. \tag{12}$$

The advantages of writing the problem in terms of quasi-Fermi levels include: (1) compression of the range of the dependent variables; (2) a guarantee that $n > 0$ and $p > 0$; and (3) the form of the equations simplify somewhat as do the matrices. The main disadvantage is that eqs. (11) and (12) are completely nonlinear in v and w, respectively. A number of other choices of variables for this problem has been suggested, such as the Slotboom variables, exponentials of the quasi-Fermi variables and a vorticity/stream-function formulation. We will largely confine our discussion to the primitive variables, but refer the reader to refs. [18,19,2,20,3,4] for a more detailed discussion.

2.2. *The discrete problem*

The domain Ω is often irregular. Moreover, sharp gradients occur in the doping profile, N, as well as in the solution variables. In particular, N may change by ten orders of magnitude in a small spatial region. Generally, there is no expectation that N and the solution variables change primarily in one variable at a time, that is, ∇N need not be primarily parallel to one of the coordinate axes when $\| \nabla N \|_2$ is large. This implies that spatial discretizations based on tensor-product grids tend to waste grid points over a more general grid-generation procedure. (Tensor-product grids have been used in a number of device simulators, for example, see ref. [21].)

We have found general triangular grids to be effective when $\Omega \subset \mathbb{R}^2$ is polygonal [19,22,10]. This technology has been proven for single elliptic equations in Bank's PLTMG code [23,24]. The advantages include: (1) the ability to deal with sharp changes in the solution without realigning the grid; (2) ease of locally refining the grid; and (3) the use of substantially fewer grid points as compared to tensor-product grids. The disadvantages include: (1) sensitivity to the angles in the triangulation [25]; (2) the introduction of grid-orientation effects [11]; and (3) an increase in the implementation complexity. We believe the advantages outweigh the disadvantages. These techniques have been employed in the PADRE device simulator being developed within AT&T Bell Laboratories (which is a successor to the DEVICE [8,10] and PISCES [26,27] simulators) and the GENSIM simulator being developed at the Integrated Systems Laboratory of the Swiss Federal Institute of Technology (which is also a successor to DEVICE).

The traditional method of discretizing eqs. (1)–(6) (or eqs. (10)–(12) for that matter) is through a box-method generalization [28] or a finite-element scheme [18]. For the Poisson eqs. (1) and (6), either a box or finite-element discretization [28,25] can be used for discretization as long as the charge terms, $q(n - p - N)$, are well represented. The difficult equations to discretize are the continuity eqs. (2)–(5). The problem is that the solution may change dramatically over a small interval; by small, we mean that there may be dramatic changes over distances of $\mathcal{O}(10^{-7})$ cm while a linear dimension of Ω may be $\mathcal{O}(10^{-4})$ cm. However, it is known that the current densities, J_*, in eqs. (4) and (5) vary slowly – locally, they can be approximated by a constant. Scharfetter and Gummel [29] took advantage of this property to construct an exponentially-fitted method that is exact in one-dimension if, on each interval, μ_*, D_* and J_* are assumed constant.

This method can be generalized to two (and three) dimensions [18,10].

The Scharfetter–Gummel discretization in two dimensions is known to be sensitive to the angles in the triangulation [30,22]. For general polygonal domains, it is difficult to insure that the triangulation is nonobtuse (no triangle having a obtuse angle), which is a sufficient condition for the stability of the discretization. It is possible to generate a nonobtuse triangulation [31], but the adaptive-grid techniques used make this approach problematical. There is a corresponding problem for tetrahedral grids in three dimensions, $\Omega \subset \mathbb{R}^3$ [32]; it is extremely difficult to generate a graded tetrahedral mesh with specialized geometry properties to stabilize the Scharfetter–Gummel discretization.

Given that the Scharfetter–Gummel discretization interacts poorly with the underlying triangulation, some effort has been made to develop another method that is much more resistant to bad triangle properties [33]. This method is related to the upwinded streamline-diffusion algorithm [34]. It also resolves an important problem by making it easy to accurately evaluate $\nabla\psi \cdot J_*$ on a triangular grid; this quantity is often incorporated in the mobilities, μ_*, and is difficult to approximate with Scharfetter–Gummel. This approach, unlike Scharfetter–Gummel, can add an arbitrary off-diagonal contribution to the linearized systems that arise with Newton methods (section 3). Hence, the additional stability of this discretization can give rise to more poorly conditioned matrices.

3. Algorithms for the nonlinear problem

There are two basic approaches that have been used to solve the discretized versions of eqs. (1)–(6) or (10)–(12): Newton-like and Gummel (Nonlinear Gauss–Seidel) iterations. (We will refer to these as the coupled and plug-in approaches, respectively.) The nonlinear equations are of the form:

$$g_1(\psi, n, p) = 0, \tag{13}$$

$$g_2(\psi, n, p) = 0, \tag{14}$$

$$g_3(\psi, n, p) = 0. \tag{15}$$

The Newton-like (coupled) methods that we have advocated in the past [35,18] are damped methods for the solution of the coupled form of eqs. (13)–(15),

$$g(z) = 0, \tag{16}$$

where $z = (\psi^T, n^T, p^T)^T$ and $g(z) = (g_1^T(z), g_2^T(z), g_3^T(z))^T$. Given an initial guess z_0, the algorithm solves

$$g_k' \, \Delta z_k = -g_k, \tag{17}$$

where $g_k = g(z_k)$ and $g_k' = g'(z_k)$, and then computes an updated value

$$z_{k+1} = z_k + s_k z_k. \tag{18}$$

The damping parameter, $0 < s_k \le 1$, is chosen to satisfy a sufficient-decrease condition

$$1 - \frac{\| g_{k+1} \|}{\| g_k \|} > \epsilon_M s_k, \tag{19}$$

where ϵ_M is the machine epsilon. There are various strategies for selecting s_k to satisfy eq. (19) [35,36]. This algorithm is known to be globally, and quadratically, convergent if $\|(g_k')^{-1}\| \le M < \infty$ and certain other technical assumptions are met [35]. There is also a two-parameter method where eq. (17) is replaced by

$$(g_k' + \lambda_k \| g_k \| I) \, \Delta z_k = -g_k. \tag{20}$$

The one-parameter (eqs. (17) and (18)) and two-parameter (eqs. (20) and (18)) damped-Newton methods have been proven effective in device and circuit simulation [18,36,8,10,37].

It is not necessary to solve eq. (17) or (20) exactly. For example, eq. (17) can be replaced by an inner iteration

$$M_k(\Delta z_{km} - \Delta z_{k,m-1}) = -(g_k' \, \Delta z_{k,m-1} + g_k), \tag{21}$$

where $\Delta \mathrm{z}_{\mathrm{k0}} = 0$ and $\Delta \mathrm{z}_{\mathrm{k}} = \Delta \mathrm{z}_{\mathrm{km_k}}$ for some m_k. This inner iteration is terminated when

$$\frac{\| g_k' \, \Delta z_{km} + g_k \|}{\| g_k \|} \le \alpha \left(\frac{\| g_k \|}{\| g_0 \|} \right)^{\beta}, \tag{22}$$

where $0 < \alpha < 1$ and $0 < \beta \le 1$ are experimentally determined. This Newton–Richardson procedure is expected to converge superlinearly with a $(1+\beta)$ Q-rate of convergence if $\| I - M_k^{-1} g_k' \| \le \rho < 1$ for all k [35] (also see ref. [38]). Less costly, but more slowly convergent, methods may also be used, such as a chord method [38].

The plug-in iteration due to Gummel is the second major approach [39] in the device modeling literature; it is known as nonlinear Gauss–Seidel in the numerical analysis literature [38]. The procedure starts with an initial guess (ψ_0, n_0, p_0) and then improves the solution by successively solving each equation in (13)–(15) by

$$g_1((\psi_k, n_k, p_k) \mapsto \psi_{k+1}) = 0, \tag{23}$$

$$g_2((\psi_{k+1}, n_k, p_k) \mapsto n_{k+1}) = 0, \tag{24}$$

$$g_3((\psi_{k+1}, n_{k+1}, p_k) \mapsto p_{k+1}) = 0. \tag{25}$$

Here, the variables to the left of $\mapsto$ are input and the variable to the right is being solved for. This amounts to applying a damped-Newton method to eq. (23) for ψ, then solving eq. (24) for n, and finally solving eq. (25) for p. (Once again the same technique is applicable to the system in quasi-Fermi variables, eqs. (10)–(12).) It is well-known that the convergence of the this (decoupled) plug-in iteration deteriorates when the equations are tightly coupled, which corresponds to high-current regimes.

Finally, it should be noted that including the derivatives of the mobility functions, μ_* from eqs. (4) and (5), in the Jacobians can cause convergence difficulties with the one-parameter damping method, eqs. (17) and (18). These difficulties arise when the Jacobian and residual are evaluated for an iterate far from the solution. Most codes exclude the derivatives of the mobilities until the iterates are sufficiently close to the solution. This amounts to using an approximate Newton method in either the coupled or plug-in case.

4. Algorithms for the linearized problem

4.1. Data structure

Let a triangulation T of the domain Ω be given. The discretization (section 2.2) provides linkages (edges) between the vertices of T. The matrices that arise are nonsymmetric but the pattern of nonzeros is symmetric, that is, the matrices are said to be structurally symmetric ($a_{ji} \neq 0$ implies $a_{ij} \neq 0$). The data structure that we have used for some time [8,10] is a variant of the Yale Sparse Matrix Package (YSMP) data structure [40,41].

Let $A \in \mathbb{R}^{n \times n}$ be a sparse matrix associated with the discretization of a single equation on T. (Here, n is the number of vertices in T not associated with Dirichlet boundary conditions, that is, n is the number of unknowns.) Let η 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 $n+1$ if the matrix is diagonal, $\eta + n + 1$ if the matrix is (value) symmetric, and $2\eta + n + 1$ if the matrix is nonsymmetric.

Let r_i be the number of nonzeroes in the strict upper triangular part of row i so $\eta = \Sigma_{i=1}^n r_i$. Then the entries of JA and A are defined as follows:

$$\mathrm{JA}_1 = n + 2, \tag{26}$$

$$\mathrm{JA}_{i+1} = \mathrm{JA}_i + r_i \quad \text{for } 1 \le i \le n, \tag{27}$$

$$\mathrm{A}_i = a_{ii} \quad \text{for } 1 \le i \le n, \tag{28}$$

$$\mathrm{A}_{n+1} = \begin{cases} -1 & \text{for diagonal matrices,} \\ 0 & \text{for symmetric matrices,} \\ \eta & \text{for nonsymmetric matrices,} \end{cases} \tag{29}$$

$$\mathrm{JA}_k = j \ (\text{column index of } a_{ij})$$
$$\text{for } \mathrm{JA}_i \le k < \mathrm{JA}_{i+1},\ 1 \le i \le n. \tag{30}$$

$$\mathrm{A}_k = a_{ij} \quad \text{for } \mathrm{JA}_i \le k < \mathrm{JA}_{i+1},\ 1 \le i \le n, \tag{31}$$

and if the matrix is not symmetric

$$\mathrm{A}_{k+\eta} = a_{ji} \quad \text{for } \mathrm{JA}_i \le k < \mathrm{JA}_{i+1},\ 1 \le i \le n. \tag{32}$$

JA_1 through JA_{n+1} capture the same information as YSMP's IA array. The remainder of JA corresponds to YSMP's JA array for symmetric matrices. 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. The matrix type is specified by A_{n+1}. In A, the diagonal is stored first, followed by the strict upper triangle stored row-wise. If the matrix is nonsymmetric, the strict lower triangle stored column-wise completes A.

Up to this point, we have discussed storing an $A \in \mathbb{R}^{n \times n}$ that arises from the discretization of a single elliptic equation. The semiconductor equations are a coupled system of 3 equations so the global Jacobian associated with eq. (16) is a $3n \times 3n$ matrix,

$$G = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{23} & A_{33} \end{bmatrix}. \tag{33}$$

With the traditional YSMP data structure, JA would be expanded to include the grid connectivity and equation interdependence embodied in G. With our data structure, JA only represents the grid connectivity; rather, there are 9 (3×3) A arrays that capture the corresponding elements of A_{ij} from above. It is also possible to specialize the type of each A_{ij} in eq. (33) above so some can be diagonal, symmetric or nonsymmetric.

In earlier work, we have discussed how this data structure facilitates a high performance sparse direct solver, based on Crout reduction [6], on a vector computer [8,10]. We also discussed the application of such procedures to general routines for forming Gv or $G^{\mathrm{T}}v$ given v, computing incomplete factorizations [42–44] (also see refs. [5,45]), and for preconditioned ORTHOMIN acceleration [46] (also see refs. [5,7]).

Preconditioned iterative methods do not always perform well on vector architectures, such as the Cray X-MP. In 1982, Schreiber and Tang suggested graph coloring [47] as a method of vectorizing preconditioned Krylov-subspace methods [48] (also see refs. [49–51]). Given the grid connectivity, we color the graph associated with a single equation using a 'greedy' algorithm [47]. This results in a block matrix with tightly banded blocks [49] and is highly vectorizable. On a single processor of a Cray X-MP, 'coloring' can decrease processor times by over a factor of ten; this runtime improvement overcomes the possible increase in the number of iterations necessary because of the reordering [49,50,52]. The coloring ideas can be extended to the block system, eq. (33).

4.2. Iterative methods

A number of Krylov-subspace methods have been suggested for solving a nonsymmetric system of linear equations

$$Ax = b. \tag{34}$$

These methods are all based on forming Av or $A^{\mathrm{T}}v$, given v, and include: BICG [53,54], ORTHOMIN [46], IOM [55,56], GCR [5], GMRES [57], CGS [58], and many others as well as extensions to the basic methods [5,7]. We will not discuss each method in detail, but note that a number of these methods have been applied to the drift-diffusion equations [59,22,60]. Elman [5] and Saad and Schultz [54,7] have developed an appropriate framework for studying the various methods.

Until recently, we favored the ORTHOMIN(k) algorithm, possibly increasing the number of back vectors k when convergence difficulties arise. The method has a local minimization property and is simple to implement. There is substantial interest in the GMRES algorithm, which also has a nice local minimization property. Moreover, Kerkhoven and Saad [60] have produced some interesting results using GMRES(k) and suggested a nonlinear variant. Hence, we have conducted a study of the methods listed above as applied to the semiconductor equations.

We summarize our experience here, although section 5 describes some of our computational experiments. The BICG and CGS algorithms are remarkably robust, even at the start of the nonlinear iteration (coupled Newton or plug-in – see section 3) when the matrices are poorly conditioned. The other algorithms sometimes fail to converge for hard problems unless a large number of back vectors are included, such as GMRES(k) for $k = 5$ or 10 [60]. We believe that our triangular (non-tensor-product) grids, discretization methods (section 2.2), ordering techniques (section 4.3) and variable mobilities, μ_*, make these equations extremely difficult to solve.

Given our experience, we will summarize the BICG and CGS algorithms. Let x_0 be the given initial solution guess and set $r_0 = b - Ax_0$. Following ref. [54], we note that the BICG method is derived from the Lanczos algorithm [61] and produces a solution, $x_0 + z_m$ such that $z_m \in \operatorname{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$ and $(r_0 - Az_m) \perp \operatorname{span}\{r_0, A^{\mathrm{T}}r_0, \ldots, (A^{\mathrm{T}})^{m-1}r_0\}$. The algorithm, as presented by Fletcher [53], starts by setting $\tilde{p}_0 = \tilde{r}_0 = p_0 = r_0$. The following is then iterated for $k = 0, 1, 2, \ldots$ until done:

$$\alpha_k = r_k^{\mathrm{T}}\tilde{r}_k / p_k^{\mathrm{T}}A^{\mathrm{T}}\tilde{p}_k, \tag{35}$$

$$x_{k+1} = x_k + \alpha_k p_k, \tag{36}$$

$$r_{k+1} = r_k - \alpha_k A p_k, \tag{37}$$

$$\tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^{\mathrm{T}}\tilde{p}_k, \tag{38}$$

$$\beta_k = r_{k+1}^{\mathrm{T}}\tilde{r}_{k+1} / r_k^{\mathrm{T}}\tilde{r}_k, \tag{39}$$

$$p_{k+1} = r_{k+1} + \beta_k p_k, \tag{40}$$

$$\tilde{p}_{k+1} = \tilde{r}_{k+1} + \beta_k \tilde{p}_k. \tag{41}$$

This procedure computes

$$x_m = x_0 + \sum_{i=1}^{m} \gamma_i A^{i-1} r_0 \tag{42}$$

such that, for $j = 1, 2, \ldots, m$,

$$\left(r_0 - A\sum_{i=1}^{m}\gamma_i A^{i-1} r_0\right)^{\mathrm{T}}\left((A^{\mathrm{T}})^{j-1} r_0\right) = 0. \tag{43}$$

The preconditioned form of this algorithm requires two matrix multiplications per iteration and six vectors of length n. (See refs. [53,54] for further details regarding BICG.)

The CGS algorithm essentially squares the optimal polynomial in eq. (43) but requires an additional n-vector [58,59]. For completeness, we now describe the algorithm in some detail. Let $q_0 = p_{-1} = 0$. The following is then iterated for $k = 0, 1, 2, \ldots$ until done:

$$\beta = \begin{cases} r_0^{\mathrm{T}} r_k & \text{if } k = 0, \\ (r_0^{\mathrm{T}} r_k)/(r_0^{\mathrm{T}} r_{k-1}) & \text{otherwise,} \end{cases} \tag{44}$$

$$u_k = r_k + \beta_k q_k, \tag{45}$$

$$p_k = u_k + \beta_k(q_k + \beta_k p_{k-1}), \tag{46}$$

$$\alpha_k = r_0^{\mathrm{T}} r_k / r_0^{\mathrm{T}} A p_k, \tag{47}$$

$$q_{k+1} = u_k - \alpha_k A p_k, \tag{48}$$

$$r_{k+1} = r_k - \alpha_k A(u_k + q_{k+1}), \tag{49}$$

$$x_{k+1} = x_k + \alpha_k(u_k + q_{k+1}). \tag{50}$$

The advantages of the bi-conjugate methods include: (1) simple recurrence relations; (2) (low) fixed storage requirements; and (3) extreme robustness. The low storage requirements are becoming more important as three-dimensional simulators are being constructed. The disadvantages include: (1) no local descent property so termination criteria can be subtle; and (2) less satisfying theoretical understanding. Nevertheless, we believe the bi-conjugate methods are to be preferred for these problems.

4.3. Preconditioners

We will now mention three preconditioners that may be used to precondition Krylov-space algorithms: (1) SSOR [28]; (2) incomplete factorizations (ILU) [42–44,5,45]; and (3) an alternate-block factorization (ABF) [62]. (Appropriate modified incomplete factorizations represent another possible class of preconditioners.) We normally precondition on the right so the system is of the form:

$$Ax = [AP][P^{-1}x] = \tilde{A}y = b. \tag{51}$$

Our current implementations of SSOR and incomplete factorization are based on the ideas presented in section 4.1 and refs. [8,63,10]. Hence, we will not go into detail here. Note that our non-tensor-product grids make line SSOR preconditioners [59] less attractive.

Let us now consider the ABF idea, proposed by Bank and Smith, by examining a 2 × 2 block system as would arise from a system of two partial differential equations,

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \tag{52}$$

where each $A_{ij} \in \mathbb{R}^{n \times n}$. Let $D \in \mathbb{R}^{2n \times 2n}$ be the block 2 × 2 matrix made up of the diagonals of the blocks, that is,

$$D = \begin{bmatrix} \operatorname{diag}(A_{11}) & \operatorname{diag}(A_{12}) \\ \operatorname{diag}(A_{21}) & \operatorname{diag}(A_{22}) \end{bmatrix}. \tag{53}$$

There is a permutation matrix, P, such that

$$\tilde{A} = PAP^{\mathrm{T}} = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} & \cdots & \tilde{A}_{1n} \\ \tilde{A}_{21} & \tilde{A}_{22} & \cdots & \tilde{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{A}_{n1} & \tilde{A}_{n2} & \cdots & \tilde{A}_{nn} \end{bmatrix}, \tag{54}$$

where

$$\tilde{A}_{ij} = \begin{bmatrix} (A_{11})_{ij} & (A_{12})_{ij} \\ (A_{21})_{ij} & (A_{22})_{ij} \end{bmatrix} \in \mathbb{R}^{2\times 2}. \tag{55}$$

Here, A is the matrix blocked by equation while $\tilde{A}$ is blocked by grid point; $\tilde{A}$ represents the *alternate blocking.* Let

$$\tilde{D} = PDP^{\mathrm{T}}. \tag{56}$$

Obviously, $\tilde{D}^{-1}v$ can be computed using dense matrix techniques since it consists of 2 × 2 matrices on its diagonal. Assuming $\tilde{D}^{-1}$ exists, the ABF preconditioned matrix is

$$\begin{aligned} AD^{-1} &= A(P^{\mathrm{T}}\tilde{D}^{-1}P) \\ &= \begin{bmatrix} (A_{11}D_{22} - A_{12}D_{21})\delta & (A_{12}D_{11} - A_{11}D_{12})\delta \\ (A_{21}D_{22} - A_{22}D_{21})\delta & (A_{22}D_{11} - A_{21}D_{12})\delta \end{bmatrix}, \end{aligned} \tag{57}$$

where

$$\delta = (D_{11}D_{22} - D_{21}D_{12})^{-1}. \tag{58}$$

The basic idea of ABF is to partially decouple the system by reducing the effect of the off-diagonal blocks. For the one-carrier semiconductor equations, the matrix in eq. (52) becomes

$$\begin{bmatrix} -\Delta & I \\ -M & C \end{bmatrix}, \tag{59}$$

where Δ is a discrete Laplacian, M is symmetric and positive definite because of physical considerations, and C is a discretization of the convection-diffusion term that arises from the drift–diffusion equation. For a tightly coupled system, the effect of the preconditioner given by eqs. (57) and (58) is to de-emphasize the nonsymmetric portions of the diagonal blocks in eq. (59) while reducing the size of the off-diagonal blocks. We will discuss ABF in more detail and generality, including a nonlinear variant, later [62].

SSOR and ILU are applicable as preconditioners for the linear systems that arise from either the nonlinear plug-in or coupled Newton method (section 3). The ABF preconditioner is obviously applicable to the coupled Newton method since the underlying linear system then has a natural 3 × 3 block structure; in that case, we always solve the preconditioned system with (3 × 3) block Gauss–Seidel. A sparse direct or preconditioned iterative method is used for the inner Gauss–Seidel solves arising from the coupled-ABF procedure.

5. Applications and conclusions

Before discussing applications, let us consider some of the details of the specific algorithms applied. With one exception, the equations are discretized using the Scharfetter–Gummel method (section 2.1). The one-parameter damping method, eqs. (17) and (18), is used for both the coupled-Newton and plug-in nonlinear algorithms (section 3). The nonlinear iterations are terminated when a relative 2-norm error of 10^{-4} is obtained in the solutions. The Newton–Richardson method, eqs. (21) and (22), is used to control the number of Gauss–Seidel iterations for the coupled-ABF method (section 4). After limited experimentation, we currently believe that $\beta \approx 0$ and a modest value of α, say $0.5 \leq \alpha < 1$, are the appropriate values to use in eq. (22); by this, we mean values that minimize the overall nonlinear solve time. We only show results for the BICG and CGS algorithms as the linear solvers for both plug-in and the Gauss–Seidel blocks in coupled-ABF. The BICG and CGS algorithms are stopped when a scaled 2-norm residual is less than 10^{-6}; this could be improved. The nonlinear iterative methods are all started with an initial guess based on a physically-motivated 'quasi-neutral' approximation. In each case, two initial plug-in loops are used to 'smooth' the initial guess, but the work required for this smoothing procedure is not included in the reported results. (In practice, it is common to use 5 or 10 plug-in loops to smooth the initial guess before a coupled Newton al-

e detail and gener-
:iant, later [62].
cable as precondi-
ıat arise from either
ed Newton method
itioner is obviously
wton method since
then has a natural
ıse, we always solve
vith (3 × 3) block
or preconditioned
inner Gauss–Seidel
l-ABF procedure.

ıns, let us consider
specific algorithms
the equations are
r–Gummel method
r damping method,
both the coupled-
algorithms (section
terminated when a
is obtained in the
dson method, eqs.
rol the number of
the coupled-ABF
d experimentation,
≠ 0 and a modest
re the appropriate
is, we mean values
ıear solve time. We
CG and CGS al-
ır both plug-in and
:oupled-ABF. The
e stopped when a
; than 10^{-6}; this
ear iterative meth-
ıl guess based on a
utral' approxima-
plug-in loops are
ess, but the work
ocedure is not in-
(In practice, it is
ı loops to smooth
ıpled Newton al-

gorithm is applied, which has recently been justified by Kerkhoven [64]; this procedure substantially improves the performance of the coupled method.) We will now consider the application of the methods described above to the simulation of two semiconductor devices.

The first device simulated is a 0.5 μm MOSFET with a structure typical for an advanced submicron VLSI technology [65]. The boundary conditions associated with eqs. (1)–(6) are such that the MOSFET is in saturation, a high-current state that is difficult for the plug-in iteration. The underlying spatial grid is a typical one for the numerical simulation of a single two-dimensional device. In table 1 we have summarized the important data as obtained for solving a static problem using different iteration schemes. The nomenclature used in the table is simple: "coupled" and "plug-in" refer to the nonlinear iteration schemes and, in some of the cases, ABF has been applied as an outer conditioner. The solution of the linear system of equations has been achieved by either sparse direct or iterative methods with an ILU preconditioner. For the coupled case, the entry in the "Nonlinear iterations" column is just the number of coupled Newton iterations; for the plug-in case, the entry is just the number of plug-in iterations. The "Normalized run time" column shows CPU times normalized by the time required for the coupled Newton algorithm with sparse direct methods – remember that the timings are subject to substantial measurement error and the detail of the implementation and machine architecture.

The results of table 1 can be analyzed in terms of the number of nonlinear and linear iterations as well as the normalized run time. Given the relatively small number of grid points, the plug-in methods are attractive even if sparse direct methods are used. However, for a more demanding problem like a bipolar transistor in the high-injection regime or CMOS latch-up, the coupled direct algorithm can outperform the plug-in direct algorithm since the convergence behavior of plug-in deteriorates. The coupled and plug-in procedures based on iterative methods for the linear systems are obviously attractive. The coupled-ABF block-iterative method is attractive, even for the case of a direct solution of the linearized blocks in the Gauss–Seidel inner iteration.

An increase in the number of grid points shifts the advantage more in direction of plug-in and coupled-ABF block-iterative methods (table 2). In this case, the ABF approach with inner BICG or CGS iterations is the clear winner. The CGS algorithm takes fewer iterations than BICG in comparable cases in both tables 1 and 2, but there is little observed difference in CPU time for our particular implementation and computer.

The third example (table 3) involves a Hall sensor under conditions of high-current flow [66]. This sensor is basically a uniformly 10^{16}-doped structure in a magnetic field of one Tesla biased in a high-current regime. The addition of magnetic field terms to the semiconductor equations adds small changes [13,33]. The uniformity of the device makes the outer ABF conditioning particularly effective; the matrices arising in the

Table 1
Performance of various algorithms for a 0.5 μm MOSFET in saturation. The underlying two-dimensional grid has 1163 vertices. All computer times are from an Alliant FX/80 with 5 computational elements, 256 Kb caches, and 112 Mb of real memory run under Concentrix version 4 with the 4.0.28 FORTRAN compiler. The computations were done in double-precision arithmetic

Nonlinear algorithm	Outer conditioner	Linear algorithm	Inner conditioner	Nonlinear iterations	Linear iterations	Normalized run time
coupled		direct		14	14	1.00
plug-in		direct		42	217	0.95
plug-in		BICG	ILU	42	6145	0.58
plug-in		CGS	ILU	42	4775	0.55
coupled	ABF	direct		41	123	0.69
coupled	ABF	BICG	ILU	41	4133	0.53
coupled	ABF	CGS	ILU	41	3113	0.49

Table 2
Simulations of the 0.5 μm MOSFET on a finer grid, consisting of 2765 vertices

Nonlinear algorithm	Outer conditioner	Linear algorithm	Inner conditioner	Nonlinear iterations	Linear iterations	Normalized run time
coupled		direct		12	12	1.00
plug-in		direct		41	200	0.83
plug-in		BICG	ILU	41	8387	0.42
plug-in		CGS	ILU	41	5698	0.37
coupled	ABF	direct		26	78	0.41
coupled	ABF	BICG	ILU	26	3346	0.26
coupled	ABF	CGS	ILU	26	1828	0.24

coupled-direct algorithm are poorly conditioned, which may be seen in the relatively large number of nonlinear iterations required. The plug-in algorithms are not competitive with the coupled algorithms for this problem. As an additional result, we have included figures for the plug-in case using the new discretization for the continuity equations introduced in ref. [33]. This discretization, while offering many advantages such as a less severe angle dependence (section 2.2), can produce matrices with considerably higher condition numbers than for the conventional Scharfetter–Gummel case. In this example, the new discretization only exhibits a higher cost per iteration as compared to the Scharfetter–Gummel discretization.

Our general experience is that CGS (section 4.2) is as robust as BICG and CGS usually takes fewer iterations so it is the method of choice. However, BICG is only marginally slower than CGS in overall CPU time on the Alliant FX/80 we used for our experiments. ORTHOMIN(k) and GMRES(k) seem to be less robust for these difficult problems and may require more storage than BICG, but further experimentation is needed. The non-tensor-product grids and other aspects of our simulations pose nontrivial problems. We will defer a discussion of Krylov-subspace algorithms applied to the linearized equations obtained from the coupled method, without ABF, to a later paper [62].

From the results we have seen through the examples above, several conclusions can be drawn that shed light on the performance of different iteration schemes used in device simulation. It seems clear that the use of physically-based, outer-conditioning schemes such as ABF offer a clear advantage over the conventional coupled or plug-in iterative methods. We estimate that this advantage will be even larger in simulations with transient conditions or problems involving three spatial dimensions. For the linear equations, the

Table 3
Simulations of a magnetosensor on a grid consisting of 1173 vertices. The results marked with * use the new discretization [33], rather than Scharfetter–Gummel

Nonlinear algorithm	Outer conditioner	Linear algorithm	Inner conditioner	Nonlinear iterations	Linear iterations	Normalized run time
coupled		direct		20	20	1.00
plug-in		direct		104	515	1.92
plug-in		BICG	ILU	104	4279	0.61
plug-in		BICG	ILU	101	4283	0.89 *
plug-in		CGS	ILU	104	2583	0.68
plug-in		CGS	ILU	101	2455	0.87 *
coupled	ABF	direct		10	30	0.14
coupled	ABF	BICG	ILU	10	693	0.09
coupled	ABF	CGS	ILU	10	413	0.09

Normalized run time
1.00
0.83
0.42
0.37
0.41
0.26
0.24

ORTHOMIN(k)
robust for these
iire more storage
ntation is needed.
1 other aspects of
roblems. We will
space algorithms
ns obtained from
, to a later paper

een through the
ns can be drawn
ince of different
e simulation. It
physically-based,
as ABF offer a
ional coupled or
stimate that this
simulations with
involving three
ir equations, the

v discretization [33],

Normalized run time
1.00
1.92
0.61
0.89 *
0.68
0.87 *
0.14
0.09
0.09

proper use of iterative algorithms, such as BICG and CGS, seems to be the only viable alternative to sparse direct methods since iterative methods consume less CPU time and storage.

Acknowledgements

We gratefully acknowledge valuable discussions with Mark Pinto and Josef Bürgler. Randy Bank's work was supported in part by the Office of Naval Research under contract N00014-82K-0197.

References

[1] W. Van Roosbroeck, Bell System Tech. J. 29 (1950) 560.

[2] M.S. Mock, Analysis of Mathematical Models of Semiconductor Devices (Boole Press, Dublin, 1983).

[3] S. Selberherr, Analysis and Simulation of Semiconductor Devices (Springer-Verlag, Vienna, 1984).

[4] P.A. Markowich, The Stationary Semiconductor Equations (Springer-Verlag, Vienna, 1986).

[5] H.C. Elman, PhD thesis, Res. Rep. 229, Dept. of Computer Science, Yale Univ., New Haven (1982).

[6] G.H. Golub and C.F. Van Loan, Matrix Computations (Johns Hopkins, Baltimore, 1983).

[7] Y. Saad and M.H. Schultz, Math. Comput. 44 (1985) 417.

[8] R.E. Bank, W.M. Coughran, Jr., W. Fichtner, E.H. Grosse, D.J. Rose and R.K. Smith, IEEE Trans. CAD CAD-4 (1985) 436.

[9] J.J. Dongarra and E.H. Grosse, Commun. ACM, 30 (1987) 403.

[10] R.E. Bank, W.M. Coughran, Jr., W. Fichtner, D.J. Rose and R.K. Smith, in: Process and Device Simulation, ed. W.L. Engl (North-Holland, Amsterdam, 1986) p. 229.

[11] W.M. Coughran, Jr., M.R. Pinto and R.K. Smith, IEEE Trans. CAD-7 (1988) 307.

[12] W.M. Coughran, Jr., M.R. Pinto and R.K. Smith, J. Comput. Appl. Math., in press.

[13] H.P. Baltes and R.S. Popovic, Proc. IEEE 74 (1986) 1107.

[14] A. DeMari, Solid-State Electron. 11 (1968) 33.

[15] A. DeMari, Solid-State Electron. 11 (1968) 1021.

[16] S.M. Sze, Physics of Semiconductor Devices, 2nd ed. (Wiley-Interscience, New York, 1981).

[17] K. Hess, Advanced Theory of Semiconductor Devices (Prentice-Hall, Englewood Cliffs, 1988).

[18] R.E. Bank, D.J. Rose and W. Fichtner, IEEE Trans. Electr. Dev. ED-30 (1983) 1031.

[19] W. Fichtner, D.J. Rose and R.E. Bank, IEEE Trans. Electr. Dev. ED-30 (1983) 1018.

[20] T. Kerkhoven, PhD thesis, Res. Rep. 429, Dept. of Computer Science, Yale Univ., New Haven (1985).

[21] E.M. Buturla and P.E. Cottrell, Solid-State Electronics 23 (1980) 331.

[22] C.S. Rafferty, M.R. Pinto and R.W. Dutton, IEEE Trans. Electr. Dev. ED-32 (1985) 2018.

[23] R.E. Bank and A.H. Sherman, Computing 26 (1981) 91.

[24] R.E. Bank, tech. rep., Univ. of Calif., San Diego, Dept. of Math. (1988).

[25] G. Strang and G.J. Fix, An Analysis of the Finite Element Method (Prentice-Hall, Englewood Cliffs, 1973).

[26] M.R. Pinto, C.S. Rafferty and R.W. Dutton, tech. rep., Stanford Electronics Laboratory, Dept. of Electrical Engineering, Stanford Univ., Palo Alto (1984).

[27] M.R. Pinto, C.S. Rafferty, H.R. Yeager and R.W. Dutton, tech. rep., Stanford Electronics Laboratory, Dept. of Electrical Engineering, Stanford Univ., Palo Alto (1985).

[28] R.S. Varga, Matrix Iterative Analysis (Prentice-Hall, Englewood Cliffs, 1962).

[29] D. Scharfetter and H.K. Gummel, IEEE Trans. Electr. Dev. ED-16 (1969) 64.

[30] C.H. Price, PhD thesis, Dept. of Electrical Eng., Stanford Univ., Palo Alto (1982).

[31] B.S. Baker, E.H. Grosse and C.S. Rafferty, J. Disc. Comput. Geom. 3 (1988) 147.

[32] M.S. Mock, in: NASECODE IV, ed. J.J.H. Miller (1985) p. 36.

[33] J.F. Bürgler, R.E. Bank, W. Fichtner and R.K. Smith, IEEE Trans. CAD, in press.

[34] C. Johnson, Numerical Solutions of Partial Differential Equations by the Finite Element Method (Cambridge Univ. Press, Cambridge, 1987).

[35] R.E. Bank and D.J. Rose, Num. Math. 37 (1981) 279.

[36] W.M. Coughran, Jr., E.H. Grosse and D.J. Rose, IEEE Trans. Electr. Dev. ED-30 (1983) 1207.

[37] W.M. Coughran, Jr., E.H. Grosse and D.J. Rose, in: VLSI CAD Tools and Applications, eds. W. Fichtner and M. Morf (Kluwer Academic Publishers, Boston, 1987) p. 105.

[38] J.M. Ortega and W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables (Academic Press, New York, 1970).

[39] H.K. Gummel, IEEE Trans. Electr. Dev. ED-11 (1964) 455.

[40] S.C. Eisenstat, M.H. Schultz and A.H. Sherman, in: Sparse Matrix Computations, eds. J.R. Bunch and D.J. Rose (Academic Press, New York, 1976) p. 263.

[41] S.C. Eisenstat, M.C. Gursky, M.H. Schultz and A.H. Sherman, Intern. J. Num. Meth. Eng. 18 (1982) 1145.

[42] J.A. Meijerink and H.A. van der Vorst, Math. Comput. 31 (1977) 148.

[43] O. Axelsson and N. Munksgaard, Intern. J. Num. Math. Eng. 14 (1978) 1001.

[44] D.S. Kershaw, J. Comput. Phys. 26 (1978) 43.

[45] O. Axelsson, BIT 25 (1985) 166.

[46] P.K.W. Vinsome, in: Proc. Fourth Symp. on Reservoir Simulation (Society of Petroleum Engineers, Houston, 1976) p. 149.

[47] A.V. Aho, J.E. Hopcroft and J.D. Ullman, Data Structures and Algorithms (Addison-Wesley, Reading, MA, 1983).

[48] R. Schreiber and W. Tang, in: Proc. Symp. CYBER 205 Appl. (Colorado Springs), Control Data Corp. (1982).

[49] E.L. Poole and J.M. Ortega, SIAM J. Num. Anal. 24 (1987) 1394.

[50] C.C. Ashcraft and R.G. Grimes, SIAM J. Sci. Stat. Comput. 9 (1988) 122.

[51] R.G. Melhem and K.V.S. Ramarao, ACM Trans. Math. Software 14 (1988) 117.

[52] H.C. Elman and E. Agrón, Comput. Phys. Commun. 53 (1989) 253.

[53] R. Fletcher, in: Proc. 1974 Dundee Biennial Conf. on Numerical Analysis (Springer-Verlag, Berlin, 1975) p. 73.

[54] Y. Saad, SIAM J. Num. Anal. 19 (1982) 470.

[55] Y. Saad, Math. Comput. 37 (1981) 105.

[56] Y. Saad, SIAM J. Sci. Stat. Comput. 5 (1984) 203.

[57] Y. Saad and M.H. Schultz, SIAM J. Sci. Stat. Comput. 7 (1986) 856.

[58] P. Sonneveld, Rep. 84-16, Dept. of Math. and Informatics, Delft Univ. (1984).

[59] C. den Heijer, in: Proc. Intern. Conf. on Simulation of Semiconductor Devices and Processes, Swansea, eds. K. Board and D.R.J. Owen (Pineridge Press, Swansea, UK, 1984) p. 267.

[60] T. Kerkhoven and Y. Saad, tech. rep., Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign (1987).

[61] C. Lanczos, J. Res. Nat. Bur. Standards 45 (1950) 255.

[62] R.E. Bank, W.M. Coughran, Jr. and R.K. Smith, The alternate block factorization procedure for systems of partial differential equations, tech. rep., in preparation.

[63] S.C. Eisenstat, SIAM J. Sci. Stat. Comput. 2 (1981) 1.

[64] T. Kerkhoven, SIAM J. Sci. Stat. Comput. 9 (1988) 48.

[65] W. Fichtner, in: IEDM Tech. Dig. (IEEE, Long Beach, 1982) p. 638.

[66] H.P. Baltes, L. Andor, A. Nathan and H.G. Schmidt-Weinmar, IEEE Trans. Electr. Dev. ED-31 (1984) 996.

Computer Physics Com
North-Holland, Amste

RECENT R

John G. LEV
Scientific Compu

Received 21 Sep

We present a
linear equations
PDE's arising f
of a powerful
combination of
investigation of
is to the precon

1. Iterative algori
flow problems

The design of
solution of nonlin
lems involving co
tries. The config
properties of the
local grid refinem
varying length sca
with the efficienci
ment, the discretiz
ber of grid points
direct methods in
algebraic systems.
ities in the proble
PDE's have defea
schemes, causing
to fail completely

The need for
solving such prob
an innovative con
for large sparse s
robust iterative
conjunction with
Each preconditio
of the entire sys
direct solver, sel

0010-4655/89/$0
(North-Holland