G. D. Hachtel
M. H. Mack
R. R. O'Brien
B. Speelpenning

Semiconductor Analysis Using Finite Elements—Part I: Computational Aspects

We describe the computer implementation of SAFE, a general-purpose program for finite-element analysis of systems of nonlinear, nonvariational PDE. The form and nonlinearity of the PDE, as well as the domain, parameters, and boundary conditions of the problem, are user-specified. We discuss the problems of numerical integration in space and in time and describe the results of applying sparse-matrix methods to some practical problems. A method is given for unified treatment of nonstandard finite elements such as bicubic-spline and "current-continuous" elements. We give an assessment of the potential impact and reliability of general-purpose nonlinear finite-element programs. We conclude from out data that as the number of elements becomes large, sparse-matrix solutions dominate the overall operation count, even though the methods used are near-optimal for problems of moderate size.

1. Introduction

The last decade has been marked by significant advances in two-dimensional device simulation. Recent literature [1–11] shows that interest in this field continues apace with the inexorable scaling down of device sizes associated with the development of VLSI and VHSIC.

We describe in this paper the computer implementation of SAFE, a general-purpose program for Semiconductor-like Applications of Finite Elements, [1, 12]. Our treatment is analogous to George's [13] treatment of finite-element analysis of linear PDE (partial differential equations) deriving from a variational principle. Our program is novel, however, in that it is general-purpose and solves (simultaneously) *systems* of *nonlinear* PDE which are *not* required to derive from a variational principle. We emphasize questions of implementation and efficiency which are not specifically problem-dependent but are applicable to a specific problem class, *i.e.*, semiconductor device analysis. Specific case studies of numerical solutions for this problem class are described in a companion paper [1]. There is a voluminous literature on finite-difference (*cf.* [14] and [15] and their references) and finite-element [16–18] approaches to this problem class, and we describe in the sequel our reasons for believing SAFE to be comparatively efficient, versatile, and reliable *vis-à-vis* alternative applicable programs.

The SAFE program is "general-purpose" in the sense that it is formulated to solve *any* set of equations of the form

$$\nabla \cdot {}^{i}\mathbf{F}(\mathbf{u}, \nabla\mathbf{u}) - {}^{i}c(\mathbf{u}, \dot{\mathbf{u}}) = 0, \qquad i = 1, 2, \cdots, NSC, \tag{1a}$$

where

$$\mathbf{u} = ({}^{1}u, {}^{2}u, \cdots, {}^{NSC}u), \tag{1b}$$

and the Cartesian vector ${}^{i}\mathbf{F}$ stands for the flux of some generalized flow, the scalar function ${}^{i}c$ for the net rate of particle generation in the flow, and NSC is the number of equations.

Thus the generality of the SAFE program may be assessed as follows. Typical semiconductor device applications may be addressed using default specifications of $\mathbf{F}$ and $\mathbf{c}$. This task may or may not be difficult for a given problem. However, the key numerical portions of the program, such as the sparse matrix, nonlinear itera-

tion, and time integration algorithms, are general-purpose and invisible to the user.

The reader should be warned in advance about the notation we have found necessary for proper presentation of the material. The problem is that we must deal with vectors in several different spaces. First note that $\mathbf{F}$ is a vector in physical Cartesian space. We denote the x and y components of $\mathbf{F}$ by F_x and F_y. However, $\mathbf{F}$ is also considered to be a vector in the vector space $\mathscr{R}^{NSC}$. Each component ${}^i\mathbf{F}$ in this space is itself a vector in Cartesian space. Pre-superscripts are used only for this purpose. The "$\cdot$" notation in (1a) (and throughout the sequel) thus denotes inner product over the Cartesian space. That is,

$$\nabla \cdot {}^i\mathbf{F} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right) \cdot \begin{pmatrix} {}^iF_x \\ {}^iF_y \end{pmatrix} = \frac{\partial {}^iF_x}{\partial x} + \frac{\partial {}^iF_y}{\partial y}.$$

Similarly $\mathbf{u}$ is a vector in $\mathscr{R}^{NSC}$ whose components ${}^iu$ are scalars in Cartesian space. We shall also use the notation

$${}^iu_x \equiv (\nabla {}^iu)_x \equiv \nabla_x {}^iu \equiv \frac{\partial {}^iu}{\partial x}.$$

Wherever possible, we shall omit the presuperscript.

SAFE makes no restrictions on the algebraic form of the dependence of $\mathbf{F}$ and $\mathbf{c}$ on $\mathbf{u}$, $\nabla\mathbf{u}$, and $\dot{\mathbf{u}} \doteq d\mathbf{u}/dt$, and SAFE does not require the existence of a variational principle for the equations solved. The form of Eq. (1) is characteristic of the transport and Poisson's equations customarily used in modeling semiconductor devices and integrated circuits [14–18] and appears as well in nuclear reactor [19], air and water pollution [20, 21], and other flow problems. SAFE solves the nonlinear PDE *simultaneously* using a Galerkin's principle [22, 23]. The resulting nonlinear algebraic equations are solved by a modified Newton's method. The resulting linear system is solved with a sparse-matrix package in which minimal degree [24] and nested dissection [25, 26] ordering heuristics are options. Solutions are obtained by either the variability-typed, compiled code method [27], or the indexed fill method of the SL-MATH package [28].

The *simultaneous* solution of the nonlinear equations via *direct* methods ensures *fully implicit* solution of (1) with a *quadratically* convergent Newton's method. These properties allow us to choose relatively gross finite-element approximations with only engineering accuracy rather than having a higher accuracy enforced by numerical stability limitations.

The approach used for time integration also reflects this fully implicit philosophy. The A-stable "Backward Euler" method [29] is therefore employed for time discretion. Standard methods [29] are also employed for variable prediction, truncation error estimation, and time step control. First-order integration is employed, since our concern is for fast, approximate, stable time integration, rather than for high accuracy.

Like the linear finite-element code of George [13] and Speelpenning [30, 31], SAFE permits the user to specify the type of finite-element applications used in the approximation. However, SAFE lacks some of the desirable features of these other codes: for example, curved boundary elements and heterogeneous element populations.

So far as we know, reports on codes by Buturla and Cottrell [16], Wilson and Tchon [17], and Barnes and Lomax [18] comprised the first literature on the application of finite-element methods to semiconductor applications. Buturla and Cottrell [5, 6] and Lomax *et al.* [2–4] have followed this up with significant studies of 2-D device analysis, and Buturla *et al.* [6–7] of 3-D analysis, emphasizing convergence of nonlinear iterations, conservation properties, etc.

Wilson and Tchon employ the interesting "2–4" bilinear finite-element approximation in solving only the semiconductor Poisson's equation. Barnes and Lomax solve the semiconductor Poisson's and electron continuity equations using a bicubic spline basis. This basis may be regarded as a smooth and only slightly more computationally expensive extension of the piecewise linear (*e.g.*, "1–3" finite elements) approximation reported by Buturla and Cottrell. It is to be noted that the bicubic splines, like a bi-orthogonal basis [22, 23], require an essentially rectangular grid, as in most finite-difference methods. It is not yet clear whether the added smoothness of the bicubic splines *versus* the piecewise linear basis is adequate compensation for the extra computational expense, the loss of the capability of arbitrary local refinement, and the introduction of overshoot problems. We shall include some results on this interesting open question. However, emphasis in the present paper is on the general-purpose algorithms and data structure aspects of the SAFE program which allow it, in principle and with appropriate user input specifications, to perform all the computations of [2–14, 16–18].

We begin our report in Section 2 with a description of the sparse Galerkin equations for systems of nonlinear PDE. Section 3 describes our treatment of numerical integration problems which are concomitant with *nonlinear* finite-element problems. In this section we describe our approach to transient analysis, based on Backward Euler time integration. In Section 4 we treat the mechanism which permits user specification of the number and form of the PDE [Eq. (1)], *i.e.*, the definition of ${}^i\mathbf{F}(\mathbf{u}, \nabla\mathbf{u})$ and ${}^ic(\mathbf{u}, \dot{\mathbf{u}}), i = 1, 2, \cdots, NSC$.

Section 5 describes generation of the finite-element equations and contains no new results but is necessary for completeness in Sections 6 and 7, which include our main results. The reader familiar with George's Dissertation [13] may skip this section without appreciable loss. Section 6 describes an extension of George's method of generating the finite-element equations which permits a unified treatment of elements with unusual continuity properties, *e.g.*, tensor product spaces such as bicubic splines and current-continuous elements. Section 7 describes the application and performance of the SAFE sparse-matrix package. Treated are the effects of grid size and shape, finite-element type, boundary condition type, and algorithms used for spare-matrix ordering and solution.

We conclude in Section 8 with an assessment of the overall possibilities for general-purpose nonlinear finite-element programs. Included is a discussion of the extensions of the SAFE algorithms and data structure which seem both necessary and straightforwardly possible on the basis of our computing experience to date.

2. Sparse Galerkin equations for nonlinear, nonvariational systems of PDE

The SAFE program is intended for the class (1) of nonlinear PDE which do not necessarily derive from a variational principle. Therefore we employ an approximation

$$\mathbf{u}(x, y) = \alpha^T\phi = \phi^T\alpha = \sum_{n=1}^{NDOF} \alpha_n\phi_n(x, y), \tag{2}$$

in terms of basis functions, ϕ_n, of an *NDOF*-dimensional linear space, where *NDOF* is the number of degrees of freedom. The generalized coordinates, α_n, are determined by the Galerkin conditions

$$\int_\Omega \phi_n\{\mathbf{c}(\mathbf{u}, \dot{\mathbf{u}}, x, y) - \nabla \cdot \mathbf{F}(\mathbf{u}, \nabla\mathbf{u})\}d\Omega,$$
$$n = 1, 2, \cdots, NDOF\ , \tag{3}$$

where Ω stands for the domain over which (1) is to be satisfied.

The program permits boundary conditions of the form

$$D(x, y)\mathbf{u}(x, y) + N(x, y)F_N[\mathbf{u}(x, y), \nabla\mathbf{u}(x, y)] = U(x, y),$$
$$x, y \in \partial\Omega\ , \tag{4a}$$

where $\partial\Omega$ stands for the boundary of Ω, F_N is the flux component normal to the boundary, and D, N, and U are problem-specific. $\partial\Omega$ is to be regarded as the union

$$\partial\Omega = \partial\Omega_N \cup \partial\Omega_D\ , \tag{4b}$$

where $\partial\Omega_N$ is that so-called "natural" portion of the boundary whereon

$$\mathbf{u}(x, y) = D(x, y) = 0, \qquad x, y \in \partial\Omega_N\ . \tag{4c}$$

Then, taking the usual integration by parts to remove one level of spatial derivatives, we arrive at

$$r_n(\alpha) = \int_\Omega \{\nabla\phi_n \cdot \mathbf{F}(\mathbf{u}, \nabla\mathbf{u}) + \phi_n\mathbf{c}(\mathbf{u}, \dot{\mathbf{u}}, x, y)\}d\Omega$$
$$- \int_{\partial\Omega_D} \phi_n F_N d\partial\Omega = 0,\ n = 1, 2, \cdots, NDOF. \tag{5a}$$

Eq. (5a) gives *NDOF* equations

$$\mathbf{r}(\alpha) = 0 \tag{5b}$$

in the *NDOF* components of the unknown vector α. Further, we decompose the domain Ω into finite elements Ω^l, *i.e.*,

$$\Omega \doteq \bigcup_{l=1}^{L} \Omega^l\ , \tag{6}$$

and require that each of the $\phi_n(x, y)$ be piecewise polynomials, *i.e.*, different polynomials in Ω^l, $l = 1, 2, \cdots, L$. Thus we can express the conditions (5) as a summation over the finite elements, *i.e.*,

$$r_n = \sum_{l=1}^{L} \left\{ \int_{\Omega^l} (\nabla\phi_n \cdot \mathbf{F} + \phi_n\mathbf{c})d\Omega \right.$$
$$\left. - \int_{\partial\Omega_D^l} \phi_n\mathbf{F} \cdot d\partial\Omega \right\} = 0\ , \tag{7}$$

which allows us to treat one finite element at a time. [Note that $\partial\Omega_D^l \neq 0$ only for elements with an edge on the non-natural portion of the boundary. In some cases, the functions ϕ_n may be defined so that the boundary integral in (7) vanishes except for functions whose generalized coordinates are specified *a priori* so as to satisfy (4a). For such functions (4a) replaces (7), so the boundary integral never appears.]

The Galerkin eqs. (5b) are nonlinear in α if $\mathbf{F}$ or $\mathbf{c}$ is nonlinear in $\mathbf{u}$, $\nabla\mathbf{u}$, and are solved via a Newton iteration in which the basic step is

$$\frac{\partial\mathbf{r}}{\partial\alpha}(\alpha)\Delta\alpha = -\mathbf{r}(\alpha)\ . \tag{8}$$

We now treat the mechanism by which the SAFE program permits convenient user specification of $\mathbf{F}$ and $\mathbf{c}$, and, therefore, the residual vector $\mathbf{r}(\alpha)$ and its Jacobian $\partial\mathbf{r}/\partial\alpha$.

The Jacobian $\partial\mathbf{r}/\partial\alpha$ is a sparse matrix because some of the $\phi_n(x, y)$ are identically zero for $x, y \in \Omega^l$. We can represent this large sparse matrix as a summation of

small, usually full, matrices by introducing the incidence relation

$$\boldsymbol{\phi}^l = \mathbf{A}^l \boldsymbol{\phi} \,, \tag{9a}$$

where $\boldsymbol{\phi}$ is the vector of global, piecewise-polynomial basis functions and $\boldsymbol{\phi}^l$ is a vector of "local" basis functions which are the restriction to Ω^l of those ϕ_n which are nonzero in Ω^l. $\mathbf{A}^l$ is a matrix of 0s and 1s with exactly one nonzero entry in each row, and the ϕ^l_m, $m = 1, 2, \cdots$, $|\boldsymbol{\phi}^l|$ are polynomials. Thus for $x, y \in \Omega^l$, Eq. (2) becomes

$$\mathbf{u} = \boldsymbol{\alpha}^T \boldsymbol{\phi} = \boldsymbol{\alpha}^{lT} \boldsymbol{\phi}^l, \ \boldsymbol{\alpha}^l \doteq \mathbf{A}^l \boldsymbol{\alpha}. \tag{9b}$$

Similarly, we may define a local residual vector

$$\mathbf{r}^l \doteq \int_{\Omega^l} (\nabla \boldsymbol{\phi}^l \cdot \mathbf{F} + \boldsymbol{\phi}^l \mathbf{c}) d\Omega - \int_{\partial\Omega^l_D} \boldsymbol{\phi}^l \mathbf{F} \cdot d\partial\Omega \,, \tag{10a}$$

whose Jacobian,

$$\mathbf{R}^l \doteq \frac{\partial \mathbf{r}^l}{\partial \boldsymbol{\alpha}^l} = \int_{\Omega^l} \left(\nabla \boldsymbol{\phi}^l \cdot \frac{\partial \mathbf{F}}{\partial \boldsymbol{\alpha}^l} + \boldsymbol{\phi}^l \frac{\partial \mathbf{c}}{\partial \boldsymbol{\alpha}^l} \right) d\Omega$$

$$- \int_{\partial\Omega^l_D} \boldsymbol{\phi}^l \frac{\partial \mathbf{F}}{\partial \boldsymbol{\alpha}^l} \cdot d\partial\Omega \,, \tag{10b}$$

is a full matrix. It can be shown that

$$\mathbf{r}(\boldsymbol{\alpha}) = \sum_{l=1}^{L} \mathbf{A}^{lT} \mathbf{r}^l(\boldsymbol{\alpha}^l), \ \frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}} = \sum_{l=1}^{L} \mathbf{A}^{lT} \frac{\partial \mathbf{r}^l}{\partial \boldsymbol{\alpha}^l} \mathbf{A}^l \,. \tag{10c}$$

Although we have claimed to be solving $NSC \geq 1$ *simultaneous* PDE, the notation of Eqs. (1)–(10) has been established for the case $NSC = 1$. The extension to $NSC > 1$ involves, simply, the implicit understanding that each vector component α_n, ϕ_n, r_n, α^l_m, ϕ^l_m, or $\mathbf{r}^l$ stands for a set of NSC members, *i.e.*,

$$\alpha_n = \text{col}\,({}^1\alpha_n, {}^2\alpha_n, \cdots, {}^{NSC}\alpha_n) \,,$$

$$\alpha^l_m = \text{col}\,({}^1\alpha^l_m, {}^2\alpha^l_m, \cdots, {}^{NSC}\alpha^l_m) \,, \tag{11a}$$

and similarly for $\boldsymbol{\phi}$, $\mathbf{r}$, $\boldsymbol{\phi}^l$ and $\mathbf{r}^l$. With this in mind Eqs. (1)–(10) still make sense as NSC PDE:

$$\nabla \cdot \{{}^1\mathbf{F}(\mathbf{u}, \nabla\mathbf{u})\} = {}^1\mathbf{c}(\mathbf{u}, \dot{\mathbf{u}}, x, y) \,,$$

$$\nabla \cdot \{{}^2\mathbf{F}(\mathbf{u}, \nabla\mathbf{u})\} = {}^2\mathbf{c}(\mathbf{u}, \dot{\mathbf{u}}, x, y) \,,$$

$$\nabla \cdot \{{}^{NSC}\mathbf{F}(u, \nabla\mathbf{u})\} = {}^{NSC}\mathbf{c}(\mathbf{u}, \dot{\mathbf{u}}, x, y) \,, \tag{11b}$$

in the NSC unknown functions

$${}^1u(x, y), {}^2u(x, y), \cdots, {}^{NSC}u(x, y) \,. \tag{11c}$$

Note that it follows from (11a) that the approximation (2) represents NSC distinct approximations in NSC distinct $NDOF$-dimensional spaces. To date, we *have* approximated the ${}^k u(x, y)$ in the *same* $NDOF$-dimensional linear space, but this is not basic to the algorithms or data structure of the SAFE program.

Time integration can be accomplished by setting

$$\dot{\mathbf{u}}(t_\tau) = \mathbf{F}[t_\tau, t_{\tau-1}, \mathbf{u}(t_\tau), \mathbf{u}(t_{\tau-1}), \cdots, \mathbf{u}(t_{\tau-k})] \tag{11d}$$

for a kth-order, one-step-implicit method. In SAFE we have chosen the particular case $k = 1$ (Backward Euler). In this case (11d) becomes

$$\dot{\mathbf{u}}(t_\tau) \equiv [\mathbf{u}(t_\tau) - \mathbf{u}(t_{\tau-1})]/\Delta t \,, \tag{11e}$$

$$\Delta t \equiv t_\tau - t_{\tau-1} \,. \tag{11f}$$

As we shall see in Section 4, implementation of (11e) and (11f) is quite straightforward. Note that the usual difficulties of transient analysis are avoided by virtue of the fact that the Backward Euler method is "A-stable" [29]. The only difficulty of implementation involves the control of Δt during the transient analysis. We use the same linear-predictor, Newton corrector scheme described in [29].

3. Numerical volume and surface integrals

Note that (10) requires a volume integral over each finite element Ω^l, and depending on the location of the element, a surface integral over its boundary $\partial\Omega^l$ as well. Whereas George ([13], p. 72) was able to give exact formulae for these integrals because his problem class was linear, the form of Eq. (1) requires numerical integration. The SAFE program represents integration formulae in the generalized form

$$\int_{\Omega^l} g(x, y) d\Omega \cong |\Omega^l| \sum_{i=1}^{NIP} W_i g(x_i, y_i) \,, \tag{12a}$$

where

$$|\Omega^l| \doteq \int_{\Omega^l} d\Omega \,. \tag{12b}$$

Here NIP is the number of distinct points $(x_i, y_i) \in \Omega^l$ where the integrand $g(x, y)$ is to be evaluated and added with weight W_i to the integral. In SAFE, the numerical integration formula chosen by the user (or defaulted) is modularly imbedded in a subroutine GETXYW. GETXYW performs (12b) and, given the data structure for Ω^l, determines the coordinates (x_i, y_i) of the points with weight W_i for the operative integration formulae.

For example, if Ω^l is a triangle $\triangle$ of area A, and we use a "1-point" integration formula, we have $NIP = 1$, $W_1 = 1$, and

$$\int_{\Delta} g(x, y) d\Omega \cong \mathbf{A} g(x_c, y_c) \,, \tag{13a}$$

where (x_c, y_c) is the centroid of $\triangle$. Similarly, if Ω^l is a triangle subdivided at its centroid into the union of three triangles with centroids (x_{ci}, y_{ci}), $NIP = 3$, $W_i = 1/3$, and

$$\int_{\Delta} g(x, y) d\Omega \cong \frac{\mathbf{A}}{3} \sum_{i=1}^{3} g(x_{ci}, y_{ci}) \,. \tag{13b}$$

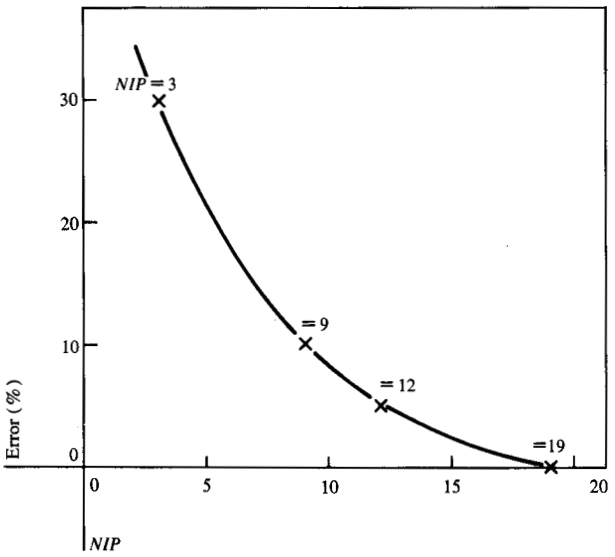


Figure 1 Percentage error in integrated transverse flux as a function of number of integration points, with the error at *NIP* = 19 defined as zero.

Figure 1 shows the dependence of an error measure of the solutions of (1) on the number of integration points (for a practical problem for which the Ω^l were triangles and the *NSC* = 2 PDE were of the "exponential-nonlinearity" type typical of semiconductor modeling problems, *cf.* [17], below. The error measure plotted in Fig. 1 is not the approximation error of (12a), but is the error incurred in computing the integrated transverse flux or "current,"

$$IX \doteq \int_{x=x_1} {}^1F_x({}^1u, {}^2u, {}^1u_x)dy ,$$

crossing the grid line $x = x_1$, which is a global functional of the $k_{\mathbf{u}(x,y)}$. The error in Fig. 1 was defined to be zero for *NIP* = 19, and the plot shows that the choice of integration formula is a crucial step in solving practical nonlinear PDE with finite-element methods.

Note from (10b) that the sparse Jacobian $\partial\mathbf{r}/\partial\boldsymbol{\alpha}$ may be expressed as the double summations

$$\frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}} = \sum_{l=1}^{L} \left\{ |\Omega^l| \sum_{i=1}^{NIP^l} W_i^l \left[\nabla \boldsymbol{\phi}^l \cdot \frac{\partial \mathbf{F}}{\partial \boldsymbol{\alpha}^l}(x_i, y_i) + \boldsymbol{\phi}^l \frac{\partial \mathbf{c}}{\partial \boldsymbol{\alpha}^l}(x_i, y_i) \right] \right.$$

$$\left. - |\partial\Omega_D^l| \sum_{j=1}^{NBIP^l} W_{bj}^l \left[\boldsymbol{\phi}^l \frac{\partial F_N}{\partial \boldsymbol{\alpha}^l}(x_j, y_j) \right] \right\} , \qquad (14)$$

where F_N stands for the component of the generalized flux vector $\mathbf{F}(\mathbf{u}, \nabla\mathbf{u})$ which is normal to the boundary $\partial\Omega^l$. Note that a separate integration formula, characterized by $(W_{bj}, x_j, y_j, j = 1, 2, \cdots, NBIP)$ is required for the boundary integral in (10b). A similar expression applies to (10a).

4. User specification of the form of F (u, ∇u) and c(u, u̇, x, y)

It may be observed from the double summation form of Eq. (14) that the particular identity of a given problem [*i.e.*, the specification of the form of (1)], is completely determined by the values of $\mathbf{F}$, $\mathbf{c}$, $\partial\mathbf{F}/\partial\boldsymbol{\alpha}^l$, and $\partial\mathbf{c}/\partial\boldsymbol{\alpha}^l$ which occur at the integration points x_i, y_i. To see how the form of (1) may be user-specified, we note that if $\mathbf{c}$ is independent of $\dot{\mathbf{u}}$,

$$\frac{\partial \mathbf{c}}{\partial \boldsymbol{\alpha}^l} = \frac{\partial \mathbf{c}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \boldsymbol{\alpha}^l} = \frac{\partial \mathbf{c}}{\partial \mathbf{u}} \boldsymbol{\phi}^l(x, y) . \qquad (15a)$$

A similar result obtains if $\mathbf{c}$ depends on $\dot{\mathbf{u}}$. Similarly,

$$\nabla\boldsymbol{\phi} \cdot \frac{\partial \mathbf{F}}{\partial \boldsymbol{\alpha}^l}(\mathbf{u}, \nabla\mathbf{u}) = \phi_x \frac{\partial F_x}{\partial \boldsymbol{\alpha}^l} + \phi_y \frac{\partial F_y}{\partial \boldsymbol{\alpha}^l} , \qquad (15b)$$

where $\partial F_x/\partial\boldsymbol{\alpha}^l$ and $\partial F_y/\partial\boldsymbol{\alpha}^l$ are obtained by the chain rule,

$$\frac{\partial F_x}{\partial \boldsymbol{\alpha}^l} = \frac{\partial F_x}{\partial \mathbf{u}} \boldsymbol{\phi}^l + \frac{\partial F_x}{\partial u_x} \phi_x^l + \frac{\partial F_x}{\partial u_y} \phi_y^l . \qquad (15c)$$

Consideration of the elementary expressions (15) will confirm to the reader that specification of the form of Eq. (1) for a user's chosen problem may be accomplished by user provision of subroutines

GETC (X, Y, U, C, DCDU) , (16a)

GETF (X, Y, U, UX, FX, DFXDU, DFXFUX, DFXDUY, UY, FY,

DFYDU, DFYDUX, DFYDUY) . (16b)

The data x_i, y_i, u_i, u_{xi}, and u_{yi} (the X, Y, and U) are input to the argument list of subroutines GETC and GETF and the results $\mathbf{c}$, $\partial\mathbf{c}/\partial\mathbf{u}$, F_x, $\partial F_x/\partial\mathbf{u}$, $\partial F_x/\partial u_x$, $\partial F_x/\partial u_y$, F_y, $\partial F_y/\partial\mathbf{u}$, $\partial F_y/\partial u_x$, and $\partial F_y/\partial u_y$ are returned in the remaining data fields of the argument lists. For example, if we are solving the heat equation $\nabla \cdot \nabla\mathbf{u} = \dot{\mathbf{u}}$, GETC and GETF would be of the form (in FORTRAN)

```
SUBROUTINE GETC(X, Y, U, C, DCDU, UOLD, DELT)
C = (U - UOLD)/DELT
DCDU = 1/DELT
RETURN
END
SUBROUTINE GETF(X, Y, U,
UX, FX, DFXDU, DFXDUX, DFXDUY,
UY, FY, DFYDU, DFYDUX, DFYDUY)
FX = UX
DFXDUX = 1
DFXDUY = 0
FY = UY
DFYDU = 0
DFYDUX = 0
DFYDUY = 1
RETURN
END.
```
(16c)

In this way, surprisingly simple user-supplied FORTRAN subroutines of the form (16c) suffice to specify even complicated nonlinear PDE of the form (1).

It is to be noted that the arguments DCDU and DFXDU, DFXDUX, $\cdots$ of subroutines GETC and GETF are matrices in the most general case. Thus if we have the normalized semiconductor equilibrium equations [1], where

$$\mathbf{u} = \text{col}\,(\psi, \phi_p, \phi_n),\ p = N_V e^{\phi_p - \psi},\ n = N_C e^{\psi - \phi_n}, \tag{17a}$$

$$\mathbf{c} = \begin{bmatrix} -p + n - D(x, y) \\ 0 \\ 0 \end{bmatrix},\ \mathbf{F} = \begin{bmatrix} \epsilon & \nabla\psi \\ \mu_p p & \nabla\phi_p \\ \mu_n n & \nabla\phi_n \end{bmatrix}, \tag{17b}$$

then,

$$\text{DCDU} = \begin{bmatrix} (p+n) & (-p) & (-n) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \equiv \frac{\partial \mathbf{c}}{\partial \mathbf{u}}, \tag{17c}$$

$$\text{DFXDU} = \begin{bmatrix} 0 & 0 & 0 \\ -\mu_p p \nabla\phi_p & \mu_p p \nabla\phi_p & 0 \\ \mu_n n \nabla\phi_n & 0 & -\mu_n n \nabla\phi_n \end{bmatrix} \equiv \frac{\partial F_x}{\partial \mathbf{u}}, \tag{17d}$$

$$\text{DFXDUX} = \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & \mu_p p & 0 \\ 0 & 0 & \mu_n n \end{bmatrix} \equiv \frac{\partial F_x}{\partial u_x}, \tag{17e}$$

Note that the SAFE routines which call GETC and GETF expect and are prepared to receive $NSC \times NSC$ matrices, when applicable.

5. Generation of the finite-element equations

From Eqs. (10), (14), and (15) it can be seen that once evaluation of the ϕ_m^l at the integrand evaluation points (x_i, y_i) is understood, the reader may become fully aware of how SAFE generates the finite-element equations. We do not evaluate the ϕ_m^l directly but instead determine, from the definitions of the ϕ_m^l and of the local generalized coordinates $\boldsymbol{\alpha}^l$, the coefficient vector

$$\boldsymbol{\gamma}^l = \text{col}\,(\gamma_1, \gamma_2, \cdots, \gamma_{NDOFL(l)}) \tag{18a}$$

of the polynomial representation

$$p_m^l(x, y) = \sum_{\mu=0}^{NDEG} \sum_{\nu=0}^{\mu} \gamma^l_{\frac{(\mu)(\mu+1)}{2}+\nu}\, x^{\psi X(\mu)} y^{\psi Y(\nu)} = (\boldsymbol{\gamma}^l)^T \boldsymbol{\psi}^l, \tag{18b}$$

of ϕ_m^l in Ω^l. Here $NDOFL(l)$ is the number of degrees of freedom associated with Ω^l, and $NDEG$ is the highest power of x or y in (18b) and the monomial exponent arrays ψX and ψY usually take the cumulative form (as noted below, for representation of tensor product spaces such as bicubic splines, ψX and ψY take on a slightly different form):

$$\begin{aligned} NDEG &= 0 \quad 1 \quad 2 \quad 3 \quad \cdots \\ \psi X &= (0) \quad (1\ 0) \quad (2\ 1\ 0) \quad (3\ 2\ 1\ 0) \quad \cdots \\ \psi Y &= (0) \quad (0\ 1) \quad (0\ 1\ 2) \quad (0\ 1\ 2\ 3) \quad \cdots. \end{aligned} \tag{19a}$$

Thus it can be seen that the binomial vector

$$\boldsymbol{\psi}^l = \text{col}\,[(x^{\psi X(1)} y^{\psi Y(1)}), (x^{\psi X(2)} y^{\psi Y(2)}), \cdots] \tag{19b}$$

constitutes an alternative local basis for $\mathbf{u}$ in Ω^l.

The data structure which characterizes the ϕ_m^l (and therefore determines the coefficient vector $\boldsymbol{\gamma}$) is customarily (*cf.* George [13]) referred to as the "stencil" of the finite element. In SAFE, the stencil is represented by four $NDOFL(l)$-dimensional arrays, which constitute the four rows of the INDex of Local Generalized Coordinates array $INDLGC$. Thus for $m = 1, 2, \cdots, NDOFL(l)$,

$INDLGC[1, m] \doteq$ Order of x-differentiation associated with mth degree of freedom of Ω^l,

$INDLGC[2, m] \doteq$ Order of y-differentiation of mth degree of freedom of Ω^l,

$INDLGC[3, m] \doteq$ Node of Ω^l associated with mth degree of freedom of Ω^l,

$INDLGC[4, m] \doteq$ Index at $INDLGC[3, m]$th node of Ω^l of mth degree of freedom of Ω^l.

The mth degree of freedom of Ω^l is thought to reside at a particular node or nodal coordinate point (x_m, y_m), $m = 1, 2, \cdots, NDOFL(l)$. The "degree of freedom" is represented by a generalized coordinate,

$$\alpha_m^l = \frac{\partial^{\mu+\nu} u(x_m, y_m)}{\partial x^\mu \partial y^\nu} = \alpha_m^l \frac{\partial^{(\mu+\nu)} \phi_m^l}{\partial x^\mu \partial y^\nu}(x_m, y_m), \tag{20a}$$

where

$\mu = INDLGC[1, m]$,

$\nu = INDLGC[2, m]$,

$\mathbf{u} = \boldsymbol{\alpha}^{lT} \boldsymbol{\phi}^l$,

and basis function ϕ_m^l, for which

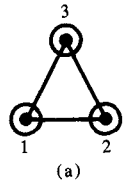
$$\frac{\partial^{(\bar{\mu}+\bar{\nu})} \phi_m^l}{\partial x^{\bar{\mu}} \partial y^{\bar{\nu}}}(x_{\bar{m}}, y_{\bar{m}}) = \begin{cases} 1 & \bar{\mu} = \mu, \bar{\nu} = \nu, \bar{m} = m, \\ 0 & \text{otherwise.} \end{cases} \tag{20b}$$

We collect the nodal coordinate points x_m, y_m into arrays

$NCX[l, INDLGC(3, m)] = x_m$, $NCY[l, INDLGC(3, m)] = y_m$.

The $INDLGC$ arrays of typical linear, quadratic, cubic, bilinear, and bicubic elements are shown in Fig. 2, where $DOFL$ stands for the index set $1, 2, \cdots, NDOFL(l)$.

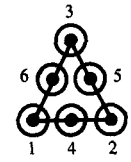
The simplest case, Fig. 2(a), shows a linear element where the three generalized coordinates represent the



(a)

DOFL 1 2 3

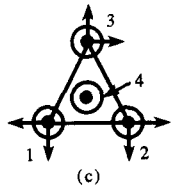
$$\text{INDLGC} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$



(b)

DOFL 1 2 3 4 5 6

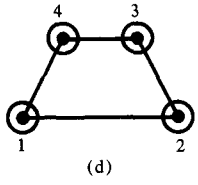
$$\text{INDLGC} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



(c)

DOFL 1 2 3 4 5 6 7 8 9 10

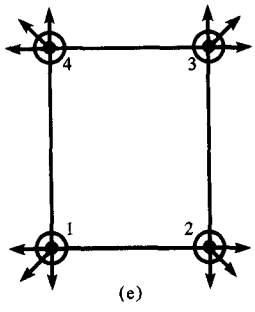
$$\text{INDLGC} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 2 & 3 & 4 & 1 & 1 & 2 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 & 2 & 3 & 2 & 3 & 2 & 3 \end{bmatrix}$$



(d)

DOFL 1 2 3 4

$$\text{INDLGC} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$



(e)

DOFL 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

$$\text{INDLGC} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 2 & 3 & 4 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 & 2 & 3 & 4 & 2 & 3 & 4 & 2 & 3 & 4 & 2 & 3 & 4 \end{bmatrix}$$

Figure 2 Stencils of typical finite elements: (a) linear; (b) quadratic; (c) cubic; (d) bilinear; and (e) bicubic.

values of **u** (circles in Fig. 2) at the three nodes (dots in Fig. 2). Figure 2(c) shows a cubic case wherein the x- and y-directed arrows represent first x- and y-derivatives of **u** at the nodes 1, 2, and 3. Figure 2(d) shows a bilinear case which illustrates that the Ω^l are not required to be triangular.

For most element types, including those of Figs. 2(a–d), it may be shown that the continuity class of the approximation (2) is determined by the properties (20) of the definitions of $\boldsymbol{\alpha}$ and $\boldsymbol{\phi}$. [The bicubic element of Fig. 2(e) is different, but we shall discuss this element in the next section.] Further, these same properties determined the γ-vector of (18) through the *NDOFL*(l) equations

$$\frac{\partial^{\mu+\nu} p_m^l(x_m, y_m)}{\partial x^\mu \partial y^\mu} = \frac{\partial^{\mu+\nu} \psi^l(x_m, y_m)^T}{\partial x^\mu \partial y^\nu} \gamma^l = \alpha_m^l, \tag{21}$$

which, when appropriately assembled, determine a ma-

trix Γ^{-T} [here we use the notation Γ^{-T} for $(\Gamma^{-1})^T$], such that

$$\boldsymbol{\alpha}^l = \Gamma^{-T}\boldsymbol{\gamma}, \text{ or } \boldsymbol{\gamma} = \Gamma^T\boldsymbol{\alpha}^l . \tag{22}$$

It may be further shown (*cf.* George [13]) that

$$\boldsymbol{\psi}^l(x, y) = \Gamma^{-1}\boldsymbol{\phi}^l(x, y), \qquad \boldsymbol{\phi}^l = \Gamma\boldsymbol{\psi}^l . \tag{23}$$

Equation (23) above represents a key step because it can now be easily shown that (2) and (10) can be written

$$\mathbf{u} = \boldsymbol{\alpha}^{lT}\boldsymbol{\phi}^l = \boldsymbol{\alpha}^{lT}\Gamma\boldsymbol{\psi}^l = \boldsymbol{\gamma}^T\boldsymbol{\psi}^l , \tag{24a}$$

$$\boldsymbol{\gamma}^l = \Gamma\left\{\int_{\Omega^l} \nabla\boldsymbol{\psi}^l \cdot \mathbf{F} + \boldsymbol{\psi}^l\mathbf{c})d\Omega - \int_{\partial\Omega^l_D} \boldsymbol{\psi}^l\mathbf{F} \cdot d\partial\Omega\right\} , \tag{24b}$$

$$\begin{aligned}\frac{\partial \mathbf{r}^l}{\partial \boldsymbol{\alpha}^l} = \Gamma\Bigg\{\int_{\partial\Omega^l}\Bigg[& \boldsymbol{\psi}^l_x\left(\frac{\partial F_x}{\partial u}\boldsymbol{\psi}^l + \frac{\partial F_x}{\partial u_x}\boldsymbol{\psi}^l_x + \frac{\partial F_x}{\partial u_y}\boldsymbol{\psi}^l_y\right) \\ & + \boldsymbol{\psi}^l_y\frac{\partial F_y}{\partial u}\boldsymbol{\psi}^l + \frac{\partial F_y}{\partial u_x}\boldsymbol{\psi}^l_x + \frac{\partial F_y}{\partial u_y}\boldsymbol{\psi}^l_y\Bigg) \\ & + \boldsymbol{\psi}^l\left(\frac{\partial c}{\partial u}\boldsymbol{\psi}^l\right)\Bigg]d\Omega \\ & - \int_{\partial\Omega^l_D}\Bigg[\boldsymbol{\psi}^l\left(\frac{\partial F_N}{\partial \mathbf{u}}\boldsymbol{\psi}^l + \frac{\partial F_N}{\partial u_x}\boldsymbol{\psi}^l{}_x \right. \\ & \left. + \frac{\partial F_N}{\partial u_y}\boldsymbol{\psi}^l{}_y\right)\Bigg]d\partial\Omega\Bigg\}\Gamma^T .\end{aligned} \tag{24c}$$

There are many terms in (24) but the point to grasp is that given Γ, it is only necessary to evaluate the binomial vector $\boldsymbol{\psi}$ and its derivatives ψ_x and ψ_y at the *NIP* points (x_i, y_i) implied by the integration formula (12a). Then, by appropriately calling the subroutines *GETC* and *GETF* of Section 4, every term in (24) can be computed as required by (12a). The only step in this procedure which depends on the element type, size, or location is the determination of the matrix Γ.

6. Mappings for associative elements—tensor product space and current-continuous cases

In Section 5 we treated the generation of the finite-element equations for elements whose generalized coordinates α^l_m were associated by Eqs. (20) with a specific generalized derivative of $\mathbf{u}(x, y)$ evaluated at a specific node of the element. We introduce in this section the notion of an "*associative*" finite element, *i.e.*, one whose generalized coordinates $\boldsymbol{\beta}^l$ are *not* associated with a specific derivative of $\mathbf{u}$ at a given node of the element. Such elements require special treatment because without (21), the SAFE mechanism of evaluating $\mathbf{r}$ and $\partial\mathbf{r}/\partial\boldsymbol{\alpha}$ through the polynomial representation (18) fails through lack of a Γ matrix. We handle this difficulty by defining a one-to-one onto mapping

$$\mathbf{m}^l = \boldsymbol{\mu}(\boldsymbol{\alpha}^l), \frac{\partial \mathbf{m}^l}{\partial \boldsymbol{\alpha}^l} = \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\alpha}} = \mathbf{M}(\boldsymbol{\alpha}^l) , \tag{25a}$$

whose inverse

$$\boldsymbol{\alpha}^l \doteq \boldsymbol{\mu}^{-1}(\mathbf{m}^l), \frac{\partial \boldsymbol{\alpha}^l}{\partial \mathbf{m}^l} = \mathbf{M}^{-1} \tag{25b}$$

can be used to map the generalized coordinates $\mathbf{m}^l$ of an associative element onto transformed coordinates $\boldsymbol{\alpha}^l$ which do satisfy property (20). Once this is accomplished, (20)–(23) may be applied, and all the terms in (24) evaluated. This yields the local residuals

$$\mathbf{r}^l(\boldsymbol{\alpha}^l) = \mathbf{r}^l[\boldsymbol{\mu}^{-1}(\mathbf{m}^l)] , \tag{26a}$$

and their Jacobians

$$\frac{\partial \mathbf{r}^l}{\partial \mathbf{m}^l} = \partial\mathbf{r}^l\partial\boldsymbol{\alpha}^l \frac{\partial \boldsymbol{\alpha}^l}{\partial \mathbf{m}^l} = \mathbf{R}^l\mathbf{M}^{-1} . \tag{26b}$$

We now give two examples of the map $\boldsymbol{\mu}(\boldsymbol{\alpha}^l)$, *i.e.*, 1) for natural bicubic splines and 2) for current-continuous elements. [The treatment applies as well to any other tensor product space, *e.g.*, biquintic splines (*cf.* [13], pp. 14, 15).]

The bicubic-spline stencil of Fig. 2(e) shows $\boldsymbol{\gamma}$ and $\boldsymbol{\psi}$ of (18) being determined from values of $\mathbf{u}$, u_x, u_y, and u_{xy} [45° arrows in Fig. 2(e)], at each corner of a (necessarily) rectangular element. The problem is that if the approximation (2) is written in terms of the so-called B-splines (*cf.* [32] or [33], p. 89), $\boldsymbol{\gamma}$ and $\boldsymbol{\psi}$ are partially determined by generalized coordinates associated with nodes $(x, y) \in \Omega^l$. This situation is indicated in Fig. 3. Suppose the approximation in terms of cubic B-splines is written

$$\mathbf{u} = \sum_{i=0}^{I}\sum_{j=0}^{J}\beta_{ij}\,BX_i(x; x_0, x_1, \cdots, x_I)BY_j(y; y_0, y_1, \cdots, y_J) , \tag{27}$$

where I and J stand for the number of x and y subdivisions of the necessarily rectangular grid. Figure 3(a) shows that $BX_i(x; x_0, x_1, \cdots, x_I)$ is associated with the ith grid line $x = x_i$ and is supported only on elements with nodes on the grid lines $x = x_{i-2}, x_{i-1}, \cdots, x_{i+2}$. Also, in elements bracketed by $x - x_{i-1}$ and x_i, $BX_{i-2}, BX_{i-1}, \cdots, BX_{i+1}$ are supported. Similar arguments apply to the y direction, and Fig. 3(b) shows that if Ω^l has nodes on $x = x_{i-1}, x_i$ and $y = y_{i-1}$ and y_i (shaded), then it is precisely the 16 generalized coordinates [circles in Fig. 3(b)],

$$\begin{aligned}\mathbf{m}^l = \text{col } \{&\beta_{i-2,j-2}, \beta_{i-1,j-2}, \cdots, \beta_{i+1,j-2}, \beta_{i-2,j-1}, \cdots, \\ &\beta_{i+1,j+1}\} ,\end{aligned} \tag{28}$$

which combine to determine $\boldsymbol{\gamma}$ and $\boldsymbol{\psi}$ of (18). The required mapping of (28) onto the bicubic stencil of Fig. 2(e) is linear in this case, and is given by

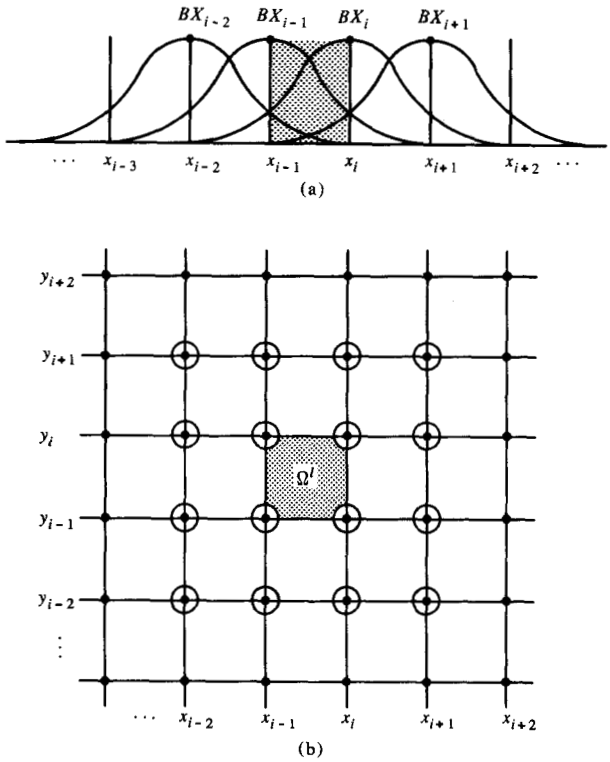


Figure 3 Bicubic B-splines and their region of support: (a) cubic B-splines associated with x-grid lines and (b) 16 basis functions supported in Ω^l.

$$\alpha_1^l = \mathbf{u}(x_{i-1}, y_{j-1}) = \sum_{i,j} \beta_{ij}\, BX(x_{i-1;})BY(y_{j-1;}) ,$$

$$\alpha_2^l = u_x(x_{i-1}, y_{j-1}) = \sum_{i,j} \beta_{ij} \frac{\partial BX}{\partial x}(x_{i-1;})BY(y_{j-1;}) ,$$

$$\alpha_3^l = u_y(x_{i-1}, y_{j-1}) = \sum_{i,j} \beta_{ij}\, BX(x_{i-1;}) \frac{\partial BY}{\partial y}(y_{j-1;}) ,$$

$$\alpha_4^l = u_{xy}(x_{i-1}, y_{j-1}) = \sum_{i,j} \beta_{ij} \frac{\partial BX}{\partial x}(x_{i-1;}) \frac{\partial BY}{\partial y}(y_{j-1;}), \quad (29)$$

where all the summations are over the range indicated by circles in Fig. 3(b). Equations (29), plus four similar equations written at the other three nodes of Ω^l, uniquely determine the inverse mapping $\boldsymbol{\alpha}^l = \boldsymbol{\mu}^{-1}(\mathbf{m}^l) = \mathbf{M}^{-1}\mathbf{m}^l$. In this case the components of $\mathbf{M}^{-1}$ are just the coefficients of β_{ij} in (29).

Note that the computations of $\mathbf{M}^{-1}$ for use in (26) require only the evaluation of the $I + 3$ (*cf.* [34]) cubic B-splines and their x-derivatives on the points x_{-1}, x_0, x_1, $\cdots$, x_{I+1} (similarly for y). Note that the extra grid lines at coordinates x_{-1}, x_{I+1}, y_{-1}, x_{J+1} are determined from the given x_i, y_i, $i = 0, 1, \cdots, I, j = 0, 1, \cdots, J$, as described in [33]. This can be done efficiently and without round-off error using the divided difference B-spline representation described in [21]. Note also that only $I + 3 + J + 3$ such calculations are required and that these may be done once only and stored for each element if desired. The specification of the bicubic-spline basis is completed by giving the binomial exponent array $\boldsymbol{\psi}\mathbf{x}$ and $\boldsymbol{\psi}\mathbf{y}$. Here the binomial form (19) is replaced by the tensor product form

$$\boldsymbol{\psi}\mathbf{x} = 0 \quad 10 \quad 210 \quad 3210 \quad 321 \quad 32 \quad 3,$$
$$\boldsymbol{\psi}\mathbf{y} = 0 \quad 01 \quad 012 \quad 0123 \quad 123 \quad 23 \quad 3. \quad (30)$$

Finally, note that if nodes are laid out on a regular rectangular grid, the bicubic-spline elements have almost the same total number, *i.e.*,

$$(I + 3) \times (J + 3)_{\text{bicubic}} \cong (I + 1) \times (J + 1)_{\text{linear,bilinear}} \quad (31)$$

of degrees of freedom as the linear [Fig. 2(a)] or bilinear [Fig. 2(d)] elements. This means that at the price of some extra sparse-matrix operations (17 nonzeros per row of $\partial\mathbf{r}/\partial\boldsymbol{\alpha}$ *versus* 7 for linear elements on a regular rectangular grid) the bicubic splines give a smooth approximation of continuity class C^2 (*cf.* [18], [34]). Of course, the linear and bilinear elements give continuity class C^0. However, it is not yet clear whether the added smoothness is worth the price paid, *i.e.*, more sparse-matrix operations, overshoot problems (*cf.* [11]), and sacrifice of the option of arbitrary local refinement of the grid.

A second "associative" element is motivated by the flux conservation [1, 3, 11] and lack of continuity of finite elements of the standard type of Figs. 2(a)–(d) which are all of continuity class C^0. To obtain an approximation (2) of continuity class C^1 (*i.e.*, first derivatives continuous everywhere in Ω) requires at least a fifth-degree finite element with 21 generalized coordinates [35]! Since the normal derivative $\nabla_N\mathbf{u}$ of (2) is discontinuous, it follows that $F_N(\mathbf{u}, \nabla\mathbf{u})$ is discontinuous across the edges of Ω^l. It follows that the total current

$$I_j^l = \int_{\partial\Omega_j} F_N(\mathbf{u}, \nabla\mathbf{u})\, d\partial\Omega \quad (32)$$

leaving the side $\partial\Omega_j^l$ of Ω^l is different from the current $-I_j^{l1}$ entering the same side of adjacent element Ω^{l1} (Fig. 4). As described in [36], the quadratic element of Fig. 2(b) can be modified so that the currents I_j^l are continuous by introducing the mappings

$$m_i^l = \alpha_i^l , \quad (33a)$$

$$m_{i+3}^l = \int_{\partial\Omega_i} F_{Ni}(\mathbf{u}, \nabla\mathbf{u})\, d\partial\Omega , \quad (33b)$$

$$i = 1, 2, 3 . \quad (33c)$$

Equations (33) constitute a mapping $\boldsymbol{\mu}(\boldsymbol{\alpha}^l)$ whose Jacobi-

an $\partial\mu/\partial\alpha = \mathbf{M}$ can be determined by the methods of Sections 3 and 4 of this paper.

7. Sparse-matrix considerations and program efficiency

We now present an analysis of the sparse-matrix data we have compiled to date in applications of the SAFE program. The data are displayed graphically in Fig. 5. Numerical values of the same data are given in Table 1. At the end of the section, this analysis is applied to a brief overview of overall program efficiency. The data of Fig. 5 express the dependence of the multiplication count of LU factorization upon the following factors:

1. Number (*NSC*) of simultaneous equations,
2. Grid size; *i.e.*, number of elements (*NEL*) and/or degrees of freedom (*NDOF*),
3. Grid elongation; *i.e.*, ratio of x-subdivisions (*NX*) to y-subdivisions (*NY*),
4. Element type; *i.e.*, linear (*NDEG* = 1), quadratic (*NDEG* = 2), etc.,
5. Boundary conditions; *i.e.*, natural, mixed, or Dirichlet,
6. Type of ordering algorithm; *i.e.*, Markowitz (minimum degree heuristic [24]) or George (nested dissection heuristic [25, 26]),
7. Solution method (in SAFE, the options are compiled code; *i.e.*, 123 GNSO [27] or indexed fill code; *i.e.*, SL-MATH [28]).

The method of presentation of Fig. 5 is chosen for easy comparison to an important result about finite-element matrix ordering [25]. This result, obtained by J. A. George, states that nested dissection ordering for a square grid of bilinear elements [Fig. 2(d)] gives an asymptotically optimal multiplication count, *MLDLT*, and factorized nonzero count, *SLDLT*, for Gauss elimination (assuming symmetry of $\partial\mathbf{r}/\partial\alpha$). These counts are asymptotically proportional to powers of a grid parameter, $N_\square$, defined below. In fact, as $N_\square \to \infty$,

$$MLDLT \to 9.5N_\square^3 , \tag{34a}$$

$$SLDLT \to 7.75\, N_\square^2 \ln N_\square . \tag{34b}$$

Figure 5 gives $(MLDLT/N_\square^3)$ for nested dissection ordering of bilinear elements (solid line), minimal degree ordering of bilinear elements (dashed line), and, for comparison, minimal degree ordering of a square grid derived from the conventional "five-point" finite difference approximation of the Laplacian operator (dot-dash line). These three curves have been obtained by Duff, Erisman, and Reid [26].

The abscissa of Fig. 5 is either $N_\square$ or the number of subdivisions $\tilde{N}_\square$ on the side of an "equivalent square grid," defined by

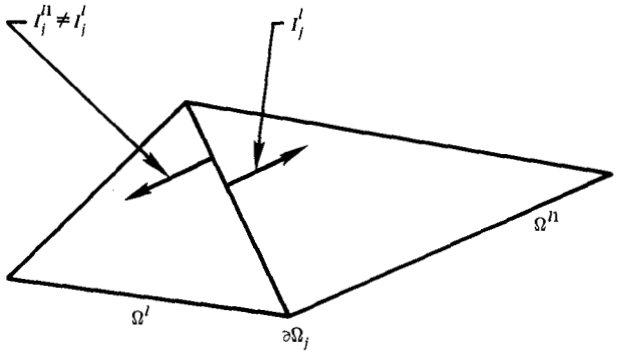


Figure 4 Current discontinuity across element edges.

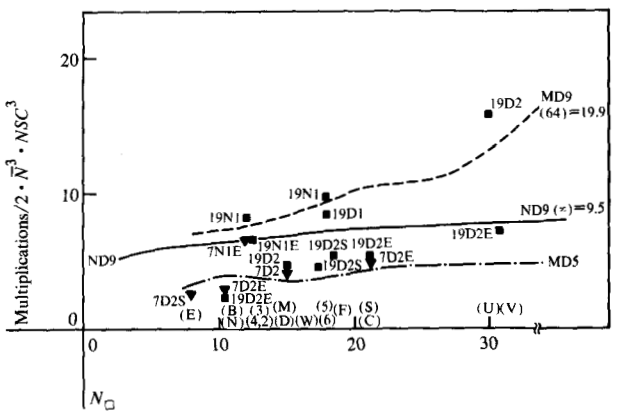


Figure 5 Normalized multiplication counts for various finite-element grids.

$$\tilde{N}_\square = NDEG^2 \cdot NEL/2 . \tag{35a}$$

The factor of 2 accounts for the use of triangular [Figs. 2(a)–(c)] rather than rectangular [Fig. 2(d)] elements. Thus, for an $NX = 14$, $NY = 16$ grid of rectangles subdivided into two linear [Fig. 2(a)] finite elements each,

$$\tilde{N}_\square = \sqrt{(1) \cdot (448)/2} = \sqrt{224} \cong 15 . \tag{35b}$$

Similarly for an $NX = 7$, $NY = 8$ grid of quadratic finite elements,

$$\tilde{N}_\square = \sqrt{4 \cdot (112)/2} = \sqrt{224} \cong 15 . \tag{35c}$$

By this measure, the grids of (35b) and (35c) are equivalent, which is appropriate since the two grids have the same number of nodes (*i.e.*, 225).

The data points represent LU factorization multiplication counts, *MLU*, normalized for comparison to (34a), *i.e.*,

$$MLU \equiv (MLU'/\tilde{N}_\square^3)/2\, NSC^3) , \tag{36}$$

where *MLU'* represents unnormalized data, and where the factor of 2 accounts for the asymmetry of $\partial\mathbf{r}/\partial\alpha$ and the

Table 1 Numerical values of data in Fig. 5.

	NX	*NY*	*NSC*	*NEL*	*Nodes*	*NDOF*	*NDIR*†	$NZ\left(\frac{\partial \mathbf{r}}{\partial \boldsymbol{\alpha}}\right)$	*Fill-in*	*Mults* (LU)	*Solution time* (s)	*Storage* (kb)	*Setup time* (s)
Linear													
B	7	16	2	224	136	272	58	2 694	3 336	49 126	6		7
C	14	32	2	896	495	990	110	11 694	32 528	737 934	23		54
D	14	16	2	448	255	510	78	5 654	11 696	224 750	10		14
E	(special)		2	152	94	188	14	1 926	1 840	25 622	4		4
4	12	12	1	288	169	169	0	*	*	22 935	*	75.6	*
Quadratic													
2	12	3	1	72	175	72	0	*	*	21 804	*	*	*
3	6	6	1	72	169	72	0	*	*	28 812	*	87.5	*
5	9	9	1	162	361	361	0	*	*	124 878	*	350	*
6	9	9	1	162	361	361	53	*	*	101 394	*	288	*
F	(special)		2	170	377	754	62	14 606	29 215	939 809	15		33
M	7	8	2	112	255	510	78	8 742	?	245 054	11.5		14
N	7	4	2	56	135	270	62	3 894	?	47 566	5.5		5
S	7	16	2	224	995	990	110	18 438	30 320	856 974	19		50
U	14	16	2	448	957	1914	150	38 190	111 550	5 068 214	45		458
V	7	32	2	448	975	1950	174	37 830	84 728	2 893 937	29		186
W	(special)		2	152	339	678	62	12 918	23 714	478 414	13		24

*Cf. Ref. [12].
†"NDIR" stands for the number of generalized coordinates specified *a priori* by Dirichlet B.C. [*cf.* (4)].

factor NSC^3 accounts for the fact that in most cases the NSC unknowns present at each nodal point can be thought of as a single block of NSC unknowns (*cf.* the Hypermatrix method of [37]), represented by a $NSC \times NSC$ block in the Jacobian $\partial \mathbf{r}/\partial \boldsymbol{\alpha}$. In this case one block multiplication is equivalent to NSC^3 real multiplications.

In Fig. 5, the dashed curve is labeled MD9 since, for $NSC = 1$, it derives from minimal degree ordering of a matrix with 9 NZ (nonzeros) in each row of $\partial \mathbf{r}/\partial \boldsymbol{\alpha}$ which represents an interior (*i.e.*, nonboundary) node. Similarly, the other two curves are labeled MD5 and ND9 (nested dissection, 9 NZ per row). The data points have labels with four fields. A first field of 7 indicates linear finite elements [Fig. 2(a), 7 NZ per row] and 19 indicates quadratic elements [Fig. 2(b), 19 NZ per row, representing a vertex rather than a midpoint of the triangular element). The second field (D or N) represents boundary conditions, with N signifying all natural boundary conditions [$D(x, y) = 0$ in (1a)], and D signifying that $D(x, y) \neq 0$ for $x, y \in \partial\Omega_D$. The third field signifies the values of NSC, and the fourth field, if present, signifies either a special (nonrectangular, locally refined) or elongated ($NX \geq 2\,NY$). If the fourth field is absent, an essentially square grid is signified. Finally, the characters in parentheses at the bottom of the graph in Fig. 5, distinguished by their vertical alignment, cross-reference the plotted data to the numerical data of Table 1.

We now discuss items 1–7; it is to be understood that in the sequel all data points are the result of Markowitz, *i.e.*, minimum degree, or locally minimum multiplication count ordering of the finite-element equations (8).

Number of simultaneous equations

First, note that all linear element cases with $NSC = 2$ (7_2_ labels) lie well beneath the MD9 line as they should, since for a given $N_\square$, and, hence, for a given total number of unknowns, 7 signifies a sparser matrix than 9. Of course, it must be remembered that the Dirichlet boundary conditions replace many matrix rows with row singletons, so finding the linear data near the MD5 line is not surprising. It is clear, however, that if the multiplication count had a dependency on NSC *stronger* than NSC^3, the $NSC = 2$ cases would be higher on the chart. Thus we may conclude that the assumed proportionality between multiplication count and NSC^3 is appropriate.

Grid size

Notice that as $N_\square \to \infty$, the ND9 line is quite flat (since it asymptotically approaches the value 9.5). In contrast, the experimental data points display a pronounced upward trend, suggesting that the multiplication count might have an $N_\square^4$ component, which supports the conjecture of Duff, Erisman, and Reid [26]. The MD9 line (from [26]) reaches a value of 19.9 at $N_\square = 64$ and shows no sign of

saturation. Note that the high data points at $N_{\square} = 12$, 18, and 30 (Cases 3, 5, 6, M, and U of Table 1) indicate that a rectangular grid of quadratic [Fig. 2(b)] elements has a dependence quite similar to the MD9 line. Of course, the data may ultimately be discovered to reach a level asymptote for large $N_{\square}$, but if it exists, this asymptote must have a value much larger than 9.5. It may be concluded that minimal degree ordering with "first encounter" tie-breaking is inferior to nested dissection for squarish grids, even for other than the bilinear elements for which George's result [Eq. (34)] was obtained. Clearly, it would be worthwhile to find a way to combine the generality of minimal degree ordering with the global, efficient properties of nested dissection.

• *Grid elongation*

The data show that problems with elongated grids (with "E" in the fourth label field) are generally cheaper for a given number of unknowns, and, in addition, have a weaker growth with $N_{\square}$. Compare, for example, Case 3 with Cases 4 and 2, and Case U with Case V. As has been variously observed (see, for example, Rose and Whitten [38]), the nested dissection heuristic is inapplicable to very elongated grids, so it is not surprising that all "E" data points (remember that all data were obtained with minimal degree ordering) lie below the ND9 line.

• *Type of finite element*

The paired data near the MD5 line at $N_{\square} = 10.6$, 15, and 21.2 show the surprisingly weak dependence of multiplication count on original matrix sparsity (7 NZ per row *versus* 19), all other factors being equal. The culprit [*cf.* Table 1 for cases (B *versus* N), (M *versus* D), and (S *versus* C)] is of course fill-in, and it is apparent that after a few eliminations the matrices remaining for original row counts of 7 and 19 are quite similar. This effective equivalence is important because it signifies that the finite elements should be chosen primarily for their ability to approximate the true solutions; *i.e., a priori* sparsity considerations seem to be secondary.

• *Type of boundary condition*

As pointed out above and in [12], when certain generalized coordinates are fixed *a priori* instead of by the Galerkin conditions (3), the corresponding rows and columns of $\partial \mathbf{r}/\partial \boldsymbol{\alpha}$ created by (10) are overwritten by singletons. Thus, for sufficiently small $N_{\square}$, $\partial \mathbf{r}/\partial \boldsymbol{\alpha}$ ends up much sparser with Dirichlet boundary conditions than it does with natural or "Neumann" boundary conditions, so data points with D in the second label field are generally lower than corresponding data with N in the second field. However, the high data point 19D2 shows that elementary volume-to-surface-ratio considerations weaken the effect of the boundary for large $N_{\square}$.

• *Effect of ordering method*

Overall, the preponderance of data points below the ND9 line indicates that our multiplication counts are fairly near optimal. Thus, we are satisfied with minimal degree ordering for all but large effectively square grids. Unfortunately, as discussed in [11], these cases are very important in most applications we have encountered, so it must be concluded that improvements to Markowitz are necessary.

An important aspect of the ordering method is the ordering (setup) time (last column of Table 1). The nested dissection method, which *does not*, in contrast to the minimal degree methods, require a symbolic factorization, would improve substantially on the ordering times listed. Note, however, that it can be expected that the basic Newton step [Eqs. (8)] will be executed many times for each set of boundary values [Eqs. (4)]. Thus, the "ordering time overhead" is usually a worthwhile expenditure.

• *Effect of solution method*

Figure 5 relates only to the *ordering* of the finite-element equation (8). When *solving* (8), the solution method chosen determines the CPU time and storage required for a given matrix and concomitant operations count. In SAFE the options are the compiled code approach, *i.e.*, 123 GNSO [27] and the indexed fill approach, *i.e.*, SL-MATH [28]. Proper comparison of these methods depends strongly on the computer environment, but Gustavson has shown [39] that compiled code is two to four times faster and requires about one and one-half to three times as much storage. The data of Table 1 generally support Gustavson's conclusions. Note that the storage requirements are quite large and even in the virtual memory environment of an IBM VM/370, the slower method is generally chosen.

Keeping in mind the considerations of the section on simultaneous equations, we conclude that some combination of the generalized element [31] and Hypermatrix methods [37] would be needed to obtain something close to the speed of the compiled code at storage requirements even smaller than those of indexed fill methods.

Recently, McMullen, Gustavson, and Buturla investigated alternative sparse-matrix methods, including generalized elements, Cholesky-conjugate gradients, and quotient-tree implicit block factorization [40]. They have discovered that for certain problems significant improvements could be made over the SL-MATH approach [28].

• *Overall program efficiency*

Note that there exist applications when as many as 19 volume integration points [*cf.* Fig. 1 and Eq. (10), *NIP* =

19] can be required for engineering accuracy results. We have performed timing breakdowns which show that the time required to evaluate $\partial \mathbf{r}/\partial \boldsymbol{\alpha}$ and $\mathbf{r}(\boldsymbol{\alpha})$ is much smaller than that required to solve linear eqs. (8).

However, many problems exist for which matrix evaluation dominates linear equation solution [40]. We suspect that problems of this type might be more typical, although in the limit of very large problem size we believe the linear equation solution time will dominate. The results of [40] also show that by *not* updating the matrix values except when necessary for convergence, a significant savings in overall solution time can be obtained.

In fact, for moderate-sized problems such as the $N_{\square} = 15$ data of Fig. 5, even $NIP = 19$ integrand evaluation points requires only 1.5 seconds evaluation time for $NSC = 2$, $NEL = 112$, $NX = 7$, $NY = 8$ (which was the case in Fig. 1). In contrast, the SL-MATH matrix solution time was 11.5 seconds for the same problem. If the number of finite elements NEL increases, NIP may be decreased, so the *solution* of the linear equations is generally the dominant factor in SAFE program efficiency. [It is to be noted that the PDEs used to obtain the data in Figs. 1 and 5 represented practical semiconductor applications [1], with complicated expressions for **F** and **c** which involved exponential nonlinearities, *cf.* (17).]

8. Conclusions

We have discussed the computer implementation of a general-purpose program for finite-element analysis of nonlinear, nonvariational PDE. Methods for automated problem specification (*i.e.*, type and form of nonlinearity, boundary conditions, etc.) have been described. Accuracy requirements and computational expense of numerical integration methods have been discussed. A unified treatment of nonstandard "associative" elements such as "current-continuous" and bicubic-spline finite elements has been given and it has been shown how such elements are consistent with the SAFE input language and data structure. Numerical pros and cons of such elements were assessed.

We have shown that despite the stringent numerical integration requirements, overall operation counts are dominated by the sparse-matrix solution of the linear Newton's iteration equations. We have underscored this result by showing (via comparison to nested dissection [25, 26]) that for small to moderate problems (<1000 unknowns) our elimination orderings (Markowitz) are near-optimal. However, we have shown that marked excursions from optimality occur for very large problems. Most of our data were obtained using the indexed fill solution method of the SL-MATH program [28]. The compiled code method [27] is substantially faster but has been shown to require prohibitive time and storage during setup and prohibitive storage during execution. Thus, we have concluded that research leading to some combination of nested dissection and Markowitz for ordering, and "generalized element" [31] and Hypermatrix [37] methods for solution, is required.

Acknowledgments

The authors would like to acknowledge the general technical assistance of F. G. Gustavson, E. M. Buturla, and P. E. Cottrell. Discussions with S. Laux helped us to understand current conservation properties of FEM models. The comments of the reviewers were helpful to us in improving the manuscript.

References

1. G. D. Hachtel, M. H. Mack, and R. R. O'Brien, "Semiconductor Analysis Using Finite Elements—Part II: IGFET and BJT Case Studies," *IBM J. Res. Develop.* **25,** 246-260 (1981, this issue).
2. J. J. Barnes and R. J. Lomax, "Finite Element Methods in Semiconductor Device Simulation," *IEEE Trans. Electron Devices* **ED-24,** 1082-1089 (1977).
3. R. J. Lomax, "Preservation of the Conservation Properties of the Finite Element Method Under Local Grid Refinement," *Comp. Meth. Appl. Mech. Eng.* **12,** 309-314 (1977).
4. R. J. Lomax, "Application of the Finite Element Method to Semiconductor Modeling," *Technical Report No. UM-EPL-014289-T1, NTIS Accession No. PB287729/as*, Electron Physics Laboratory, University of Michigan, Ann Arbor, 1978.
5. E. M. Buturla and P. E. Cottrell, "Simulation of Semiconductor Transport Equations using Coupled and Decoupled Solution Techniques," *Solid State Electron.* **23,** 331-334 (1980).
6. P. E. Cottrell and E. M. Buturla, "Two Dimensional Static and Transient Simulation of Mobile Carrier Transport in a Semiconductor," *Numerical Analysis of Semiconductor Devices (Proceedings of NASECODE I)*, B. T. Browne and J. J. H. Miller, Eds., Boole Press, Dublin, Ireland, 1979, pp. 31-64.
7. E. M. Buturla, P. E. Cottrell, B. M. Grossman, M. B. Lawlor, C. T. McMullen, and K. A. Salsburg, "Three Dimensional Simulation of Semiconductor Devices," *IEEE International Solid State Circuits Conference Digest of Technical Papers*, San Francisco, Feb. 1980, pp. 76-77.
8. E. M. Buturla, P. E. Cottrell, B. M. Grossman, and K. A. Salsburg, "Finite-Element Analysis of Semiconductor Devices: The FIELDAY Program," *IBM J. Res. Develop.* **25,** 218-231 (1981, this issue).
9. T. Adachi, A. Yoshii, and T. Sudo, "Two-Dimensional Semiconductor Analysis Using Finite Element Methods, *IEEE Trans. Electron Devices* **ED-26,** 1026-1031 (1979).
10. S. Selberherr, A. Schutz, and H. W. Potzl, "MINIMOS—A Two-Dimensional MOS Transistor Analyser," *IEEE Trans. Electron Devices* **ED-27,** 1540-1550 (1980).
11. J. J. Barnes, K. Shimohigasha, and R. W. Dutton, "Short Channel MOSFETs in the Punchthrough Current Mode," *IEEE Trans. Electron Devices* **ED-26,** 446-453 (1979).
12. G. D. Hachtel, "The Sparse Tableau Approach to Finite Element Assembly," *Sparse Matrix Computations*, J. R. Bunch and D. A. Rose, Eds., Academic Press, Inc., New York, 1976.
13. J. A. George, "Computer Implementation of the Finite Element Method," Ph.D. Dissertation, Stanford University, Stanford, CA, 1971.

14. G. D. Hachtel, R. C. Joy, and J. W. Cooley, "A New Efficient One-Dimensional Analysis Program for Junction Device Modeling," *Proc. IEEE* **60,** 86–98 (1972).
15. M. S. Mock, "A Two-Dimensional Mathematical Model of the IGFET," *Solid State Electron.* **16,** 601–609 (1973).
16. E. M. Buturla and P. E. Cottrell, "Steady State Analysis of Field Effect Transistors via the Finite Element Method," *Technical Digest of IEEE International Electron Devices Meeting*, Washington, DC, 1975, pp. 51–54.
17. E. A. Wilson and W. E. Tchon, "Calculation of Transfer Potentials Using the Finite Element Methods," 1973 *SWIEEECO Record of Technical Papers*, 25th Annual Southwestern IEEE Conference and Exhibition, Houston, TX, April 1973.
18. J. J. Barnes, R. J. Lomax, and G. I. Haddad, "Finite Element Simulation of GaAs MESFET's with Lateral Doping Profiles and Submicron Gates," *IEEE Trans. Electron Devices* **ED-23,** 1042–1048 (1976).
19. J. K. Reid, Atomic Energy Research Establishment, Harwell, U.K., private communication.
20. J. E. Fromm, "Numerical Solutions of the Nonlinear Equations for a Heated Fluid Layer," *Phys. Fluids* **8,** 1757–1769 (1965).
21. G. L. Guymon and I. P. King, "Application of Finite Element Method to Regional Water Transport Phenomena," *Sparse Matrices and Their Applications*, D. J. Rose and R. A. Willoughby, Eds., Plenum Press, Inc., New York, 1972.
22. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.
23. J. T. Oden, *Finite Elements of Nonlinear Continua*, McGraw-Hill Book Co., Inc., New York, 1972.
24. H. M. Markowitz, "The Elimination Form of the Inverse and its Application to Linear Programming," *Manage. Sci.* **3,** 255–269 (1957).
25. J. A. George, "Nested Dissection of a Regular Finite Element Mesh," *SIAM J. Numer. Anal.* **10,** 345–363 (1973).
26. I. S. Duff, A. M. Erisman, and J. K. Reid, "On George's Nested Dissection Method," *SIAM J. Numer. Anal.* **13,** 686–695 (1976).
27. G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The Sparse Tableau Approach to Network Analysis and Design," *IEEE Trans. Circuit Theory* **CT-18,** 101–113 (1971).
28. *IBM System/360 and System/370, Subroutine Library-MATHematics, User's Guide*, Order No. SH12-5300, available through IBM branch offices.
29. R. K. Brayton, F. G. Gustavson, and G. D. Hachtel, "A New Efficient Algorithm for Solving Differential-Algebraic Equations Using Implicit Backward Differentiation Formulas," *Proc. IEEE* **60,** 98–108 (1972).
30. B. Speelpenning and F. P. Tolman, "The Element Chain Concept in Finite Element Based Programming," Internal Report, BI-72-85, TNO Institute for Applied Scientific Research, Delft, The Netherlands, 1972.
31. B. Speelpenning, "The Generalized Element Method," Preliminary Notices, American Mathematical Society, 73T-C18, Feb. 1973.
32. T. J. Rivlin, *An Introduction to the Approximations of Functions*, Blaisdell, Waltham, MA, 1969.
33. *Theory and Applications of Spline Functions*, T. N. E. Greville, Ed., Academic Press, Inc., New York, 1969.
34. C. deBoor, "BiCubic Spline Interpolation," *J. Math. Phys.* **41,** 212–218 (1962).
35. M. J. L. Hussey, R. W. Thatcher, and M. J. M. Bernal, "On the Construction and Use of Finite Elements," *J. Inst. Math. Appl.* **6,** 263–282 (1970).
36. G. D. Hachtel, M. H. Mack, and R. R. O'Brien, "Semiconductor Device Analysis Via Finite Elements," *Conference Record of the Eighth Asilomar Conference on Circuits and Systems*, Pacific Grove, CA, 1974, pp. 332–338.
37. G. vonFuchs, J. R. Joy, and B. Schrem, "Hypermatrix Solution of Large Sets of Symmetric Positive-Definite Linear Equations," *Comp. Meth. Appl. Mech. Eng.* **1,** 1972.
38. D. J. Rose and G. F. Whitten, "A Recursive Analysis of Dissection Strategies," *Sparse Matrix Computations*, J. R. Bunch and D. J. Rose, Eds., Academic Press, Inc., New York, 1976.
39. F. G. Gustavson, "Some Basic Techniques for Solving Sparse Systems of Linear Equations," *Sparse Matrices and Their Applications*, D. J. Rose and R. A. Willoughby, Eds., Plenum Press, Inc., New York, 1972.
40. C. McMullen, F. G. Gustavson, and E. M. Buturla, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1981, unpublished results.

Received October 10, 1980; revised December 19, 1980

G. D. Hachtel and M. M. Mack are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598; R. R. O'Brien is located at the IBM General Technology Division laboratory, East Fishkill Facility, Hopewell Junction, New York 12533; B. Speelpenning is located at the Hewlett Packard Corporation, Computer Systems Division, Cupertino, California 95014.